MULTICS TECHNICAL BULLETIN MTB-249

TO: Distribution

FROM: Joan Archer Scott

DATE: 16 January 76

SUBJECT: Multics Change Requests

Enclosed are copies of Multics Change Requests which were approved from 16 December 75 through 31 December 75.

Ver. 4
750508

Multics Change Request

MCR 1549
Page 1 of 1

TITLE: vfile_ utilities modified for FAST

AUTHOR: M. Asherman

STATUS	DATE
Written	24 November 75
Status	P 12/2/75 A 12/16/75
Expires	06/16/76

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? [x] yes [ ] no
-Incompatible change? [ ] yes [X] no
-Performance: [ ] Better [X] Same [ ] Worse
-Replaces MCR 1443

	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.)	Subroutines, vfs_ Commands, vfa, vfs, abc
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	(vfa, vfs).info
Other (Name)	
None (Reason)	

Objections/Comments:

Documentation Attached

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

SUMMARY:

Add the commands vfile_status (vfs) and vfile_adjust (vfa), and the subroutine vfile_status_ for use with storage system files as supported by vfile_.

REASONS:

User convenience.

Subroutine required for FAST.

IMPLICATIONS:

Availability of basic file info (type and statistics) in addition to that provided by the status command.

Files left in an inconsistent state by interrupted opening can be adjusted.

DETAILED PROPOSAL:

See MPM documentation attached.

adjust_bit_count

Name: adjust_bit_count, abc

The adjust_bit_count command is used to set the bit count of segments that for some reason do not have the bit count set properly (e.g., the program that was writing the segment got a fault before the bit count was set, or the process terminated without the bit count being set, etc.). The adjust_bit_count command looks for the last nonzero 36-bit word or (if specified) the last nonzero character in the segment and sets the bit count to indicate that the word or character is the last meaningful data in the segment.

Usage

adjust_bit_count paths -control_args-

where:

1. paths — are the pathnames of segments for which the bit count is to be adjusted.
2. control_args — are as follows and apply to all path arguments:
   - -character, -ch — set the bit count to the last nonzero character.
   - -long, -lg — print a message when the bit count of a segment is changed, giving the old and new values.

Notes

If the bit count of a segment could be computed but could not be set (e.g., the user had improper access to the segment), the computed value is printed so that the user can use the set_bit_count command after resetting access or performing other necessary corrective measures. See the description of the set_bit_count command.

The user must have write permission on the segment whose bit count is being adjusted. He need not have modify permission on the directory containing that segment.

The adjust bit count command should not be used on segments in structured files. The adjustments to all file types supported by vfile_ may be made with the vfile_adjust command.

Names: vfile_adjust, vfa

This command is used to adjust structured files left in an inconsistent state by an interrupted opening, or unstructured files in any state. For unstructured files a control_arg must specify the desired adjustment. Otherwise, no control_args are allowed. A sequential or blocked file is adjusted by truncation after the last complete record. An indexed file is adjusted by finishing the interrupted operation.

Usage

vfile_adjust path -control_arg-

Where:

1.	path	is the pathname of the file to be adjusted.
2.	control_arg	must be specified only for unstructured files and is selected from the following:
	-set_nl	if the last non-zero byte in the file is not a newline character, a new line character is appended. The bitcount of the file's last nonempty segment is then set to the file's last nonzero byte (which is now sure to be a newline character).
	-use_nl	The file is truncated after the last newline character.
	-set_bc	the bit count of the file's last nonempty segment is set to the last nonzero byte in that segment. Any components beyond it are deleted.
	-use_bc -n-	the file is truncated to the byte specified by the bit count of msf component n. If n is not given, it is taken to be the last nonempty segment.

Notes

See the description of the vfile_ I/O module for further details. The command adjust_bit_count used with the -ch control_arg is equivalent to vfile_adjust with the -set_bc control_arg, except that the latter will only operate on a file which appears to be unstructured.

Name: vfile_status, vfs

This command prints the apparent type (unstructured, sequential, blocked, or indexed) and length of storage system files. For structured files, information about the state of the file (if busy) and the file version (unless current) is printed. The maximum record length is printed for blocked files. For indexed files, the following statistics are also provided:

1. The number of records in the file, including zero length records.
2. The number of non-null records in the file, if different from the above.
3. The total length of the records (bytes).
4. The number of blocks in the free space list for records.
5. The height of the index tree (=0 for empty files).
6. The number of nodes (each 1k words, page aligned) in the index tree.
7. The total length of all keys (bytes).

Usage

vfile_status path

where path is the pathname of the segment or multisegment file of interest. If the entryname portion of the pathname denotes a directory, it is ignored. If no files are found for the given pathname, a message to that effect is printed. If the entry is a link, the information returned pertains to the entry to which the link points. The star convention is permitted.

Notes

Additional information may be obtained through the status command.

Examples:

Assume that the file foo is in the user's working directory. The command:

```
vfile_status  foo
```

might produce the following output:

```
type:   unstructured
bytes:   4993
```

if the file is unstructured,

or

```
type:   sequential
records:   603
```

if the file is sequential,

or

```
type:   blocked
records:   1200
max recl:   7 bytes
```

if the file is blocked,

or

```
type:   indexed
records:   397
state:   locked by this process
action:   write in progress
record bytes:   3970
free blocks:   1
index height:   2
nodes:   3
key bytes:   3176
```

if the file is indexed and a write operation has been interrupted in the user's process.

Name: vfile_status_

This subroutine returns various items of information about a storage system file supported by the vfile_ I/O module.

Usage:

```
declare vfile_status_ entry (char(*), char(*), ptr, fixed(35);

call vfile_status_ (dir_name, ent_name, info_ptr, code);
```

where

1. dir_name is the pathname of the containing directory. (Input)
2. ent_name is the entryname of the file of interest. If the entry is a link, the info returned pertains to the entry to which it points. (Input)
3. info_ptr is a pointer to the structure in which information is to be returned - see File Information below. (Input)
4. code is a storage system status code. (Output)

File Information:

The info_ptr argument points to one of the following self-describing structures:

```
dcl  1  uns_info based (info_ptr),
     2  info_version fixed,
     2  type fixed,
     2  bytes fixed(34),
     2  flags aligned,
        3  pad1 bit(2) unal,
        3  header_present bit(1) unal,
        3  pad2 bit(33) unal,
     2  header_id fixed(35);

dcl  1  seq_info based (info._ptr),
     2  info_version fixed,
     2  type fixed,
     2  records fixed(34),
     2  flags aligned
        3  lock_status bit(2) unal,
        3  pad bit(34) unal,
     2  version fixed;

dcl  1  blk_info based (info_ptr),
     2  info_version fixed,
     2  type fixed,
     2  records fixed(34),
     2  flags aligned
        3  lock_status bit(2) unal,.
        3  pad bit(34) unal,
     2  version fixed,
     2  action fixed,
     2  max_rec_len  fixed(21);

dcl  1  indx_info based (info_ptr),
     2  info_version fixed,
     2  type fixed,
     2  records fixed(34),
     2  flags aligned
        3  lock_status bit(2) unal,
        3  pad bit(34) unal,
     2  version aligned,
        3  file_version fixed(17) unal,
        3  program_version fixed(17) unal,
     2  action fixed,
     2  non_null_recs fixed(34),
     2  record_bytes fixed(34),
     2  free_blocks fixed,
     2  index_height fixed,
```

```
        2  nodes fixed,
        2  key_bytes fixed(34),
        2  change_count fixed(35),
        2  reserved(4) fixed;
```

Where:

0.	info_version	identifies the version of the info structure; this must be set to 1 by the user. (Input)
1.	type	identifies the file type: 1 unstructured 2 sequential 3 blocked 4 indexed
2.	lock_status	if zero, indicates that the file's lock is not set; otherwise the file is busy "01"b busy in caller's process "10"b busy in another process "11"b busy in a defunct process
3.	records	is the number of records in the file, including those of zero length.
4.	header_present	if set, indicates that an optional header is present.
5.	header_identifier	contains the identifier from the file's header, if present. Its meaning is user-defined.
6.	bytes	gives the file's length in bytes, not including the header.
7.	max_rec_len	is the maximum record length in bytes, associated with the file.
8.	version	identifies the version number of the file and its creating program.
9.	action	if non-zero, this indicates an operation in progress on the file: -1 write in progress -2 rewrite in progress -3 delete in progress +1 truncation in progress

10.	record_bytes	is the total length of all records in the file, in bytes.
11.	free_blocks	is the number of blocks in the file's free space list for records.
12.	index_height	is the height of the index tree (=0 if file is empty)
13.	nodes	is the number of single page nodes in the index.
14.	key_bytes	is the total length of all keys in the file, in bytes.
15.	non_null_recs	is a count of the records in the file, not including those of zero length.
16.	change_count	is incremented after each file modification.

Notes:

The user must provide the storage space required by the above structures.

See the description of the vfile_ I/O module for further details.

9/10/75 vfile_status,vfs

Function: this command prints the apparent type and length of storage system files. Additional info is provided for structured files.

Syntax: vfile_status pathname

Arguments: pathname is the path name specifying the file of interest. The star convention is permitted.

