MULTICS TECHNICAL BULLETIN MTB-263

TO: Distribution

FROM: Joan Scott

DATE: 27 February 76

SUBJECT: Multics Change Requests

Enclosed are copies of Multics Change Requests which were approved from 1 February 76 through 15 February 76.

Ver. 4
750508

Multics Change Request

MCR 1609
Page 1 of 1

TITLE: More vfile_ bug fixes for 3.1

AUTHOR: M. Asherman

STATUS	DATE
Written	01/27/76
Status	A 02/03/76
Expires	08/03/76

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

Category (Check One)

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

DOCUMENTATION CHANGES

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

Objections/Comments:

emergency fix--documentation change attached.

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

SUMMARY:

Fix the following new vfile_ bugs:

1) incorrect specification for -no_trunc attach option--should cause initial position to be at beginning of file in openings for stream_input_output.

2) erroneous recovery from interrupted write operations on indexed files when overflow occurs (in some cases).

3) failure to reflect use of -no_trunc option in attach description.

IMPLICATIONS:

#2 could result in damage to the file, but likelihood of occurrence of this error is vanishingly small.

may wait indefinitely. If no wtime is given, a default value of 1 is used.

-blocked -n- specifies attachment to a blocked file. If a nonempty file exists, n is ignored and may be omitted. Otherwise, n is used to set the maximum record size (bytes).

-no_trunc indicates that a put_chars operation into the middle of an unstructured file (stream_input_output) is permitted, and no truncation is to occur in such cases. Also prevents

MCR 1609
MPM for vfile_

14.

the truncation of an existing file at open. (This control argument is provided to support Fortran "direct" files; in conjunction with the -header control argument, it supports FORTRAN/BASIC random numeric files.)

-append in input_output openings, this causes put_chars and write_record operations to add to end_of_file instead of truncating when the file position is not at end_of_file. Also the position is initially set to beginning_of_file, and an existing file is not truncated at open.

+ (in stream_input_output openings) causes the next byte position to be initially set to beginning of file.

Missing paper on MCR approved this a.m. 2/3/76

Ver. 4
750508

Multics Change Request

MCR 1612
Page I of I

TITLE: Remove fault-time lock-looping in page control

AUTHOR: Bernard S. Greenberg

STATUS	DATE
Written	1/26/76
Status	A 02/03/76
Expires	08/03/76

-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
	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 #)	61, 73
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	
Other (Name)	
None (Reason)	

Objections/Comments:

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

SUMMARY: Multiprogram when the page-table lock is found locked by a pendant page fault in a two-or-more CPU system. The current system loop-locks.

REASONS: Performance. This change has been shown to buy back one-half of a cpu on a three-cpu configuration.

IMPLICATIONS: Substantial speedup in multi-cpu configurations.

Even greater complexity and obscurity in an already arcane area of the system.

Detailed Proposal:

A cell is maintained counting processes waiting on the page table lock. This is incremented before an attempt is made at fault time to lock this lock. It is decremented whena process successfully locks the lock. When any process unlocks the page table lock, it chekcs this cell and zeros it. If it was non-zero, an event identified with the page table lock is notified. If a process finds the page table lock locked at fault time, the traffic controller page_wait interface will be called to wait on the page-table event. In this case, the page table lock will not be und ocked by this interface (to do so is the default action). The validity of the event is then checked under the traffic control lock. If still locked, the proacessor is given away, the special cell being guaranteed non-zero. If the lock has been unlocked in the window, the process retries the entire locking procedure. The default unlcking action of the page_wait interface is made to notify the necessary event when unlooking the page-table lock when the special cell is non-zero.

Ver. 3
741022

MULTICS CHANGE REQUEST

MCR 1613

TITLE: Fix problems in syserr

AUTHOR: Larry Johnson

STATUS	DATE
Written	01/30/76
Status	A 02/10/76
Expires	07/30/76

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 (X)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) MPAM (sect)
MOSN (sect) MSAM (sect)
PLMs (AN#)
Info Segs
Other
None (reason)

OBJECTIONS/COMMENTS:

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

Summary

The logging of I/O errors by ioi_interrupt has resulted in a 5-fold increase in syserr message traffic. This has revealed some flaws in the syserr mechanism that must be fixed.

Detailed Proposal

1. Many unnecessary messages are logged. About 60% of the ioi_interrupt messages are caused by printer slews past top of form. There is no need to log this. To avoid this, Data Alert and Command Reject errors for printers will not be logged. This will lose some real errors, but will temporarily solve the problem. A more general fix will be in release 4.0.

2. Equal syserr messages were not handled correctly if a message that was logged appeared between two equal messages that were written. This will be fixed. A byproduct of this is that "=" may be logged for two indentical messages whether they are written or not, resulting in smaller logs.

3. The syserr code of the ioi_interrupt message will be changed from 4 to 5. This means that these messages will not be printed if the logging mechanism can not keep up with the load.

```
 ------------------------------------------------------------------------------
| Ver. 3                                                  |                    |
| 741022            MULTICS CHANGE REQUEST                |  MCR__1614_______  |
|_________________________________________________________|____________________|
| TITLE:  tape-in, tape-out: a new tape-to-storage        |_STATUS__|_DATE_____|
|         file transfer facility.                         |_Written_|_1/19/76__|
| AUTHOR: J. Phillipps                                    |_Status__|A02/10/76_|
|_________________________________________________________|_Expires_|_07/19/76_|
| Planned for  System:  not applicable                    |____________________|
| Fixes Bug Number(s):  not applicable                    |_CATEGORY (check one)|
| Documented  in  MTB:  239 and 235.                      |( )Lib. Maint. Tools|
| Incompatible Change:  no                                |( )Sys. Anal. Tools |
| User/Operations-visible Interface Change:  yes          |( )Sys. Prog. Tools |
| Coded in: (☒)PL/I ( )ALM ( )other-see below             |( )355              |
| Performance: ( )better (☒)same ( )worse                 |( )BOS              |
|_________________________________________________________|( )Salvager         |
|_DOCUMENTATION CHANGES (specify one or more)_____________|( )Ring Zero        |
| MPM (vol,sect) tape_out   MPAM (sect)                   |( )Ring One         |
| MOSN (sect)               MSAM (sect)                   |( )SysDaemon/Admin  |
| PLMs (AN#)                                              |( )Runtime          |
| Info Segs                                               |(☒)User Command/Subr|
| Other                                                   |                    |
|_________________________________________________________|____________________|
| OBJECTIONS/COMMENTS:                                                         |
|                                                                              |
| For interchange with GCOS, use ansii formats.             (MPM volume.       |
|_We will consider moving all user level I/O documentation to seperate_________|
Headings are:  SUMMARY, REASONS, IMPLICATIONS, DETAILED PROPOSAL (optional)
```

SUMMARY:

Install tape_in, tape_out: two commands which facilitate file transfer between the storage system and magnetic tape.