Notes: for structured files (sequential,blocked, or indexed), the state of the file is printed (if busy). The following statistics are also provided for indexed files:

1. the number of records in the file, including those of zero length
2. the number of non-null records, if different from the above
3. the total length of the records (in bytes)
4. the number of blocks in the free space list for records
5. the height of the index tree (zero for empty files)
6. the number of nodes in the index (each occupies a single 1K page)
7. the total length of all keys (bytes)

References: additional information about a file may be obtained with the status command. See documentation of the vfile_ I/O module for further details.

r 1356 0.545 0.720 29

pr vfa.info

vfile_adjust.info 11/24/75 1355.9 mst Mon

9/10/75 vfile_adjust,vfa

Function: this command adjusts a storage system file which may have been left in an inconsistent state by an interrupted opening.

Syntax: vfile_adjust pathname -control_arg-

Arguments: pathname is the path name of the file to be adjusted.

Control arguments: (one specified if and only if file is unstructured)
-set_nl: append a newline char if file does not end with one
-use_nl: truncate file after last new_line character
-set_bc: set bitcount and truncate at last non-zero byte in the file
-use_bc -n-: truncate at byte specified by bitcount of component n (last non-zero component if n not specified)

Notes: a sequential or blocked file is adjusted by truncation after the last complete record. An indexed file is adjusted by completing any interrupted operation.

References: the condition of a file may be determined with the vfile_status command. See documentation of the vfile_ I/O module for further details.

r 1357 0.430 0.692 38

Ver. 4
750508

Multics Change Request

MCR 1559
Page 1 of 1

TITLE: Add "error status" control order to indexed files

AUTHOR: M. Asherman

STATUS	DATE
Written	8 December 75
Status	A 12/16/75
Expires	06/16/76

DOCUMENTATION CHANGES

-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.

Document	Specify One or More
MPM (Vol, Sect.)	
PLMS (AN #) 57 (vfile_)	
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	
Other (Name)	
None (Reason)	

Objections/Comments:

This change is required for new sys_err_ log.

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

SUMMARY:

Add new control operation to provide user with position information after an attempt to position beyond end or beginning of file.

REASONS:

There is currently no easy way to determine the number of records which precede or follow a given entry in an indexed file.

IMPLICATIONS:

The user can easily find the ordinal position of any record in a file.

DETAILED PROPOSAL:

See documentation attached.

Control Operation: "error_status"

The order "error_status" is accepted when I/O switch is open for keyed_sequential_input or keyed_sequential_update. For this order the info_ptr argument must point to a structure of the following form:

```
dcl  1  error_info based(info_ptr),
     2  version fixed,
     2  type fixed,
     2  requested fixed(34),
     2  received fixed(34);
```

The operation returns information about the most recent attempt to position beyond either end of file since the file was opened. If an error of this type has occurred, error_info.type is set to 1, and error_info.requested is set to the value of the position skip argument which caused the error. In this case error_info.received is set to the actual number of position skips which successfully took place before attempting to go beyond end of information. If no such errors have occurred, error_info.type is set to zero.

The variable error_info.version is provided for compatibility, and must be set to 1 by the user.

Ver. 3
741022 MULTICS CHANGE REQUEST

MCR 1563

TITLE: Allow additional position requests in io_call

AUTHOR: Larry Johnson

STATUS	DATE
Written	12/5/75
Status	A 12/16/75
Expires	06/05/76

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) cmds
MPAM (sect)
MOSN (sect)
MSAM (sect)
PLMs (AN#)
Info Segs
Other

CATEGORY (check one)

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

OBJECTIONS/COMMENTS:

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

Summary

io_call currently allows only -1(bof), 0, and +1(eof) on position operations. This is not sufficient.

Reasons

vfile_ has implemented position type 2.

Detailed Proposal

Allow io_call to accept any numeric position request instead of only -1, 0, and +1. An I/O module must check this code anyway, so invalid requests will still be caught. The next argument (amount of positioning) is currently required on position 0 and not allowed with +1 or -1. This will not be changed, but this argument will be made optional on other requests, with 0 being the default.

io_call position switchname type -n-

where:

1. type — may be one of the following:

   bof, -1 — set position to beginning of file

   eof, 1 — set position to end of file

   forward, 0 — set position forward n records or lines, where n is the next argument.

   reverse — set position back n records (same as forward -n records)

   othern — any other numeric argument may be specified, but its function depends on the I/O module being used and may not be implemented for all I/O modules.

2. n — is a decimal integer. It must be present if type is forward, reverse, or 0; it must be omitted if type is bof, eof, 1, or -1; for any other position request, it is optional and will default to 0 if omitted.

Ver. 4
750508

Multics Change Request

MCR 1564
Page 1 of 3

TITLE: Improve add_copyright

AUTHOR: J. Gintell, M. Meer

STATUS	DATE
Written	9 December 75
Status	A 12/16/75
Expires	06/16/76

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)	

-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 620

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

Objections/Comments:

The tool copyright_archive should be updated accordingly

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

SUMMARY:

The add_copyright command which is used to implement and enforce the Multics copyright convention should be upgraded to allow the specification of other than the "joint" copyright and to place the current year in the copyright notice.

REASONS:

The Multics copyright notice is a comment or set of comments embedded in the source program at "the beginning". It commences with a line of *'s and after several blank lines contains the string:

Copyright (c) yearlist by owner

where yearlist is the set of years (separated by commas) during which changes were made and installed, and owner is the appropriate member of the set:

-Massachusetts Institute of Technology
-Honeywell Information Systems, Inc.
-both joined with "and"
-or a string entered by the user.

The add_copyright will be modified to save the dates on the existing copyright and add the current year to them if the "name" or "owner" remains the same. If the "name" or "owner" differs, then the old copyright is discarded and a new one is inserted with the current year.

A subroutine interface will be added that will return the copyright information so that it can be used as part of a report preparing program to summarize the status of a library of programs.

An optional year argument will be added with default being the current year.

IMPLICATIONS:

This change is an extension to that implemented by the current version of the add_copyright command which management people have agreed is adequate.

This command does not implement the policy of choosing which owner to place in the copyright notice but leaves that decision up to the user of the command.

DETAILED PROPOSAL:

Name: add_copyright

Usage: add_copyright pathname -control arguments-

pathname	is the pathname of the segment whose copyright notice is to be checked or modified.

Control arguments:

-check	print a message stating what the notice would contain and what was changed.
-his	make the owner be HIS.
-mit	make the owner be MIT.

-joint — make the owner be MIT and HIS.

-name — make the owner be that given in the next argument string.

-year — next argument is a year or series of years separated by commas to be inserted in the copyright.

There is no default for the owner. The default for the year is the current year.

Name: return_copyright_

Usage: call return_copyright_ (segptr, length, owner, year, switch);

segptr — is a pointer to the segment to be checked. (Input)

length — is the length (in characters) of the segment. (Input)

owner — is a varying character string giving the control argument used to specify the owner. (Output)

year — is a varying character string giving the year(s). (Output)

switch — is a fixed binary field

= 0 if a copyright is present
= 1 if there is no copyright (Output)

Ver. 3
741022 MULTICS CHANGE REQUEST MCR 1565

TITLE: changes to copy_file control arguments

AUTHOR: Ross E. Klinger

STATUS | DATE
Written | 12/03/75
Status | 12/16/75
Expires | 06/03/76

Planned for System: not applicable
Fixes Bug Number(s): not applicable
Documented in MTB: 222, amended
Incompatible Change: yes
User/Operations-visible Interface Change: no
Coded in: (■)PL/I ( )ALM ( )other-see below
Performance: ( )better (■)same ( )worse

CATEGORY (check one)
( )Lib. Maint. Tools
( )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) II, 3 MPAM (sect)
MOSN (sect) MSAM (sect)
PLMs (AN#) 57
Info Segs copy_file.info
Other

OBJECTIONS/COMMENTS:

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

SUMMARY: change the -input_file and -output_file control arguments to -input_description (-ids) and -output_description (-ods).

REASONS: these control arguments specify iox_ attach descriptions, not pathnames. current usage of -input/output_file is to specify a file's pathname. commands should use identical control arguments in identical contexts, as much as possible.

IMPLICATIONS: none, coding has not yet begun. N.B. - the appropriate person responsible for maintaining the list of control arguments should be informed.

Ver. 3
741022 MULTICS CHANGE REQUEST

MCR 1566

TITLE: Modify status command for new storage system

AUTHOR: VanVleck

STATUS | DATE
Written | 12/08/75
Status | A 12/16/75
Expires | 06/08/76

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: (■)PL/I ( )ALM ( )other-see below
Performance: ( )better (■)same ( )worse

DOCUMENTATION CHANGES (specify one or more)
MPM (vol,sect) ag92 MPAM (sect)
MOSN (sect) MSAM (sect)
PLMs (AN#)
Info Segs
Other

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

OBJECTIONS/COMMENTS:

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

SUMMARY:

Remove the error message printed if the device id returned from status_ is 0. In the new storage system it is always 0.

REASONS:

This change allows the status command to be usable with both old and new storage systems.

IMPLICATIONS:

none

750508

Multics Change Request

MCR 1557
Page 1 of 1

TITLE: Changes to sort_items_ and sort_items_indirect_

AUTHOR: C. D. Tavares

STATUS	DATE
Written	03 December
Status	P 12/9/75 A 12/23/75
Expires	06/23/76

-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? [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 Cmnd/Subr.

DOCUMENTATION CHANGES

Document	Specify One or More
MPM (Vol, Sect.)	51, System Tools
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 new entrypoints to sort_items_ and sort_items_indirect_ to perform sorting of varying character strings.

REASONS:

Submission of graphic editor for MR3.1 requires facility for the sorting of varying character strings. Inclusion of this code into sort_items_ was both facile, and useful to users in the general case. Changes to sort_item_ indirect_ were made on recommendation of the original implementor, to keep both modules on an organized "package" basis.