REASONS:

Unsophistocated users with interchange tapes need a tool to make file transfer to and from Multics in a manageable and familiar manner. Once a tcl control file has been written, a user may transfer files back and forth between the storage system and tape with ease and without being confronted with iox_.

DETAILED PROPOSAL:

See documentation attached.

<u>Name</u>: tape_in, tin
tape_out, tout

These commands allow the user to transfer files between the storage system and magnetic tape. tape_in reads from tape to the storage system; tape_out writes from the storage system to tape. To accomplish a file transfer, the tape_in and tape_out commands access either the tape_ansi_ or the tape_ibm_ IO module for the tape interface, and the vfile_ IO module for the storage system interface. Unstructured format storage system files (for stream I/O) and sequential format storage system files (for record I/O) may be specified; 9-track ANSI standard labeled tapes, 9-track IBM standard labeled tapes, and any 9-track unlabeled tape which is structured according to OS standard, may be be read or written.

<u>Usage</u>: tape_in pathname -control_args-
tape_out pathname -control_args-

pathname
is the path rame of the control file which governs the file transfer. If pathname does not end with the suffix .tcl, .tcl will be supplied.

control_args:

1) severity<u>i</u>, -sv<u>i</u>
causes the tape_in, tape_out compiler's error messages with severity less than <u>i</u> (where <u>i</u> is 0, 1, 2, 3, or 4) not to be written into the "error_output" IO stream. The default value for <u>i</u> is 0. See APPENDIX A, <u>Error Diagnostics</u>, for further information on error reporting.

2) -check, -ck
specifies that only semantic checking be done on the tcl control file. No tapes will be mcunted if this option is specified.

3) -force, -fc

specifies that the expiration oate of a tape file to be overwritten is to be ignored. This option extends unconditional permission to overwrite a tape file, regardless of the file's "unexpired" status. This unconditional permission supresses any query made by the IO module to inquire about tape file's expiration date. The -force option is only meaningful when used with the tape_out command. If the -force option is used with the tape_in command, an error is indicated.

Part I: THE BASIC TCL CONTROL FILE

The control file which governs the file transfer is actually a program, written by the user, in the tape control language (tcl). The contents of this control file describes the file transfer(s) to take place. When the user issues the tape_in or tape_out command, the control file named in the command line (pathname) is compiled and if the compilation is successful, the generated object code is executed to accomplish the desired file transfer(s). The same control file may be used with both the tape_in command (to read a file from tape into the storage system) and with the tape_out command (to write a file from the storage system onto tape).

Example:

simple.tcl

```
Volume: 012345;
File: File_1;
Path: >udd>Project>User_dir>demo;
End;
```

All tcl control files must have at least four statements: a Volume statement, a File statement, a Path statement and an End statement; all other tcl statements are optional. The simplest control file has just these four statements, as in the example above. This example control file relies on tcl control file defaults, which are listed below in the section <volume-group> Defaults. The File transfers possible with this sample control file are two: either writing tape file "File_1" from storage system file "demo", or writing storage system file "demo" from tape file "File_1".

The Volume Statement

Volume: <volid>;

The Volume statement specifies the tape volume to be used in file transfer. This statement causes a tape volume whose volume identifier is <volid> to be mounted on a 9-track drive. <volid> must consist of from 1 to 6 ASCII characters. If <volid> is less than 6 characters and numeric, it will be padded on the left with zeros to a length of six. If <volid> is less that 6 characters and not numeric, then it is padded on the right with blanks to a length of six. If <volid> contains any of the following characters, <volid> must be enclosed in quote characters ("):

1) any ASCII control character
2) : ; , or blank
3) the sequence /* or */
4) If <volid> itself contains a quote character, the quote must be doubled and the entire <volid> string enclosed in quotes.

Examples

Volume: 23;	mounts volume 000023
Volume: 001234;	mounts volume 001234
Volume: XJ56;	mounts volume XJ56ƀƀ
Volume: "as"";56";	mounts volume as";56
Volume: -00451;	mounts volume -00451

The File Statement

File: <fileid>;

The File statement specifies that a tape file is to be read or written. The tape file is identified by <fileid>. <fileid> must be from 1 to 17 characters for ANSI labeled tapes, and must be a valid DSNAME for IBM labeled tapes. A valid DSNAME consists of 1 to 8 characters. The first character must be an alphabetic or national (@,$,#) character; the remaining characters can be any alphameric or national characters, a hyphen, or a plus zero (12-0 punch). <fileid> for IBM unlabeled tapes, which are discussed below, is the character "*". The File statement marks the beginning of any local attributes for a given tape file transfer.

The Path Statement

Path: <pathname>;

Associated with every File statement must be one Path statement. The Path statement specifies the path name of the storage system file to be read or written. <pathname> may be either a relative or absolute path name.

The End Statement

Associated with every Volume statement must be an End statement, to mark the end of the TCL for that volume. <volume-group>.

End;

Part II: SOME TCL CONTROL FILE OPTIONS AND DEFAULTS

The TCL Program

tcl statements must begin with a keyword, may or may not take a keyword argument, and must end with a semi-colon. <global-statement> keywords and the "Volume", the "File", "Path", and the "End" statement keywords must begin with an uppercase letter. <local-statement> keywords must begin with a lower case letter.

A tcl program consists of one or more <volume-group>s.