IMPLICATIONS: None apparent

DETAILED PROPOSAL:

Entrypoint "...$varying_char" to be added to each module, in keeping with the established entrypoint convention used. Coding has been submitted. (i.e. is "ready for submission").

Entry: sort_items_indirect_$~~char~~ varying-char

varying This entry sorts a group of information fields, which are ~~fixed length unaligned~~ character strings into ASCII collating sequence by reordering an index array. The elements of this index array are indices into an array of pointers to the character strings in the group.

Usage

```
declare sort_items_indirect_$~~char~~ entry (ptr, ptr);
    ~~fixed bin (24)~~;
call sort_items_indirect_$~~char~~ (vP, ~~index~~, ~~length~~);
```

varying-char

varying-char

iP

where:

1. vP — points to a structure containing an array of unaligned pointers to the ~~unaligned~~ ~~fixed-length~~ varying character strings to be sorted. (Input)

2. iP — points to the structure into which the ordered array of fixed binary (24) indices into the unaligned pointer array will be placed. (Input)

~~3. length — is the number of characters in each string. (Input)~~

note: It seems that this "index" should read "iP" in all of the other entry points for this module also.

Entry: sort_items_$varying_char

This entry sorts a group of Varying character strings into ASCII collating sequence by reordering a pointer array whose elements point to the character strings in the group.

Usage

```
declare sort_items_$varying_char entry (ptr);
call sort_items_$varying_char (vP);
```

where:

1 vP points to a structure containing an array of unaligned pointers to the Varying character strings to be sorted. (Input)

Ver. 4
750508

Multics Change Request

MCR 1562
Page 1 of 2

TITLE: Change ep_basic for FAST

AUTHOR: M. Weaver

STATUS	DATE
Written	8 December 75
Status	P 12/10/75 A 12/23/75
Expires	06/23/76

-Coded in: [X] PL/I [X] ALM [ ] other- explain in DETAILED PROPOSAL
-Planned for System MR 3.1
-Fixes Bug Number(s) see list
-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.
X	Runtime
X	User Cmnd/Subr.

DOCUMENTATION CHANGES

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

Objections/Comments:

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

SUMMARY:

Change extended precision basic and its runtime so they support Basic in the FAST subsystem (as well as extended precision Basic via the use_ep_basic command).

Principal changes:

1. Use vfile_ for random files.
2. single or double precision
3. interface with FAST run-unit (library and chain statements)

Error messages will be improved and minor bug fixes made as time permits.

REASONS:

Support both single and double precision in FAST with a single integrated package. Make Basic random files compatible with corresponding Fortran types.

IMPLICATIONS:

Random string and random numeric files will be incompatible with current basic files and, for a while, with files created by the

non-FAST basic. Also, a program cannot access both single and double precision random numeric files. Programs containing library or chain statements will not run correctly outside of [the DTSS compatible] FAST.

If performance is satisfactory, ep_basic should be used for Multics single precision basic (Release 4.0?). Otherwise basic should be changed to use the same file types and support the same language (except library and chain statements).

DETAILED PROPOSAL:

Two new entries are being added to basic_ and the current main entry is being redefined (see attached documentation). Most of the work for the library and chain statements is actually done by the FAST run unit manager. See the attached appendix to the FAST manual, which is the user documentation. The setdigits statement and the "apo" abbreviation for the asc function already exist in extended precision basic. Some other minor changes include:

1. on an input error, the message will make more clear where to retype from,
2. the per function will check the files write access for print, write and scratch.

Amendment and Note for MCR 1562 (Postponed)

Additional Details:

Both basic and ep_basic (i.e. their runtime routines) will be changed to use the new random file formats. Thus in 3.1 everyone will use the same file formats.

Conversion to the new formats will be automatic (access, etc. permitting). Commands will be added to handle the old to new conversions and conversions between sp and dp in either direction. See the attached info files.

Note: In Release 3.1 there will be 3 levels of the Basic Language:

1. FAST Basic
2. ep_basic in Multics: This is (1) less the library and chain statements.
3. basic in Multics: This is (2) less the set_digits statement.

There will be two implementations of Basic:

ep_basic implements (1) and (2)
basic implements (3)

There is also an option as to whether the arithmetic is single or double precision. In FAST this works in neatly via the "system" concept. In Multics it is not so neat.

The following steps will be taken (probably for 4.0) to improve things:

A. Add the set_digits statement to the Multics language, thus leaving only the library and chain statements as language differences.

B. If performance is satisfactory use the ep_basic implementation for all languages. (Performance is the only problem.)

C. Try and improve the basic_ep_basic user interface in Multics or admit that it is good enough.

Then ep_basic can be removed from the list of installed approved software.

The formats of basic random string and numeric files are being changed soon. These changes do not affect the content of the files but only the way they look to the basic I/O system. The new formats are incompatible with the current formats in that they cannot be used on the current system and the old formats cannot be used by the new basic runtime. However, to ease the transition, the new basic runtime will (for awhile) automatically convert any old format file it sees to a new format file if access and quota permit, print a message and continue processing.

It is recommended that basic users convert their random files before using them with the new basic runtime by typing

convert_old_basic_file file_pathname

which will replace the file.

12/17/75 convert_numeric_file

Function: convert_numeric_file converts random numeric files used by basic and fortran programs from single to double precision and vice versa using PL/I conversion rules.

Syntax: convert_numeric_file old_path new_path -control_arg-

Arguments:
1) old_path is the pathname of the original file
2) new_path is the pathname of the comverted file

Control Arguments:
-double_precision, -dp: convert from single to double precision; default
-single_precision, -sp: convert from double to single precision

MCR 1562

Entry: basic_

This entry compiles a basic program. It does not implement library or chain statements.

Usage

```
declare basic_ entry (ptr, ptr, fixed bin, bit(1)
                                  aligned, ptr, fixed bin);

call basic_ (source_info_ptr, output_ptr, output_length,
                        go_mode, main_ptr, err_count);
```

1) source_info_ptr points to the source info structure (see Note below). Input

2) output_ptr points to the output (must be 0 mod 2). Input

3) output_length is the length of the output in words. Output

4) go_mode is "1"b if a complete object segment is not desired; is "0"b if a complete object segment is desired. Input

5) main_ptr points to the entry of the main program Output

6) err_count is the number of errors found during compilation. Output

Entry: compile

This entry compiles a basic program, always generating an object segment. Library statements are converted to a list stored in the object segment.

Usage:

```
declare basic_$compile entry (ptr, ptr, fixed bin,
                                  fixed bin(35));

call basic_$compile (source_info_ptr, output_ptr,
                        output_length, code);
```

1) source_info_ptr points to the source info structure (see Note below). Input

2) output_ptr points to the output (must be 0 mod 2). Input

3) output_length is the length of the output in words. Output

4) code is a system status code. Output

Entry: run_unit_compiler

This entry compiles a basic program, always generating an object segment. It interacts with its caller to process library statements, obtaining all source libraries before returning.

Usage:

```
declare basic_$run_unit_compiler entry (ptr, ptr,
          fixed bin, bit(1) aligned, entry, entry,
          fixed bin(35));

call basic_$run_unit_compiler (source_info_ptr,
          output_ptr, output_length, debug_sw,
          get_next_source_seg_, add_to_lib_list_, code);
```

1) source_info_ptr — points to the source info structure (see Note below). Input

2) output_ptr — points to the output (must be 0 mod 2). Input

3) output_length — is the length of the output in words. Output

4) debug_sw — is "1"b if the output program is to be run with probe. Input

5) get_next_source_seg_ — is the entry to call to get more source. Input

6) add_to_lib_list_ — is the entry to call to pass the names of libraries to the caller. Input

7) code — is a system status code. Output

Note

The structure pointed to by source_info_ptr above contains information to locate the source and to fill in the source map in the object segment. It is described as follows:

```
declare   1 source_info aligned based,
          2 input_pointer ptr,
          2 input_lng fixed bin(21),
          2 dirname char(168) varying,
          2 segname char(32) varying,
          2 date_time_modified fixed bin(71),
          2 unique_id bit(36);
```

1) input_pointer — points to the source program.

2) input_lng — is the length of the source in characters.

3) dirname — is the name of the source sement's directory.

4) segname — is the segment name of the source segment.

5) date_time_modified is the date-time modified of the source segment.

6) unique_id is the unique id of the source segment.

This file contains a list of all bugs in the Multics BASIC compiler.
Bugs marked with ! have been fixed in the installed basic.
Bugs marked with * have been fixed in the basic located in >ldd>exl>o.

052 next free number
061 75.12.08 gosub returns after on x gosub... cause error
060 75.11.21 sometimes use of complicated expressions in output statements causes succeeding references or functions in the statement to be incorrectly compiled
059 75.10.10 reset as first operation on existing file gets confusing error message
058*75.07.30 for ... to s step ... gets compiler error message
057*75.07.10 local arrays in subprograms can't be accessed

APPENDIX C

FAST BASIC

The following changes have been made to Multics BASIC (as described in AM82, Rev. 0):

1. The library statement is supported and has the form:

   library <list of segment names>

   The library statement lists the pathnames of segments containing subprograms to be used by the program. The names are enclosed quotation marks and separated by commas. There are no passwords.

2. The setdigits statement is supported to provide a means of controlling dynamically the number of digits in a numeric value that may be printed as output. It has the form:

   setdigits formula

   The value expressed by the formula in the statement is truncated to its integer value. That integer value represents the number of print columns that will be utilized by all future print statements until another setdigits statement is executed, or until program execution terminates. One to 19 printed columns may be specified.

   In addition to the specified number of digits, the sign of the number is printed. An exponent is also printed if all digits to the left of the decimal point cannot be contained in the number of digits expressed by the formula. This statement is valid only for double precision programs.

3. The chain statement is supported and has the form:

   chain <string expression or file reference>
   system <string expression> with <file reference list>

The chain statement terminates running of the current program and begins execution of the program specified. Neither the current program nor any caller is returned to again and the values of variables in the new program are completely reset. The only communication between the current and new programs consists of the specified files, which remain open. The specification of a system is optional; if it is omitted, the current system is assumed. The list of files to be passed is also optional.

4. The characters "-" and "." are now allowed in subprogram names.

5. The requirement that a $ used in a format statement as a field delimiter must be followed by "+" or "-" is dropped. In this case, "-" is assumed.

6. The asc function recognizes the abbreviation "apo" to mean apostrophe.

ver. 4
750508

Multics Change Request

MCR 1567
Page 1 of 1

TITLE: Changes to "vfd" -pseudo-op in ALM

AUTHOR: Eugene E. Wiatrowski

STATUS	DATE
Written	12 December 75
Status	A 12/23/75
Expires	06/23/76

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? [ ] 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.

Document	Specify One or More
MPM (Vol, Sect.)	(SWG, 6)
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:

Extend the implementation limit on number of words that can be contructed with a single usage of the "vfd" -pseudo-op.

REASONS:

The present limitation of ten words can be somewhat cumbersome, especially in "macro-expansion" situations.

PROPOSAL:

Extend the implementation limit from 10 to 128 words.

use name
assembles subsequent code into the location counter name. The default location counter is ".text.".

vfd T1L1/expression1,T2L2/expression2,...,TnLn/expressionn
is variable format data. Each expressioni is of type Ti and is stored **in the** next Li bits of storage. As many words are used as required. **Individual** items can cross word boundaries and exceed 36 bits in **length.** Type is indicated by the letters "a" (ASCII constant) or "o" **(logical** expression) or none (arithmetic expression). Regardless of **type,** the low-order Li bits of data are used, padded if needed on the **left.** The Ti can appear either before or after Li.

Restrictions: The total length of the variable format data cannot **exceed** ~~10~~ 128 words. A relocatable expression cannot be stored in a field **less than** 18 bits long, and it must end on either bit 17 or bit 35 of **a word.**

ok or

zero expression1,expression2
assembles expression1 into the left 18 bits of a word and expression2 **into the** right 18 bits. Both subfields default to zero.

Ver. 4
750508

Multics Change Request

MCR 1568
Page 1 of 1

TITLE: Change bound_parse_.bind (bindfile)

AUTHOR: R. Schoeman

STATUS	DATE
Written	12 December 75
Status	A 12/23/75
Expires	06/23/76

DOCUMENTATION CHANGES

-Coded in: ☐ PL/I ☐ ALM ☒ other- explain in DETAILED PROPOSAL
-Planned for System MR
-Fixes Bug Number(s) unnumbered
-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 Cmnd/Subr.

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.**

SUMMARY:

Retain the name clean_up in the bound_parse_ bindfile.

REASONS:

At present there is a bug. The MPM documentation says that the command pl1$clean_up can be executed, but since the name clean_up is not retained, it cannot be.

DETAILED PROPOSAL:

The language of a bindfile is its very own special language.

ver. 4
750508

Multics Change Request

MCR 1569
Page 1 of 1

TITLE: Add abs_seg keyword to generate_mst

AUTHOR: B. Greenberg

STATUS	DATE
Written	15 December 75
Status	A 12/23/75
Expires	06/23/76

-Coded in: [X] PL/I [ ] ALM [ ] other- explain in DETAILED PROPOSAL
-Planned for System MR 4.0
-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 Cmnd/Subr.

DOCUMENTATION CHANGES

Document	Specify One or More
MPM (Vol, Sect.)	
PLMS (AN #)	51, 60
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 keyword "abs_seg" to generate_mst, to turn on slte.abs_seg, a new bit.

REASONS:

The New Storage System needs a semblance of coherence with regard to which supervisor segments represent real data, and which are used to address core/disk.

IMPLICATIONS:

None.

Ver. 4
750508

Multics Change Request

MCR 1570
Page 1 of 1

TITLE: Fix Bugs in BOS

AUTHOR: Noel I. Morris

STATUS	DATE
Written	12/16/75
Status	A 12/23/75
Expires	06/23/76

-Coded in: [ ] PL/I [X] 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? [ ] yes [X] no
-Performance: [ ] Better [X] 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 Cmnd/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.**

Reasons:

1. The TEST, SAVE, and RESTOR commands contain a bug in the code which does single sector I/O after a disk error. This bug causes an extra iteration through the 64 word I/O loop.

2. The CONFIG command contains a bug which makes it impossible to invoke the command as part of a BOS runcom.

Proposals:

1. Change a TPL to a TPNZ.

2. Fix the code for saving and restoring runcom status in CONFIG.

Multics Change Request

MCR 1571
Page 1 of 1

TITLE: Improve Disk DIM performance

AUTHOR: Noel I. Morris

STATUS	DATE
Written	12/16/75
Status	A 12/23/75
Expires	06/23/76

-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? [ ] yes [X] no
-Performance: [X] 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)	

Objections/Comments:

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

Summary:
When a disk channel becomes free, a primitive in the disk DIM attempts to find more work for the channel to do. If there are disk requests queued, these queues are searched for an eligible request. A disk I/O request is eligible only if no other channel is performing I/O on the same disk drive as the one to which the request is directed. A method is proposed here which will enable the disk DIM to determine if any eligible requests are queued without actually searching the queues.

Proposal:
The disk DIM currently maintains a bit array indicating which disk drives are busy. It is proposed to add an additional bit array indicating the drives for which requests are queued. Before the queues are searched, a simple logical operation on the two bit arrays can be used to determine if the queues should be searched at all. In order to maintain this second bit array, a count of queued requests for each drive must also be maintained.

Implications:
The size of the disk DIM data base will have to increase by 32 words. However, the increase in code in the DIM itself will be negligible. In some cases, a search of the disk queues will be avoided and performance should improve.

Ver. 4
750508

Multics Change Request

MCR 1572
Page 1 of 1

TITLE: Modify Multics to run with a 4 Mega-Word System Controller

AUTHOR: Noel I. Morris

STATUS	DATE
Written	12/16/75
Status	A 12/23/75
Expires	06/23/76

-Coded in: [X] PL/I [ ] ALM [ ] other- explain in DETAILED PROPOSAL
-Planned for System MR 4.0
-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
X	Ring Zero
	Ring One
	SysDaemon/Admin.
	Runtime
	User Cmmd/Subr.
X	Initialization

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 4MW SCU differs from the 6000 System Controller in that some RSCR and SSCR formats are incompatible. It is proposed here to change those programs in system initialization and dynamic reconfiguration to allow use of the 4MW SCU with Multics. It is not proposed here to modify Multics to allow a configuration containing one 4MW SCU and two CPU's. Such a proposal may be made at a later time.

Proposal:
Modify scas_init and any programs in the dynamic reconfiguration package dependent on the RSCR and SSCR formats which change in the 4MW SCU.

Ver. 4
750508

Multics Change Request

MCR 1573
Page 1 of 1

TITLE: Modify BOS to run 4MW SCU

AUTHOR: Noel I. Morris

STATUS	DATE
Written	12/16/75
Status	A 12/23/75
Expires	06/23/76

-Coded in: ☐ PL/I ☒ ALM ☐ other- explain in DETAILED PROPOSAL
-Planned for System MR 4.0
-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
X	BOS
	Salvager
	Ring Zero
	Ring One
	SysDaemon/Admin.
	Runtime
	User Cmnd/Subr.

DOCUMENTATION CHANGES

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

Objections/Comments:

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

Summary:
It has been proposed to use the 4MW SCU with Multics. This system controller differs slightly from the 6000 controller. One basic difference is that the calendar clock cannot be loaded from the maintenance panel. It must be loaded via a program. Another difference is that some of the RSCR and SSCR data formats are not compatible.

Proposal:
Modify the BOS TIME command to allow loading the 4MW SCU clock. Modify any parts of BOS dependent on the SSCR or RSCR formats which have changed.

Ver. 4
750508

Multics Change Request

MCR 1574
Page 1 of 1

TITLE: Modify Multics for Dual IOM Operation

AUTHOR: Noel I. Morris

STATUS	DATE
Written	12/17/75
Status	A 12/23/75
Expires	06/23/76

-Coded in: [X] PL/I [X] ALM [ ] other- explain in DETAILED PROPOSAL
-Planned for System MR 4.0
-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
X	Ring Zero
	Ring One
	SysDaemon/Admin.
	Runtime
	User Cmnd/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 minor change is needed to the Multics IOM interface to allow the simultaneous operation of two IOM's. This change will allow future expansion to the 64 channel (NSA) IOM at a later time.

Proposal:
Modify the modules "iom_manager" and "iom_data_init" to be compatible with the IOM mailbox locations used by BOS.