<volume-group>

A <volume-group> is a series of statements which specifies the file transfer(s) to be performed between the storage system and a particular tape volume. A <volume-group> must begin with a Volume statement, contain one or more <file-group>s, and terminate with an End statement. In addition, a <volume-group> may optionally contain one or more <global-statement>s which apply to all the file groups within the <volume-group> and one or more <local-statement>s which applies to the current <file-group> only.

<volume-group> Defaults

Associated with a <volume-group> are a set of default characteristics. In the absence of overriding <global-statement>s or <local-statement>s, these defaults will apply to all <file-group>s within the <volume-group>. If no tape-type is specified in the control file, ANSI standard labeled tape will be assumed. If, however, a tape-type is specified (in the Tape statement), the <volume-group> defaults for that tape-type will preside until overridden.

Tape: ANSI;   or no Tape statement

1) density: 800 bpi
2) file expiration: immediate
3) storage system file format: unstructured
4) mode: ascii character code
5) tape file record format: spanned, blocked
6) physical block length: 2048 characters (maximum)
7) logical record length: 1044470 characters (maximum)

Tape: IBMSL | IBMNL | IBMDOS;

1) density: 1600 bpi
2) file expiration: immediate
3) storage system file format: unstructured
4) mode: ebcdic;
5) tape file record format: variable length, blocked
6) physical block length: 8192 characters (maximum)
7) logical record length: 1044480 characters (maximum)

<global-statement>

A <global-statement> changes a <volume-group> default. The Tape and the Density <global-statement>s may appear only once in a <volume-group>. The Storage, Mode, Format, Block and Record <global-statement>s may appear any number of times within a <volume-group>.

Tape: <tape-type>;

The Tape <global-statement> specifies the kind of tape which will be processed. <tape-type> may be IBMSL for IBM standard labeled tape, IBMNL for IBM unlabeled tape, IBMDOS for IBM DOS standard labeled tape, or ANSI for ANSI standard labeled tape. The tape label processing is done automatically by the IO module in use. This <global-statement> may appear only once within a <volume-group> or an error is indicated.

Density: <den>;

The Density <global-statement> indicates the density in which the volume is (to be) recorded <den> must be either "800" or "1600" or "2" or "3" (for IBM compatibility) to indicate 800 or 1600 bpi respectively. WARNING: the use of 1600 bpi for ANSI interchange tapes is non-standard. It will not be standard for interchange. This <global-statement> may appear only once within a <volume-group> or an error is indicated.

Storage: <structure>;

The Storage <global-statement> states the internal (logical) structure of the storage system file(s) to be specified by subsequent <file-group>s. The unstructured file is referenced as a series of 9-bit bytes, commonly called lines: the sequential file is referenced as a sequence of records, each record being a string of 9-bit bytes. <structure> must be either "unstructured" or "sequential". When an unstructured file is written into the storage system from a tape, the NL character is appended as each line is written, unless the record already ends in a NL character, in which case nothing further is appended. When an unstructured file is written from the storage system to tape the

NL character is stripped off before writing the tape record. If a line of an unstructured file consists of just a NL character, it is written to tape as a zero length record. If the Storage <global-statement> is omitted from a control file <volume-group>, the assumed storage system file format will be "unstructured". If, then a sequential file is referenced within that <volume-group>, the results are undefined and an error is indicated. Processing is terminated on that file in which the error is indicated.

Mode: <mode>;

The Mode <global-statement> specifies the tape mode and character code to be used with subsequent <file-group>s. <mode> may be either "ascii"" or "ebcdic" for IBM tapes (using tape_ibm_ IO module) and may be either "ascii", "ebcdic", or "binary" for ANSI tapes (using tape_ansi_ IO module). WARNING: the use of ebcdic mode or binary mode is not standard for ANSI tapes.

Note: Refer to APPENDIX E for a description of the interaction between a given combination of format, block and record specification. Values must be carefully chosen to ensure desired results.

Format: <form>;

The Format <global-statement> specifies the tape record format to be used with subsequent <file-group>s. <form> must be either "U", "F", "FB", "D", "DB", "S", of "SB" for ANSI tapes (using tape_ansi_ IO module) and "F", "FB", "U"","V", "VB", "VS", "VB", "or "VBS" for IBM tapes (using tape_ibm_ IO module).

Block: <blklen>;

The Block <global-statement> specifies the tape file (maximum) physical block length, in characters, to be used with subsequent <file-group>s. <blklen> must be a decimal integer, such that 18 ≤ <blklen> ≤ 8192. For output, <blklen> must be evenly divisible by 4! WARNING: <blklen> greater that 2048 does not comply with the ANSI standard for tapes.

Record: <reclen>;

The Record <global-statement> specifies the tape file (maximum) logical record length, in characters, to be used with subsequent <file-group>s. <reclen> must be a decimal integer, such that 1 ≤ <reclen> ≤ 1044480.

<file-group>

Every <volume-group> must contain one or more <file-group>s. Each <file-group> defines one tape to storage system file transfer. A <file-group> must begin with a File statement, and contain a Path statement. In addition, it may contain one or more <local-statement>s. A <file-group> is terminated by a <global-statement>, an End statement, or a File statement (new <file-group>).

<local-statement>

A <file-group> may contain one or more <local-statement>s. A <local-statement> overrides the <volume-group> defaults in effect at the time a <file-group> is evaluated. A <local-statement> has no effect outside of the <file-group> in which it occurred, and may appear anywhere within the <file-group>.

The storage, mode, format, block and record <local-statement>s operate exactly as do their <global-statement> counterparts, except that they affect only the <file-group> in which they are contained.

The number Statement

number: <number>;

The number <local-statement> further identifies the tape file to be used in file transfer. <number> is the file sequence number; it specifies the position of the file on the specified tape volume. <number> must be either an integer between 1 and 9999 inclusive, or the character "*". For output, <number> = * appends the current file to the tape volume. If the control file is to be used with the tape_in command, <number> specified in a number statement, must correspond with a file on the specified tape volume. If both the <fileid> in the File statement and the <number> in the number statement are specified in the <file-group>, they must identify the same tape file; otherwise an error is indicated. When reading unlabeled tapes, the number statement is required to identify the file to be read.

When the control file is to be used with the tape_out command, the number statement is optional. If the number statement is given in a control file for use with the tape_out command, the file location specified in the number statement <number>, is the location where the the first file of the <volume-group> will be written on the tape. Otherwise, with no number statement, the first file to be written in a <volume-group> will be the first file position on the tape. Subsequent files on that volume will be appended after the first file.

Control File Execution

When the tcl control file is being executed in response to the tape_in command, the volume named in each <volume-group> of the control file is mounted in turn without a write ring. If any file output options appear in a control file being executed in response to the tape_in command, these statements will be ignored. When the control file is being executed in response to the tape_out command, the volume named in each <volume-group> of the control file is mounted in turn with a write ring.

Control File Comments

Comments may be inserted anywhere within the tcl program by surrounding the comment text with the comment delimiters. /* is the delimiter which begins a comment, and */ is the delimiter which terminates a comment.

Notes

File transfer is performed as described below. See APPENDIX A for the amount of data returned per io call for each tape record format.

1) tape_in: one logical record is read from the tape file, and as many characters as were read are written into the storage system file either as a line with new-line (NL) character appended, if necessary, (unstructured case) or as one logical record in a sequential format file.

2) tape_out: an attempt is made to read either a line or a record from the storage system file to be written onto tape. For unstructured format storage system files, a line read is a line from the file up to and including the first new-line character (NL) encountered; for sequential format storage system files, a record read is one logical record of the file. The characters read from the storage system are then written on the tape as one logical record of the tape file.

Under certain circumstances, tape records being written must be padded in accordance with a set of per-format padding rules. (For a discription of record and block padding for all formats, see the MPM write-ups of tape_ansi_ and tape_ibm_.) Because of padding rules and treatment of new-line characters when writing tape, a file that is written out to tape may not appear the same when read back in from tape. It is therefore recommended that the following suggestions be heeded:

2) to write character data -- that is source files or text files, use the defaults, or with tape_ansi_, use D, DB, S, or SB format, with the maximum block length, and the record length chosen so that the amrl -- see below Appendix E, is greater than the longest line in the storage system file. to avoid unwanted pad characters resulting from block padding, do not use F or FB format.

3) to write binary data with tape_ansi_, use S or SB format, with the maximum permissible block and/or record lengths.

4) to write character data with tape_ibm_, use VBS format with the maximum block length, and the record length chosen so that the amrl -- see below in Appendix E, is greater than the longest line in the storage system file. (VB may cause 1 - 3 blanks to be appended to lines.)

5) when transfering sequential format files to tape, use a variable length record format (D, DB, S, or SB with tape_ansi_ and V, VB, or VBS with tape_ibm_) to avoid unwanted padding characters being inserted into records. (VB may cause 1 - 3 blanks to be appended to lines.)

Examples

Below are examples of two typical control files. In the first example, the user wishes to produce 2 tapes, one for Multics, the other for an OS installation. The Multics tape will contain the source code of user subsystem SUBSYS, as well as it's object code. The OS tape will contain only the source code. In the second example, the user wishes to load into the storage system, the contents of volume "2314dp" which contains a dump of a disk pack containing source and data.

The numbers at the left-hand side of the page in the examples below do not actually appear in the control file, but are included only for annotation reference.

Example: sample1.tcl

```
   command line:      tape_out   sample1.tcl


   1          Volume: 001234;
   2          /* Dump source in DB and object in SB format */
   3          File: FILE_1;
   4          Path: SUBSYS.pl1;
   5          File:  FILE_2;
   6          mode: binary;
   7          Path: <object>SUBSYS;
   8          format: SB;
   9          End;
   10         Volume: DFG054;
   11         /* append source to tape */
   12         Tape: IBMSL;
   13         File:  TEST_SAVE;
   14         format: VBS;
   15         block: 4096;
   16         storage: sequential;
   17         Path: SUBSYS.pl1;
   18         number: 3;
   19         End;
```

Annotations for sample1.tcl

1) causes volume 001234 to be mounted. The volume defaults are set to ANSI standard labeled tape-type, 800 bpi density,, ascii encoding mode, DB record format, block length = 2048, and record length = 1044480.

2) is a comment in the control file. As the storage statement is missing, the default storage system file format will be set to transfer unstructured files.

3) as there is no number statement, the default causes the tape to be positioned so that FILE_1 will be created as a new file at the first file position on the tape volume.

4) specifies the path name of the storage system file to be written to tape. As the <file-group> contains no <local-statement>s the file will be written according to the current volume defaults.

5) causes the tape to be positioned so that the file to be written will be appended at file position two on the tape volume.

6) specifies that the file is to be written in binary encoding mode.

7) specifies the path name of the storage system file to be written to tape.

8) specifies that the file is to be written in SB format. Note that the block length will be the current volume default block length (2048) and the record length will be the current volume default record length (1044480).

9) signifies end of <volume-group>. The IO switch is closed and detached. The volume set is taken down and the drive is released.

10) causes volume DFG054 to be mounted.

11) is a comment.
Storage format is still unstructured.

12) changes tape-type to IBM standard labeled changes the <volume-group> defaults to those associated with IBMSL: 1600 bpi, ebccic, VB format, block length = 8192, and record length = 8188.

13) specifies name of file to be written onto tape.

14) changes the record format to VBS. We want a spanned record format here for transfering a sequential file, so that unwanted block padding does not get inserted into the file as it is transfered. The default record length for VBS format is 1044480 bytes.

15) changes the block length to 4096.

16) storage system file is sequential format.

17) specifies the path name of the storage system file to be written.

18) this number statement is required to make sure the file appended to an already existing tape volume. Without this number statement, the file would be created as the first file on the tape volume, overwriting any existing files. If the tape is empty, that is if files one and two do not exist, an error is indicated, but if the tape is not empty, the file will be written at file position t

19) causes the volume to be rewound and taken down as no more <file-group>s in the control file reference the current tape volume.