Implications:
Additional wired-down memory is needed for the "iom_data" segment, but the increase in size is negligible.

er. 3
41022

Multics Change Request

MCR 1575
Page 1 of 1

TITLE: Emergency change to mc_wakeups_
AUTHOR: Paul Green

STATUS	DATE
Written	12/16/75
Status	A 12/23/75
Expires	06/23/76

-Coded in: ☑ PL/I ☐ AIM ☐ other- explain in DETAILED PROPOSAL
-Planned for System MR
-Fixes Bug Number(s) ERF 4554
-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
	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)	no change

Objections/Comments:

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

SUMMARY:

Change the any_other handler in mc_wakeups_ (one of the programs of the message coordinator) to be more fail-safe. Fix a bug noticed while reviewing the program.

REASONS:

MIT has crashed several times when the any_other handler got into a loop. This change to the handler should eliminate the loop. Fixing the bug noticed while analyzing the dumps may eliminate the problem which had caused the handler to be invoked.

IMPLICATIONS:

More reliable operation.

DETAILED PROPOSAL:

Submitted to MIT as an emergency change.

Ver. 3
741022 MULTICS CHANGE REQUEST

MCR 1576

TITLE: Eliminate "infqcnt"

AUTHOR: VanVleck

STATUS | DATE
Written | 12/10/75
Status | A 12/23/75
Expires | 06/10/76

Planned for System: 4.0
Fixes Bug Number(s): not applicable
Documented in MTB: not applicable
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
( )Sys. Anal. Tools
( )Sys. Prog. Tools
( )355
( )BOS
( )Salvager
(X)Ring Zero
( )Ring One
( )SysDaemon/Admin
( )Runtime
( )User Command/Subr

DOCUMENTATION CHANGES (specify one or more)

MPM (vol,sect) ag93 MPAM (sect)
MOSN (sect) MSAM (sect)
PLMs (AN#)
Info Segs
Other

OBJECTIONS/COMMENTS:

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

SUMMARY:

Eliminate the item "infqcnt " the inferior quota count, from the quota information maintained by the system.

Modify the tools sweep, print_disk, and charge_disk not to expect this item. Modify getquota not to print it when the "-all" control argument is given.

REASONS:

This item is difficult to maintain under the new storage system. It is not particularly useful. Its elimination simplifies the hardcore slightly.

IMPLICATIONS:

This is an incompatible change. If any user program does anything with infqcnt, the program must be changed. It is very unlikely that any such dependency exists.

ver. 4
750508

Multics Change Request

MCR 1577
Page 1 of 1

TITLE: Keeping Bad Pathnames out of the Pathname Associative Memory

AUTHOR: R. Bratt

STATUS	DATE
Written	11 December 75
Status	A 12/23/75
Expires	06/23/76

-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? [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
X	Ring Zero
	Ring One
	SysDaemon/Admin.
	Runtime
	User Cmnd/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)	PLM not yet written

Objections/Comments:

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

REASONS:

For historical reasons Multics hardcore accepts pathnames of the form >>x for >x. As a result, it is possible for find_ to place the legal, but offensive pathname ">>sll" in the pathname associative memory. This in turn allows "pwd" to return ">>sll". This is unacceptable.

PROPOSAL:

Prevent find_ from entering pathnames starting with ">>" in the pathname associative memory.

IMPLICATIONS:

Since the address space manager will not generate pathnames starting with ">>", pwd will cease its objectionable behavior.

ver. 2
7309191

MULTICS CHANGE REQUEST

MCR 1578

TITLE: Enhancement of the on line dump peruser, ol_dump

AUTHOR: James A. Bush MSS - Phoenix

JWG
AGB

STATUS	DATE
Written	11/03/75
Approved	12/12/75
Rejected	
Postponed	
Withdrawn	
Expires	06/23/76

SOURCE: (if external; e.g., "User", "Marketing")

CLASSIFICATION	JUSTIFICATION
Incompatible Change	Marketing Requirement
X Extension	Conformance to Standard
Restriction	X Increased Consistency
X Performance Improvement	Simplification
Reliability Improvement	Generalization
	(Unreported) Bug Fix

Replaced by proposal MCR

Implemented in System

Objections/Comments:

Documented in system Dump Analysis PLM AN53

Use these headings: REASONS, SUMMARY, IMPLICATIONS, and optionally DETAILED PROPOSAL

REASONS: New functions have been added to "ol_dump", to aid dump readers in analyzing system "crash" dumps on line. This updated version of "ol_dump", with the temporary name "dmp_anz", has been installed in ">auth_maint" on system "M" and has been in general use by dump readers in Phoenix for several months. In addition "dmp_anz" has been kept up to date with the latest system changes.

SUMMARY: Dccumentation for "ol_dump" and its assoiated external subroutines ("hran_", form_inst_", "prtscu_", and "namef_") are attached to this MCR.

IMPLICATIONS:
1. "ol_dump" will have more utility.
2. A globol name change from "dmp_anz" to "ol_dump" would allow replacement of the existing version of "ol_dump" in ">tools".

ol_dump

Name: ol_dump

The ol_dump command can be used to look at selected parts of an online dump created by the BOS FDUMP command and copied into the Multics hierarchy by the copy_fdump command. The command is designed to aid system programmers in the task of crash analysis. The command assumes all dumps of interest are found in the directory ">dumps".

Usage

ol_dump -erfno-

where erfno is an optional error report form number given in decimal. If erfno is not specified, ol_dump enters its request loop described below. If an erfno is given, ol_dump searches the directory >dumps for a copy of the dump and if it finds the dump, it initializes itself to be able to process the given dump. If the dump is not found, the user is told and the request loop is entered.

Request_Loop

Once ol_dump has processed the erfno argument it enters a loop reading requests from user_input. The requests allow the user to look at selected regions of the dump currently under analysis or to choose another dump (erfno) for analysis. The following requests are implemented (letters in parentheses are abbreviations):

Request	Function
erf nn	Selects another dump (the one whose erfno is nn) for immediate analysis.
quit (q)	Returns.
command (c)	Passes the rest of the request line onto the current command processor.
list (l)	Lists the dumps in >dumps by showing the name of the first component of the dump. The names of dumps tell when the dump was taken and what the erfno is.

ol_dump

help (?)

Lists the requests of ol_dump.

dump (d) args (mode)

Displays selected words located in the current dump under analisis. There are three mandatory arguments and one optional "mode" argument. The first argument can be one of the following:

segn

Display selected words from segment "n" in the current process. "n" may be a segment number or name.

mem

Display selected words starting at absoulute memory location "n". where "n" is the second argument and represents an absolute memory location. A search is made of all running process's descriptor segments and AST/PT entrys. If the requested address is found, the segment number and name, segment offset, and the process DBR value is ouput as well as the requested number of words. If the requested memory address is not found, it is assumed to be in free store.

The second argument is the segment offset or the absolute memory address (if argument one is "mem").

The third argment is the number of words to be displayec.

The optional "mode" argument is any of the output modes used with the debug command ("o" for octal, "a" for ASCII, "p" for pointer, "i" for instruction format etc.). If the instruction mode ("i") is used, and if the requested segment is not found in the dump (only segments with read and write

ol_dump

access are found in the dump. This usually precludes executable object segments from being dumped) then a search of the library directories are made and if found, the segment is dumped in instruction format.

dbr Arg — Switches to another process (in the same dump). Arguments are as follows:

- cpun — Switches to process which is executing on cpu (n). Where (n) is the cpu #.
- value — Switches to another process by specifying the dbr value for the new process.

name (n) segno (offset) — Displays the SLT or SST name for the given segment number. If the segment in question is a bound segment, and the optional argument of "offset" is given, then the bound segment name as well as the component name and the relative offset in that component are displayed.

amsdw (ams) — Displays the contents of the SDW Assiociative Memory in the Bootload CPU, at the time of the dump.

amptw (amp) — Displays the contents of the PTW Assiociative Memory in the Bootload CPU, at the time of the dump.

syserdta (sdta) — Displays the message entries in the wired message segment "syserr_data".

syserlog (slog) no — Displays the specified number of message entries in the paged message segment "syserr_log" starting with most recent entry.

ol_dump

proc (p) Arg — Displays some APT data for the process specified. Arguments are as follows:

- all — All of the APTE'S are displayed.
- cur — Only the APTE for the current process (as defined by the dbr value) will be displayed.
- run — Displays only those APTE'S which are currently executing on the configured CPU'S.
- rdy — Only those APTE'S whose execution state is "ready" will be displayed.
- wat — Only those APTE'S whose execution state is "waiting" will be displayed.
- blk — Only those APTE'S whose execution state is "blocked" will be displayed.
- stp — Only those APTE'S whose execution state is "stopped" will be displayed.
- emp — Only those APTE'S whose execution state is "empty" will be displayed.
- no. — One specfic APTE is displayed.

stack (s) seg (os) (lg) — Displays a stack trace of the stack segment specifiec by the "seg" argument which may be a segment name or number. If the optional offset argument (os) is given, then the trace will start at the frame specified by "os" and continue to the end of the stack. If the offset argument is omitted, then the trace is started at the stack base. The segment name of the return pointer is displayed for

ol_dump

all segments. If the name is a bound segment, the component name as well as the relative offset is displayed in the form "bound_seg$comp_name|offset". If the return pointer indicates "pl1_operators", then Pointer Register 0 is picked up and used instead. This is indicated by the flag "[pr0]" being displayed after the segment name. If the optional "lg" argument is given, an octal dump of each stack frame, including relative offsets, is produced.

mcprds (mcpr) arg (lg) Displays the PRDS machine conditions for the specified argument. Only an interpreted version of the SCU data is displayed unless the "lg" argument is used. The arguments are as follows:

int Displays machine conditions for prds|interrupt data.