Example: sample2.tcl

command line: tape_in sample2.tcl

```
1         Volume: 2314dp;
2         /* Source Pack being loaded */
3         Tape: IBMSL;
4         Storage: unstructured;
5         Density: 800;
6         Format: fb;
7         Record: 80;
8         Block: 800;
9         File: FILEX;
10        Path: <setup>data_entry>FILEX;
11        File: FILEXX;
12        Path: <setup>data_entry>FILEXX;
13        File: FILEY;
14        Path: <setup>data_entry>FILEY;
15        File: FILEYY;
16        Path: <setup>data_entry>FILEYY;
17        File: FILEZ;
18        Path: <setup>data_entry>FILEZ;
 .
 .
 .
58        File:  FILEZZ;
59        Path: <setup>data_entry>FILEZZ;
60        End;
```

Annotations for sample2.tcl

1) mounts the volume 2314dp with a write ring.

2) comment

3) read an IBM standard labeled tape.

4) storage system the files will be created in unstructured format, ready for use in stream i/o. NL characters will be appended as the file is written to disk.ice that the mode will be the default for IBMSL tape-type, namely, ebcdic.

5) tape is recorded at 800 bpi.

6) all files on tape are in fixed block format. Possible record padding problems may be encountered.

7) all logical records are 80 characters (card image files).

8) all files blocked to 800 characters.

9) first file to be read from tape is named FILEX. It may be at any file location on the taoe. The tape is automatically positioned to the file by name.

X0) read tape file, FILEX, into storage system file named FILEX. The relative path name, <setup>data_entry>FILEX, will be expanded.

11) continue reading files off the tape volume, one by one, into files in the storage system with the same name.

.
.
.

60) end of <volume-group> and end of control file.

Part III: ADDITIONAL OPTIONS AVAILABLE FOR THE TCL USER

A number of options are available to the user who wants to do more than the simple file transfer between storage and a new tape volume. These features need not be of concern to most users, but for the user with specialized needs, these additional options are explained below.

Multi-volume Files

Multi-volume files are specified in a control file by slightly more complicated Volume statement than shown above. The multiple <volid>s of such a volume set are separated from one another by commas and are listed either in the order in which they became members of the volume set, for input, or in the order in which they are candidates for volume set membership, for output. The entire volume set membership need not be specified in a Volume statement referencing a volume set, but the first (possibly only) member must be mentioned. Up to 64 <volid>s may be specified in a single control file Volume statement.

Volume switching for multi-volume files is handled automatically by the IO modules. If sufficient volume set members are given in the tcl control file, the volume switching will be transparent to the user. If insufficient members of a volume set are given or the membership is being developed, the user will be querried during execution, for names of additional volume set members.

Sending Messages to the Operator

If it is necessary for the user to have a message displayed on the operator's console, the comment phrase can be included in the Volume statement. The comment phrase consists of the keyword -comment followed by the text of the message; this phrase in enclosed in quotes. Whenever the volume with the <volid> immediately preceding the comment phrase is to be mounted, the specified message will be displayed on the operator's console. The message may be concerned with any subject, but it is typically used to display the slot identifier of the tape being mounted when it differs from the volume label. The message, **text**, may be from 1 to 64 characters and must be a contiguous string with no embedded spaces.

```
        Volume: 060082 -comment "tape_is_Smith's", 060083
                -comment "tape_also_Smith's";
```

Protecting Tape Files From Accidential Overwriting

```
                        Expiration: <date>;
                        expiration: <date>;
```

The Expiration <global-statement> and the expiration <local-statement> specify the expiration date of a file to be written (created). <date> must be a contiguous string, with no embedded spaces and must be of a form acceptable to the convert_date_to_binary subroutine, for example "09/12/77". (see MPM writeup on convert_date_to_binary subroutine.) Because overwriting a file on a tape logically truncates the file set at the point of overwriting, the expiration date of a file must be eariler than or equal to the expiration date of the previous file (if any) on the tape; otherwise an error is indicated. If an attempt is made to overwrite an unexpired file, the user will be querried for explicit permission at the time of writing, unless the -force option was specified in the command line.

Special Output Modes

Normally, when a user sets up a tcl control file <file-group> to write a storage system file onto a tape volume, that is for use with the tape_out command, it is intended that a new file be created on the tape volume. The tcl default output mode is create. This is the only output mode available for unlabeled tapes. For labeled tapes however, the tcl language offers three additional specialized output modes; they are replace, modify, and tape_extend. The replace mode causes the tape file labels to be rewritten using specified and default file structure attributes. The tape_extend and modify <local-statement>s do not cause the tape file labels to be recomposed, so any file attributes specified in the control file <file-group> or <volume-group> which do not match those recorded in the tape labels, will cause an error to be indicated. The description of the three <local-statement>s for specialized output mode designation follow.

replace: <fileid>;

If an existing tape file is to be replaced on an ANSI or IBM standard labeled tape and its name is known, the file to be overwritten is identified by <fileid> in the replace <local-statement> and the new file to be written is identified by <fileid> in the File statement. If the file identified in the replace statement does not exist, an error is indicated.

```
File:  <fileid>;
replace: <fileid>;
```

tape_extend;

The tape_extend <local-statement> allows new data records to be appended to an existing file on an ANSI or IBM standard labeled tape without in any way altering the previous contents of the tape file. This statement cannot be used for unlabeled tapes. The tape file to be extended is identified by the File statement or by the File statement and number <local-statement> in combination. If the tape file to be extended does not exist on the tape, an error is indicated. Recorded in the labels of an ANSI or IBM labeled tape file is the version number. Initially it is zero when the file is created. Every time a file is extended, its version number is incremented. The version number field is two digits and is reset to zero when the one-hundredth revision is made.

modify;

The modify <local-statement> causes the entire contents of a file on an ANSI or IBM labeled tape to be replaced while retaining the structure of the file itself. The file to be modified is identified by the File statement, or by a combination of the File statement and the number statement.

storage_extend:

Normally when a user sets up a tcl control file <file-group> to transfer a tape file to a storage system file, it is intended that a new file be created in the storage system. Should the user want to extend an already existing file in the storage system, that is should he want to append to a storage system file, the <local-statement> "storage_extend" may be used in the tcl control file. If the storage system file to be extended does not exist, an error is indicated. If the storage_extend <local-statement> exists in a control file used with tape_out, it will be ignored.