systroub Displays machine conditions for prds|system trouble data.

systrsav Displays machine conditions for prds|system trouble save data.

fim Displays machine conditions for prds|fim data.

all Displays all machine condition save areas in PRDS.

lg Displays Pointer Registers and Processor Registers as well as SCU Data.

mcpds (mcp) arg (lg) Displays the PDS machine conditions for the specified argument. Only an interpreted version of the SCU data is displayed unless the "lg" argument is used. The

ol_dump

arguments are as follows:

pgflt	Displays machine conditions for pds|page fault data.
int	Displays machine conditions for pds|interrupt data.
fim	Displays machine conditions for pds|fim data.
sig	Displays machine conditions for pds|signal data.
all	Displays all machine condition save areas in PDS.
lg	Displays Pointer Registers and Processor Registers as well as SCU Data.

dumpregs (dregs) (Arg) — Displays the processor registers which were saved at the time of the dump, from the bootload CPU. If no arguments are given, all of the registers are displayed. The optional arguments are as follows:

ptr	Displays the pointer registers only.
preg	Displays the processor registers only.
scu	Displays the saved SCU data only.
all	Displays all of the above.

lrn (arg1) (arg2) — Displays a breakout of the descriptor segment (dseg) by printing the SDW's, segment numbers, and names for specified segment numbers of dseg. If no optional arguments are given, the descriptor segment is broken out from segment number 0 to the last segment in dseg. If optional

ol_dump

argument "arg1" only is given, the descriptor segment breakout starts at segment number "arg1" and continues to the end of dseg. If optional arguments "arg1" and "arg2" are both given, the descriptor breakout starts at segment number "arg1" and continues to segment number "arg2".

segno (segn) name — Displays the segment number for a given entry name.

ssd arg1 (arg2) (arg3) — Allows the user to specify up to 3 directories for finding offsets & bindmaps for hardcore segments. The default directory is ">ldd>hardcore>object". Values for mandatory "arg1" are as follows:

- pr — Displays the current directories searched.
- def — Resets the directories searched to the default value.
- path — Pathname of directory to search (first).

Optional arguments "arg2" & "arg3" are used to specify pathnames of directories to search.

hisregs (hregs) arg — Displays a composite analysis of the processor History Registers. Arguments are as follows:

- pds — Displays the stored History Registers from the PDS.
- dmp — Displays the History Registers stored at the time of the dump by the Bootload Processor.
- help — Displays a list of the abreviations used in the

History Register analysis, and their meaning.

pcd	(arg)	Displays the contents of the "config_deck" segment in an interrupted fashion. Arguments can be any one of the card types found in the configuration deck (cpu, mem, prph, etc.). The pcd command will process from 1 to 32 arguments. If no arguments are given, the entire config deck is displayed.
ast (pt) name		Displays the AST entry and page table for the given segment. Name may be an segment name or number.
queue (tcq)		Displays the scheduler's priority queue in order of priority.

If the request line is none of the above, the **entire** line is passed directly to the current command processor.

Name: hran_

The hran_ subroutine provides a means of displaying a composite analysis of Multics processor history register cata.

Entry: hran_$hranl

This entry, given a pointer to tne base of the nistory register data structure, will display a composite analysis of the history register data.

Usage

```
dcl hran_$hranl entry (ptr);

call hran_$hranl (hregptr);
```

where:

1. hregptr is a pointer to the history register data which should be in the following format:

```
dcl 1 hregdata based (hregptr),
    2 ouhist (0:15) bit (72),
    2 cuhist (0:15) bit (72),
    2 duhist (0:15) bit (72),
    2 auhist (0:15) bit (72);
```

Entry: hran_$hrlgnd

This entry point, called with no arguments, will display a legend giving the definitions of all flags and symbols used in history register analysis output.

Usage

```
dcl hran_$hrlgnd entry;

call hran_$hrlgnd;
```

form_inst_

Name: form_inst_

The form_inst_ subroutine provides a means of displaying 36 bit words in Multics processor instruction word format.

Usage

```
dcl form_inst_ entry (ptr, fixed bin, fixed bin, bit (1));

call form_inst_ (wdptr, reloff, nwds, pdesc);
```

where:

1. wdptr — is a pointer to the first word to be displayed.
2. reloff — is the relative offset of the first word to be displayed.
3. nwds — is the number of words to be displayed.
4. pdesc — if pdesc = "1"b, then EIS descriptor words will be interpreted, following an EIS multiword instruction. If pdesc = "0"b, then the words following an EIS multiword instruction are treated like they were valid instruction words. This feature is useful when displaying the instruction word pair contained in SCU data, where the word following an EIS multiword instruction is not necessaraly an EIS descriptor.

prtscu_

Name: prtscu_

The prtscu_ subroutine, given a pointer to data stored by an SCU (store control unit) instruction, and other control information, will display an English description of the contents of the SCU data.

Usage

```
dcl prtscu_ entry (ptr, fixed bin, bit (1), ptr, ptr, ptr,
                   ptr, ptr, ptr, fixed bin);

call prtscu_ (segptr, offset, lgsw, sltp, sltnp, sstp,
              sstnp, dsegp, libp, numdir);
```

where:

1. segptr — is a pointer to the SCU data to be displayed.
2. offset — is the relative offset of the SCU data.
3. lgsw — if lgsw = to "0"b, then only an English interpretation of the SCU data will be displayed. If lgsw = to "1"b, then the SCU data will be displayed in octal as well as in an interpreted fashion.
4. sltp — is a pointer to a copy of the slt segment.
5. sltnp — is a pointer to a copy of the slt_names_ segment.
6. sstp — is a pointer to a copy of the sst segment.
7. sstnp — is a pointer to a copy of the sst_names_ segment.
8. dsegp — is a pointer to a copy of the descriptor segment.
9. libp — is a pointer to an array of library names which will be searched for bindmap information for hardcore segments.
10. numdir — is the number of directorys in the directory array to be searched.

namef_

Name: namef_

The namef_ subroutine provides a means of obtaining the full pathname for an slt or non-hardcore segment given certain control information.

Usage

```
dcl namef_ entry (ptr, ptr, ptr, ptr, ptr, ptr, ptr,
                  fixed bin) returns (char (*));

ascii_str = namef_ (segptr, sltp, sltnp, sstp, sstnp,
                    dsegp, libp, numdir);
```

where:

1. segptr — is a pointer to the segment whose path name is desired. If the segment is bound then the component name and relative offset are also returned.
2. sltp — is a pointer to a copy of the slt segment.
3. sltnp — is a pointer to a copy of the slt_names_ segment.
4. sstp — is a pointer to a copy of the sst segment.
5. sstnp — is a pointer to a copy of the sst_names_ segment.
6. dsegp — is a pointer to a copy of the descriptor segment.
7. libp — is a pointer to an array of library names which will be searched to find bind map information for hardcore segments.
8. numdir — is the number of directorys in the directory array to be searched.
9. ascii_str — is the resultant char (*) pathname. (output)

Entry: namef_$no_comp

This entry point performs the same function as namef_, except that no component name and relative offset breakout are performed on bound segments.

Usage

```
dcl namef_$no_comp entry (ptr, ptr, ptr, ptr, ptr, ptr)
                          returns (char (*));

ascii_str = namef_$no_comp (segptr, sltp, sltnp, sstp,
                            sstnp, dsegp);
```

where: The arguments are as described above.

ver. 4
750508

Multics Change Request

MCR 1579
Page 1 of 2

TITLE: Plans for the PL/I Compiler

AUTHOR: R. Barnes

STATUS	DATE
Written	16 December 75
Status	A 12/23/75
Expires	06/23/76

-Coded in: [X] PL/I [X] ALM [ ] other- explain in DETAILED PROPOSAL
-Planned for System MR 4.0 (+)
-Fixes Bug Number(s)
-Documented in MTB
-User/Operations-visible Interface change? [ ] yes [X] no
-Incompatible change? [ ] yes [X] no
-Performance: [X] 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 Cmnd/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)	Just a planning MCR

Objections/Comments:

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

SUMMARY:

Various improvements are planned for the PL/I compiler.

DETAILED PROPOSAL:

Release 4.0

- Make several optimizations including:
  - Improve handling of logical expressions in if statements and while clauses;
  - Improve handling of indicators so that code for if a<b then ...; else if a>b ... is improved;
  - Improve aq and pointer register management in expression evaluation;
  - Improve code for halfword offset and length references;
  - Simplify offset expressions of the form (a+b) -a;
  - Possibly improve some simple cases of picture unpacking.

- Convert the compiler to generate code for and use the new area package.

- Use *system links for external data references without $'s, provided that the detailed MCR is approved.

- Implement any changes needed for quick stream output.

After Release 4.0

- Implement a loop optimizer

- Bring the compiler closer to the ANSI standard by improving the string builtin and implementing the some and every builtins.

- Implement any additional changes necessary for quick stream I/O.

Ver. 3
741022 MULTICS CHANGE REQUEST

MCR 1580

TITLE: Install FAST run unit manager

AUTHOR: M. Weaver

STATUS | DATE
Written | 12/15/75
Status | 12/23/75
Expires | 06/15/76

Planned for System: MR 3.1
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 ( )same ( )worse

DOCUMENTATION CHANGES (specify one or more)
MPM (vol,sect) MPAM (sect)
MOSN (sect) MSAM (sect)
PLMs (AN#)
Info Segs
None (reason) user interface covered in language and FAST manuals

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

OBJECTIONS/COMMENTS:

Change names of interfaces to include subsystem name.

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

SUMMARY:

Install the run unit manager for the DTSS-compatible version of FAST.

REASONS: Needed to run user programs in FAST.

DETAILED PROPOSAL:

The run unit manager implements the FAST run command. In FAST, all user programs are executed within a run unit so that each program's name space and storage can be isolated. fortran, basic and double precision basic programs may not be mixed within a run unit but pl1 and alm programs may be called if they don't make any external references themselves. A single temporary scratch segment is used to store name lists, linkage sections, etc. and for language runtime use; this is truncated at the end of the run unit.

The run unit manager's steps are as follows:

1) Find all programs referenced via library statements. (Library statements are lists of pathnames of segments containing subprograms referenced by the program.)

2) Store pointers to all entry points in the programs found by step 1.

3) Compile all source programs; the result is a single object segment compiled from all the source libraries plus the main program, if it is

source. The compiler and the run unit manager interact to perform this step and step 1, the one supplying names of "libraries" and the other supplying pointers to the segments.

4) Allocate all common blocks and snap all links. This step applies only to fortran programs.

5) Execute programs.

6) Recover all storage, terminating segments.

The main entry point is called as follows:

```
declare run_unit_manager_  entry (char(8) aligned, ptr,
          fixed bin(24), char(168) aligned, bit(1) aligned,
          ptr, fixed bin(35));

call run_unit_manager_ (system, program_ptr, program_lng,
          main_path, debug_sw, code);
```

1) system — is the name of the current system. Input

2) program_ptr — points to the main program, which can be either source or object. Input

3) program_lng — is the bit count of the main program. Input

4) main_path — is the pathname of the main program. Input

5) debug_sw — "1"b->program is to be run in debug mode. Input

6) code — is a system status code. Output

The run unit manager interacts with compilers by way of the following 2 internal procedures.

```
declare   get_next_source_seg_ entry (ptr);

call get_next_source_seg_ (source_info_ptr);
```

1) source_info_ptr — points to the following structure. Output

```
declare   1 source_info aligned,
          2 input_pointer ptr,
          2 input_lng fixed bin(21),
          2 dirname char(168) varying,
          2 segname char(32) varying,
          2 date_time_modified fixed bin(71),
          2 unique_id bit(36);
```

1) input_pointer points to the source program.

2) input_lng is the length of the source in characters.

3) dirname is the name of the source segment's directory.

4) segname is the segment name of the source segment.

5) date_time_modified is the date-time modified of the source segment.

6) unique_id is the unique id of the source segment.

```
declare add_to_lib_list_ entry (char(*));

call add_to_lib_list_ (pathname);
```

1) pathname is the pathname of a library segment.

Ver. 3
741022

Multics Change Request

MCR 1581
Page 1 of 1

TITLE: New gate entry: hcs_$star_dir_list_

AUTHOR: T. Casey

STATUS	DATE
Written	12/18/75
Status	A 12/23/75
Expires	06/23/76

-Coded in: [X] PL/I [ ] ALM [X] other- explain in DETAILED PROPOSAL
-Planned for System MR 3.1
-Fixes Bug Number(s)
-Documented in MTB unpublished
-User/Operations-visible Interface change? [X] yes [ ] no
-Incompatible change? [ ] yes [X] no
-Performance: [ ] Better [ ] Same [X] 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 (see comment)
PLMS (AN #)	
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	
Other (Name)	
None (Reason)	

Objections/Comments: Document in 4.0 update to SWG unless there is a new replacement interface by 4.0

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

SUMMARY:

Install a new gate entry: hcs_$star_dir_list_ and a new version of star_ implementing the new entry. This entry has the same argument list as hcs_$star_list_, but it does not return any items that will be stored in the VTOC under NSS.

REASONS:

It is necessary to install list and check_info_segs commands that do not, by default, access the VTOCE's of all entries in a directory.

IMPLICATIONS:

When the new gate entry is installed, callers of the old entry will continue to work as they did before, and it will then be possible to test and then install the NSS versions of list and cis, before NSS is installed.

DETAILED PROPOSAL:

The new entry returns bitcount instead of records used, date_time_entry_modified instead of date_time_modified, and zero instead of date_time_used.

SWG writeup of hcs-$star. part of MCR 1581.

Entry: hcs_$star_list_

This entry point returns more information about the selected entries, such as the mode and records used for segments and directories and link pathnames for links.

Usage

```
declare hcs_$star_list_ entry (char(*), char(*), fixed bin(3), ptr,
     fixed bin, fixed bin, ptr, ptr, fixed bin(35));

call hcs_$star_list_ (dir_name, star_name, select_sw, area_ptr,
     branch_count, link_count, entry_ptr, n_ptr, code);
```

where:

1. **dir_name** is the pathname of the containing directory. (Input)

2. **star_name** is the entryname that can contain asterisks or question marks. (Input)

3. **select_sw** indicates what information is to be returned. (Input) It can be:
   - 1 information is returned about link entries only
   - 2 information is returned about segment and directory entries only
   - 3 information is returned about segment, directory, and link entries
   - 5 information is returned about link entries only, including the pathname associated with each link entry
   - 7 information is returned about segment, directory, and link entries, including the pathname associated with each link entry

4. **area_ptr** is a pointer to the area in which information is to be returned. If the pointer is null, branch_count and link_count are set to the total number of selected entries. See "Notes" below. (Input)

5. **branch_count** is a count of the number of segments and directories that match the entryname. (Output)

6. **link_count** is a count of the number of links that match the entryname. (Output)

7. **entry_ptr** is a pointer to the allocated structure in which information on each entry is returned. (Output)

8. n_ptr — is a pointer to the allocated array in which selected entrynames and pathnames associated with link entries are stored. (Output)

9. code — is a storage system status code. See "Status Codes" below. (Output)

Notes

Even if area_ptr is null, branch_count and link_count may be set. If information on segments and directories is requested, branch_count is set to the total number of segments and directories that match star_name. If information on links is requested, link_count is the total number of links that match star_name.

If area_ptr is not null, an array of entry information structures and the names array, as described in the hcs_$star_ entry point above, are allocated in the user-supplied area. The number of structures allocated is count, which is equal to branch_count plus link_count. Each element in the structure array may be either of the structures described below (the links structure for links or the branches structure for segments and directories). The correct structure is indicated by the type item, the first item in both structures.

The first three items in each structure are identical to the ones in the structure returned by the hcs_$star_ entry point.

The following structure is used if the entry is a segment or a directory:

```
dcl 1 branches (count)          aligned based (entry_ptr),
    (2 type                     bit(2),
     2 nnames                   fixed bin(15),
     2 nindex                   fixed bin(17),
     2 dtm                      bit(36),
     2 dtu                      bit(36),
     2 mode                     bit(5),
     2 pad                      bit(13),
     2 records                  fixed bin(17)) unaligned;
     2 raw_mode                 bit (5)
     2 master_dir               bit (1)
     2 records                  fixed bin (24) unaligned;
```

where:

1. type — specifies the storage system type of entry:
   0 ("00"b) link
   1 ("01"b) segment
   2 ("10"b) directory

2. nnames — specifies the number of names for this entry that match star_name.

3. nindex — specifies the offset in the array of names (pointed to by n_ptr) for the first name returned for this entry.

4. dtm — is the date and time the segment or directory was last modified.

5. dtu — is the date and time the segment or directory was last used.

6. mode — is the current user's access mode to the segment or directory. See "Access Modes" below.

7. raw_mode — is the current user's access mode before ring brackets and access isolation are considered.

~~7. pad — is unused space in this structure.~~

8. master_dir — is "1"b if the entry is a master directory.

9. records — is the number of 1024-word records of secondary storage that have been assigned to the segment or directory.

The following structure is used if the entry is a link:

```
dcl 1 links (count)              aligned based (entry_ptr),
    (2 type                      bit(2),
     2 nnames                    fixed bin(15),
     2 nindex                    fixed bin(17),
     2 dtm                       bit(36),
     2 dtd                       bit(36),
     2 pathname_len              fixed bin(17),
     2 pathname_index            fixed bin(17)) unaligned;
```

where:

1. type — is the same as above.

2. nnames — is the same as above.

3. nindex — is the same as above.

4. dtm — is the date and time the link was last modified.

5. dtd — is the date and time the link was last dumped.

6. pathname_len — is the number of significant characters in the pathname associated with the link.

7. pathname_index — is the index in the array of names for the link pathname.

If the pathname associated with each link was requested, the pathname is placed in the names array and occupies six units of this array. The index of the first unit is specified by pathname_index in the links array. The length of the pathname is given by pathname_len in the links array.