Example: sample3.tcl

```
   command line:      tape_out   sample3.tcl   -fc
```

```
     1      Volume: 070067 "-comment in_slot_10000" 070068;
     2      Tape: ANSI;
     3      File: BIG_LISTING;
     4      replace: FILE_20;
     5      number: 20;
     6      expiration: 2weeks;
     7      format: db;
     8      block: 2048;
     9      record: 133;
     10     Path: >udd>Example>Mega>test.list
     11     End;
```

Annotations for sample3.tcl

1) The first member of the volume set, 070067, is mounted with a ring, displaying the message "in_slot_10000" on the operator's console. Later if necessary, the volume set member 070068 may be mounted to continue writing a large listing file. No message will appear upon mounting the second member of the volume set.

2) writing an ANSI standard tape.

3) tape file named BIG_LISTING, into which the storage system file is to be written.

4) is to replace tape file named FILE_20.

5) by the number statement FILE_20 is the 20th file on the current volume set.

As no density statement is included in the control file, the default for tape_ansi_, 800 bpi, will be used.

Upon execution of the control file, the tape will be positioned at the 20th file automatically, providing 20 files exist on the tape.

As no Storage statement is present in the control file, the default storage system format is unstructured, and as the files are written to tape, the NL character will be stripped.

6) The file, BIG_LISTING, will be protected against accidental overwriting for two weeks, meaning that if the user attempts to overwrite the file within that time, he will first be querried for permission to do so.

The -force option in the command line will inhibit a query for permission to overwrite FILE_20, in case it has not yet expired.

7) BIG_LISTING will be recorded in variable length blocked record format.

Mode will be the default for tape_ansi_, namely ascii.

8) Block length is maximum allowed for ANSI interchange standard, 2048.

9) record length is 137, for printer line width plus 4 for RDW.

10) the listing file will be transfered from test.list in the storage system.

11) signifies termination of <volume-group> and of control file.

If after putting his listing file out onto tape, the user then wishes to delete the on-line listing, and at a later time, read the listing back from tape into storage, he might type the command line

```
          tape_in sample3.tcl
```

The output statements in the control file, namely the replace <local-statement> and the expiration <local-statement> will be ignored on input.

370/DOS Tapes

The tape_ibm_ IO Module will process tapes created by or destined for IBM/DOS installations as well as tapes for IBM/OS installations. The DOS <global-statement> is used in the tcl control file to specify that the tape files referencec by the given <volume-group> are destined for or have been produced by a IBM/DOS installation. The important difference between tape files created by OS and those created by DOS operating system is that the tape file structure attributes are not recorded in the tape labels under DOS. It is therefore necessary for all of the structure attributes of a DOS tape file, namely encoding mode, logical record format, logical record length, and block size to be specified in the tcl control file. If not all tape file structure attributes have been specified by <global-statement>s and <local-statement>s for each <file-group> in the scope of the DOS <global-statement>, an error is indicated.

Example: sample4.tcl   Control File for Reading DOS Tape

command line: tape_in sample4.tcl

```
Volume:  042281;
Tape: IBMDOS;
Density: 800;
Storage: unstructured;
Mode: ebcdic;
File: abc;
record: 80;
block:  800;
format: fb;
Path: >udd>Example>Foo>fargo.pl1
End;
```

Annotations for sample4.tcl

Note: Only selected statements in the control file are annotated here.

1) mount volume 042281 without a ring.

2) read IBM DOS standard labeled tape.

5) read tape files into storage system as structured format files appending NL characters to each record from tape.

Unlabeled Tapes

The tape_ibm_ IO Module supports processing of unlabeled tapes, provided that the tapes are structured according to the OS standard. DOS leading tape mark (LTM) unlabeled format tapes cannot be processed however. The IBMNL specification as a Tape statement argument, is mutually exclusive with any statement, global or local, which refers to labeled tapes: namely, the Expiration <global-statement> and the replace, modify and tape_extend <local-statement>s. If any of these appear together within the same control file <file-group>, an error is indicated. When referencing unlabeled tape files in a given <file-group>, the argument of the File statement, <fileid>, is specifiec by "*", and the tape file desired must be specified by the number <local-statement>.

Example: sample5.tcl Control File for Reading an Unlabeled Tape

```
   command line:    tape_in  sample5.tcl


      1       Volume: 042381;
      2       Tape: IBMNL;
      3       Storage: sequential
      4       File: *;
      5       Format:  VBS
      6       number: 3;
      7       Path: >udd>Expamle>Foo>foobar.data
      8       End;
```

<u>Annotations for sample5.tcl</u>

Note: Only selected statements in the control file are annotated here.

3) unlabeled tape is to be read. Files will be unnamed. This statement must appear when processing unlabeled tapes.

6) <fileid> is specified by "*" for unnamed files.

6) the number statement must be present when processing unlabeled tapes. The third file on the tape will be read.

The tape file record format is VBS, the tape file record length for VBS format is 1044480 characters, and the tape file block length is 8192 characters.

APPENDIX A

Error Diagnostics

The error messages which are issued during tape_in, tape_out compilation are graded and have the form shown below.

prefix error_number, SEVERITY severity IN STATEMENT m OF LINE n
text_of_error_message
SOURCE:
source_statement_in_error

where n is the line number on which the described statement began and m is a number identifying which statement in line n was in error. If line n contains only one statement then "STATEMENT m OF" is omitted from the error message.

The severity numbers produce one of the following prefixes:

severity	prefix	explanation
0	COMMENT	the error message is a comment.
1	WARNING	the error message warns that a possible error has been detected. However, the translation will still proceed.
2	ERROR	the error message warns that a probable error has been detected. However, the error is non-fatal, and the translation will still proceed.
3	FATAL ERROR	the error message warns that a fatal error has been detected. Processing of the input will still continue to diagnose further errors, but no translation will be performed.
4	TRANSLATOR ERROR	the error message warns that an error has been detected in the operation of the translator. No translation will be performed.