Entry: hcs_$star_dir_list_

This entry point returns ~~more~~ information about the selected entries, ~~such as the mode and records used for segments and directories and link pathnames for links.~~ similar to that returned by hcs_$star_list. However, it returns only information that is stored in the directory, thus avoiding the cost of accessing the VTOC entries for the selected directory entries

Usage

```
declare hcs_$star_dir_list_ entry (char(*), char(*), fixed bin(3), ptr,
     fixed bin, fixed bin, ptr, ptr, fixed bin(35));

call hcs_$star_dir_list_ (dir_name, star_name, select_sw, area_ptr,
     branch_count, link_count, entry_ptr, n_ptr, code);
```

The arguments are exactly the same as those for hcs_$star_list_.

where:

1.	dir_name	is the pathname of the containing directory. (Input)
2.	star_name	is the entryname that can contain asterisks or question marks. (Input)
3.	select_sw	indicates what information is to be returned. (Input) It can be: 1 information is returned about link entries only 2 information is returned about segment and directory entries only 3 information is returned about segment, directory, and link entries 5 information is returned about link entries only, including the pathname associated with each link entry 7 information is returned about segment, directory, and link entries, including the pathname associated with each link entry
4.	area_ptr	is a pointer to the area in which information is to be returned. If the pointer is null, branch_count and link_count are set to the total number of selected entries. See "Notes" below. (Input)
5.	branch_count	is a count of the number of segments and directories that match the entryname. (Output)
6.	link_count	is a count of the number of links that match the entryname. (Output)
7.	entry_ptr	is a pointer to the allocated structure in which information on each entry is returned. (Output)

8. n_ptr — is a pointer to the allocated array in which selected entrynames and pathnames associated with link entries are stored. (Output)

9. code — is a storage system status code. See "Status Codes" below. (Output)

Notes — The notes for hcs_$star_list_ also apply to this entry.

Even if area_ptr is null, branch_count and link_count may be set. If information on segments and directories is requested, branch_count is set to the total number of segments and directories that match star_name. If information on links is requested, link_count is the total number of links that match star_name.

If area_ptr is not null, an array of entry information structures and the names array, as described in the hcs_$star_ entry point above, are allocated in the user-supplied area. The number of structures allocated is count, which is equal to branch_count plus link_count. Each element in the structure array may be either of the structures described below (the links structure for links or the branches structure for segments and directories). The correct structure is indicated by the type item, the first item in both structures.

~~The first three items in each structure are identical to the ones in the structure returned by the hcs_$star_ entry point.~~ The layouts of these structures are identical to those used by hcs_$star_list_; only the meanings of two elements differ: dtem and bitcount.

The following structure is used if the entry is a segment or a directory:

```
dcl 1 branches (count)          aligned based (entry_ptr),
    (2 type                     bit(2),
     2 nnames                   fixed bin(15),
     2 nindex                   fixed bin(17),
     2 dtem                     bit(36),
     2 dtu  pad1                bit(36),
     2 mode                     bit(5),
     2 pad                      bit(13),
     2 records                  fixed bin(17)) unaligned;
     2 raw_mode                 bit (5)
     2 master_dir               bit (1)
     2 bitcount                 fixed bin (24)) unaligned;
```

where:

1. type — specifies the storage system type of entry:
   0 ("00"b) link
   1 ("01"b) segment
   2 ("10"b) directory

2. nnames — specifies the number of names for this entry that match star_name.

3. nindex — specifies the offset in the array of names (pointed to by n_ptr) for the first name returned for this entry.

4. dtem — is the date and time the directory entry for the segment or directory was last modified.

5. pad1 is unused space in this structure

~~5. dbu is the date and time the segment or directory was last used.~~

6. mode is the current user's access mode to the segment or directory. See "Access Modes" below.

7. raw_mode is the current user's access mode before ring brackets and access isolation are considered.

~~7. pad is unused space in this structure.~~

8. master_dir is "1"b if the entry is a master directory.

~~8. records is the number of 1024-word records of secondary storage that have been assigned to the segment or directory.~~

9. bitcount is the bitcount of the segment or directory.

~~The following structure is used if the entry is a link:~~

```
dcl 1 links (count)              aligned based (entry_ptr),
    (2 type                      bit(2),
     2 nnames                    fixed bin(15),
     2 nindex                    fixed bin(17),
     2 dtm                       bit(36),
     2 dtd                       bit(36),
     2 pathname_len              fixed bin(17),
     2 pathname_index            fixed bin(17)) unaligned;
```

where:

1. type — is the same as above.
2. nnames — is the same as above.
3. nindex — is the same as above.
4. dtm — is the date and time the link was last modified.
5. dtd — is the date and time the link was last dumped.
6. pathname_len — is the number of significant characters in the pathname associated with the link.
7. pathname_index — is the index in the array of names for the link pathname.

If the pathname associated with each link was requested, the pathname is placed in the names array and occupies six units of this array. The index of the first unit is specified by pathname_index in the links array. The length of the pathname is given by pathname_len in the links array.

The structure used if the entry is a link is identical to the one used by hcs_$star_list_, and identical information is returned by both entries for links. (A link does not have a VTOC entry.)

Ver. 3 741022	MULTICS CHANGE REQUEST	MCR 1582

TITLE: New version of storage system.

AUTHOR: VanVleck

STATUS	DATE
Written	12/19/75
Status	A 12/23/75
Expires	06/19/76

Planned for System: MR 4.0
Fixes Bug Number(s): 240, 386
Documented in MTB: see below
Incompatible Change: yes
User/Operations-visible Interface Change: yes
Coded in: (X)PL/I (X)ALM (X)other-see below
Performance: ( )better ( )same ( )worse

DOCUMENTATION CHANGES (specify one or more)

MPM (vol,sect) MPAM (sect)
MOSN (sect) MSAM (sect)
PLMs (AN#) AN61
Info Segs pending_changes
Other

CATEGORY (check one)

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

OBJECTIONS/COMMENTS:

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

SUMMARY:

Install a new version of the storage system with the following features:

1. Split branch into branch and VTOCE.
2. Reinterpret disk address; add PVT; change FSDCT.
3. Withdraw disk addresses at fault time; use paged FSDCT.
4. Improve directory locking.
5. Simplify AST locking.
6. Label each volume; reinterpret partition.
7. Improve system bootload sequence.
8. Integrate salvager into MST.
9. Support MSU-451 disk drives.
10. Improve disk error recovery.
11. Change rules for branch appending.

REASONS:

Improve system reliability and capacity as described in MTB-110. Provide for future implementation of removable hierarchy volumes. Simplify and clean up storage system implementation.

IMPLICATIONS:

The following consequences are described in MTB-239:

1. Need for more disk storage.
2. Improved reliability.
3. Performance changes.
4. Operator interface changes.
5. Administrative changes.

Operator documentation is currently being prepared.

DOCUMENTATION:

MTB-017, November 1973.
MTB-055, April 1974
MTB-060, May 1974
MTB-065, April 1974
MTB-095, June 1974
MTB-110, August 1974
MTB-167, February 1975
MTB-203, June 1975
MTB-206, June 1975
MTB-213, July 1975
MTB-220, September 1975
MTB-221, September 1975
MTB-229, October 1975
MTB-233, November 1975
MTB-237, November 1975
MTB-238, November 1975
MTB-239, November 1975
MTB-243, December 1975

Ver. 4
750508

Multics Change Request

MCR 1583
Page 1 of 3

TITLE: Additional FAST Interface: MTSS--Multics Time Sharing Subsystem

AUTHOR: C. T. Clingen

STATUS	DATE
Written	19 December 75
Status	A 12/23/75
Expires	06/23/76

DOCUMENTATION CHANGES

-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.
	Runtime
X	User Cmnd/Subr.

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

Objections/Comments:
Alternate name proposal:
DTSS becomes DFAST
MTSS becomes FAST

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

SUMMARY:

Provide a Multics-oriented user interface, MTSS--Multics Time Sharing Subsystem, to the FAST environment. Multics commands, concepts, names, and conventions will be used. The initial version of this facility should be planned for release with the DTSS-oriented user interface. The end result will be two user interfaces to FAST--DTSS and MTSS. This MCR proposes a new project; approval does not authorize installation of the resulting software.

REASONS FOR PROPOSAL:

Objectives of the project are:

1. Alternate to DTSS interface: This interface to Basic and Fast Fortran will provide users with an alternate to DTSS-oriented conventions. MTSS will observe Multics conventions regarding command names, terminology, pathnames, access control, etc. It should be almost effortless for a user to switch between the full Multics and this new limited environment. The goal is for the initial version of this subsystem to be released concurrently with the DTSS in MR3.1.

2. Ease of Use: The user interface should be simple to learn, remember, and use for both the novice user and the occasional user; it should not be assumed that the user is a professional programmer or a frequent computer user.

3. Performance: The implementation must emphasize efficiency and high performance.

4. Flexibility: The implementation should be sufficiently general to serve as a framework for implementing other limited user environments.

IMPLICATIONS:

Implications of the project objectives include:

1. Alternate to DTSS Interface:

   a. Multics concepts such as dynamic linking, (restricted), library searching, segment naming conventions, explicit compilations, input canonicalization, and input erase and kill processing will be observed to the extent that they do not conflict with objective 2--ease of use.

   b. The requirement for relatively early release may significantly limit the capabilities/flexibility of the first release. However, end-user documentation must be available with the initial release.

2. Ease of Use:

   a. MTSS will be a restricted system (limited service system). The user interface will be documented in a small manual similar to the DTSS User's Guide.

   b. The simplest user interfaces to this type of facility are usually constructed around a line-numbered editor/command processor interface supporting some notion of current file; this interface allows the writing, correction, and running of programs with an absolute minimum of conceptual baggage. Such an interface may be perceived as conflicting with the objectives of a Multics-like interface. Any such conflicts must be resolved as the first step in the project.

3. Performance: The performance improvement implementation techniques already proposed for FAST, such as prelinking, will continue to be used.

4. Flexibility: Relevant internal interfaces of the implementation will be documented in the Subsystem Writers' Guide after sufficient exposure and experience have been gained.

DETAILED PROPOSAL:

The complete user interface will be documented in one or more MTB's; the normal design review cycle will be followed--although the schedule may be accelerated to accomodate release requirements. One or more MCR's will be submitted for approval of the actual design.