APPENDIX B

Execution Time Errors

Any fatal error from an IO module during execution of a control file will cause the user to be querried as to whether or not he wishes to continue processing the other <file-group>s and <volume-group>s in the control file or whether to terminate processing of the control file. Specific errors are described below.

storage system file not found

If a control file is being executed and a storage system file given in the control file cannot be located, the user will be querried as to whether he wishes to halt the processing of the control file and see if he can fix the problem of the missing storage system file, or whether he wishes to continue processing onto the next <file-group> or <volume-group> in the control file, if one exists or terminate processing if no other <file-group> or <volume-group>s exist. The user will resume processing after locating the file by typing "start".

tape file not found

If a control file is being executed and a specified tape file is not found, the user will be querried as to whether he wishes to continue processing the control file or whether he wishes to terminate processing.

tape volume not found

If a tape volume specified in a control file cannot be located, the user will be querried as to whether he wishes to continue processing the control file's other <volume-group>s or whether he wishes to terminate processing.

file name duplication

If a control file is being executed, and a storage system file with the path name specified in the control file already exists, the user will be querried as to whether he wishes to overwrite the existing storage system file or not. If he choses not to overwrite, he will be querried as whether to terminate processing or continue on to the next <file-group> in the control file.

APPENDIX C

Some tcl statements are for use with one IO module only. Below is a summary of tcl <global-statement>s and the name of the IO module which supports the tcl statement.

<global-statement>	Supporting IO Module
Density: <den>;	tape_ansi_ or tape_ibm_
Tape: <tape-type>;	tape_ansi_ or tape_ibm_
Storage: <structure>;	
Expiration: <date>;	tape_ansi_ or tape_ibm_
Mode: <mode>;	tape_ansi_ or tape_ibm_
Format: <form>;	tape_ansi_ or tape_ibm_
Block: <blklen>;	tape_ansi_ or tape_ibm_
Record: <reclen>;	tape_ansi_ or tape_ibm_

APPENDIX D

Summary of tcl <local-statement>s and the IO Module which supports the tcl statement

<local-statement>	Supporting IO Module
storage: <structure>;	
expiration: <date>;	tape_ansi_ or tape_ibm_
mode: <mode>;	tape_ansi_ or tape_ibm_
format: <form>;	tape_ansi_ or tape_ibm_
block: <blklen>;	tape_ansi_ or tape_ibm_
record: <raclen>;	tape_ansi_ or tape_ibm_

Additional <local-statement>s which have no global counterparts follow. Again, some <local-statement>s are for use with specific IO modules only.

number: <number>;	tape_ansi_ or tape_ibm_
replace;	tape_ansi_ or tape_ibm_
tape_extend;	tape_ansi_ or tape_ibm_
storage_extend;	
modify;	tape_ansi_ or tape_ibm_

APPENDIX E

IO Module Compatibility and nelem Tables

tape_ansi_

mode: ascii (default) | binary | ebcdic
block length: 18 ≤ b ≤ 2048 bytes
for output mode block length must be devisible by 4.
density: d = 800 (default) | 1600
file sequence number: 1 ≤ n ≤ 9999 or *
record length: < r < 1044480
format: f = fb | f | db (default) | d | s | sb | u

tape_ibm_

mode: ascii | ebcdic (default)
block length: 18 ≤ b ≤ 8192 bytes
for output mode block length must be devisible by 4.
density: d = 800 | 1600 (default)
file sequence number: 1 ≤ n ≤ 9999 or *
record length:.. ≤ r ≤ 1044480
format: f = fb | f | vb (default) | v | vbs | u

Format	Record Length in bytes r	Block Length in bytes b
u	r is undefined	$= \mathrm{amrl} \le b \le 8192$
f	$r = \mathrm{amrl}$	$b = r$
fb	$r = \mathrm{amrl}$	b must statisfy $\mathrm{mod}(b,r) = 0$
d	$\mathrm{amrl}+4 \le r \le 2048$	$b = r$
db	$\mathrm{amrl}+4 \le r \le 2048$	$b \ge r$
s	$\mathrm{amrl} \le r \le 1044480$	$18 \le b \le 2048$
sb	$\mathrm{amrl} \le r \le 1044480$	$18 \le b \le 2048$
v	$\mathrm{amrl}+4 \le r \le 8188$	$b = r + 4$
vb	$\mathrm{amrl}+4 \le r \le 8188$	$b \ge r + 4$
vs	$\mathrm{amrl} \le r \le 1044480$	$20 \le b \le 8192$
vbs	$\mathrm{amrl} \le r \le 1044480$	$20 \le b < 8192$

Notes:

amrl is the actual or maximum record length of a given record format, i.e., the actual or maximum number of characters which can be recorded in a logical record. The value of r is dependent on the choice of record format. In every case b must be an integer in range of $18 \le b \le 8192$. For ANSI tapes, in order to comply with the ANSI standard, b must be in the range of $18 \le b \le 2048$. For IBM tapes, the condition $\mathrm{mod}(b,4) = 0$ must be satisfied. The tcl record statement should not be used for a U-format file transfer.

(END)

750508

Multics Change Request

MCR 1613
Page 1 of 1

TITLE: Avoid locking indexed files on passive shared operations

AUTHOR: M. Asherman

STATUS	DATE
Written	01/28/76
Status	A 02/10/76
Expires	08/10/76

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

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

DOCUMENTATION CHANGES

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

Objections/Comments:

documentation attached

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

SUMMARY:

Alter vfile_ to avoid unnecessary locking/unlocking of files which are subject to concurrent updates by other processes. Specifically, operations which do not inherently alter a file (e.g., read_record, seek_key, etc.) should never have to set the file's lock for the sake of synchronizing access.

REASONS:

The current scheme has the following disadvantages:

1) forces users to have write access to shared files, even if no updates intended;
2) causes passive operations to tie up files, resulting in degradation of performance with heavy passive use;
3) imposes additional cost of two external calls to each shared operation.

IMPLICATIONS:

The above disadvantages disappear. Shared files can be used under AIM without need for special ring 1 segments. Additional cost of shared operations is reduced when the file is not actively being updated.

DETAILED PROPOSAL:

Passive entries will wait for file lock to be clear, perform operation, verify results by checking change count, and possibly repeat this procedure until a valid result is obtained or the wait-time is exhausted.

vfile_ MPM documentation changes

Opening and Access Requirements

All opening modes are supported. For an existing file, the mode must be compatible with the file type. (See "File Input/Output" in Section IV of the MPM Reference Guide.) The mode must be compatible with any control arguments given in the attach description.

An existing file is not truncated at open if its safety switch is on and its bit count is nonzero.

If the opening is for input only ~~and without the -share control argument~~, only read access is required on the file. In all other cases, rw access is required on the file.

Position Operation

An additional type of positioning is available with unstructured and blocked files that are open for input, input_output, or update. When the type

(under section titled Multiple Openings)

4. Openings with the -share control argument. ~~(This applies to direct_input, direct_update, and direct_output with the -extend control argument only).~~ Any number of openings of this type are allowed. When a process performs an ~~operation~~ update on the file, the file is locked. Other processes attempting an operation while the file is locked will wait up to the limit specified by wtime in the -share control argument. If the operation is not carried out because of the wtime limit, the code error_table_$file_busy is returned.

750508

Multics Change Request

MCR 1616
Page 1 of 1

TITLE: "set_file_lock" and "set_wait_time" orders for indexed files

AUTHOR: M. Asherman

STATUS	DATE
Written	01/28/76
Status	A 02/10/76
Expires	08/10/76

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

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

DOCUMENTATION CHANGES

Document	Specify One or More
MPM (Vol, Sect.)	vfile_
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 two new control orders for indexed files attached with the -share option. "set_file_lock" allows a file's lock to be set or cleared explicitly. "set_wait_time" permits the user to change the time limit his process will wait to perform an operation when the file is locked by another process.

REASONS:

The alternative method of achieving the same ends requires remembering the current file position, closing, detaching, re-attaching, opening, and re-positioning.

IMPLICATIONS:

User convenience.

DETAILED PROPOSAL:

See attached documentation.

Control Operation

set_file_lock

The order "set_file_lock" is accepted when the I/O switch is open for output or update and attached to an indexed file with the -share control argument. For this order, the info_ptr argument must point to a structure of the following form:

```
dcl  set_lock_flag bit(1) aligned based(info_ptr);
```

This operation causes the file to be locked (if possible within the wait-time limit) or unlocked, depending on the user's setting info_ptr→set_lock_flag to "1"b or "0"b, respectively.

The possible error codes are those returned by set_lock_$lock and set_lock_$unlock, excepting the code error_table_$invalid_lock_reset, which is not treated as an error.

When the file is locked, other processes are prevented from performing operations on the file until its lock is cleared, or the locking process ceases to exist.

Control Operation

set_wait_time

The order "set_wait_time" is accepted when the I/O switch is open and attached to an indexed file with -share control argument. For this order the info_ptr argument must point to a structure of the following form:

```
dcl new_wait_time fixed based(info_ptr);
```

This operation specifies a limit on the time that the user's process will wait to perform an operation when the file is locked by another process. The interpretation of new_wait_time is the same as that described earlier for the optional wtime argument used with the -share attach option.

Ver. 3
741022

MULTICS CHANGE REQUEST

MCR 1617

TITLE: Change indent to work on cds segments.

AUTHOR: Steve Webber

STATUS	DATE
Written	01/29/76
Status	A 02/10/76
Expires	07/29/76

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

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

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

OBJECTIONS/COMMENTS: This will not be documented in the MPM until cds segments are promoted to the MPM.

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

SUMMARY:

Change the indent command to expect the suffix "cds" as well as the suffix "pl1".

REASONS:

The new language type, cds, has a syntax identical to that of PL/I and hence it would be appropriate for indent to work on these programs as well.

IMPLICATIONS:

Since indent is an MPM command, we will have to promote cds to MPM level if we want the MPM documentation to be complete with respect to which segments are handled by indent.

Name: indent, ind

The indent command improves the readability of a PL/I or CDS. source segment by indenting it according to a set of standard conventions described below.

Usage

```
indent oldpath -newpath- -control_args-
```

where:

1. oldpath — is the pathname of the input PL/I or CDS source segment. If the input segment name does not have a suffix of pl1, the suffix is assumed.
   (or cds   of pl1

2. newpath — is the pathname of the output PL/I or CDS source segment. If the ouput segment name does not have a suffix of pl1, the suffix is assumed. If this argument is omitted, newpath is assumed to be the same as oldpath, and the indented copy of the program replaces the original copy.
   or cds
   pl1

3. control_args — can be any of the following:

   -brief, -bf — suppress warning comments on illegal or non-PL/I characters found outside of a string or comment. (Such characters are never removed.)

   -lmargin XX, -lm XX — set the left margin (indentation for normal program statements) to XX. If this argument is omitted, the default for XX is 11.

   -comment YY, -cm YY — set the comment column to YY. Comments are lined up in this column unless they occur in the beginning of a line or are preceded by a blank line. If this argument is omitted, the default for YY is 61.

   -indent ZZ, -in ZZ — set indentation for each level to ZZ. Each do, begin, proc, and procedure statement causes an additional ZZ spaces of indentation until the matching end statement is encountered. If this argument is omitted, the default for ZZ is 5.

Conventions

Declaration statements are indented five spaces for dcl declarations and ten for declare declarations. Identifiers appearing on different lines of the same declare statement are lined up under the first identifier on the first line of the statement. Structure declarations are indented according to level number; after level two, each additional level is indented two additional spaces.

An additional level of indentation is also provided for the then-clause of an if-statement; else clauses are lined up with the corresponding if. Statements that continue over more than one line have an additional five spaces of indentation for the second and all succeeding lines.

Multiple spaces are replaced by a single space, except inside of strings or for nonleading spaces and tabs in comments. Trailing spaces and tabs are removed from all lines. The indent command inserts spaces before left parentheses, after commas, and around the constructs =, ->, <=, >=, and ^=. Spaces are deleted if they are found after a left parenthesis or before a right parenthesis. Tabs are used wherever possible to conserve storage in the output segment.

The indent command counts parentheses and expects them to balance at every semicolon. If parentheses do not balance at a semicolon, or if the input segment ends in a string or comment, indent prints a warning message. Language keywords (do, begin, end, etc.) are recognized only at parenthesis level zero, and most keywords are recognized only if they appear to begin a statement.

Restrictions

Lines longer than 350 characters are split, since they overflow indent´s buffer size. This is the only case in which indent splits a line.

Labelled end statements do not close multiple open do statements.

The indent command assumes that the identifiers begin, end, procedure, proc, declare, and dcl are reserved words when they appear at the beginning of a statement. If the input contains a statement like:

```
do = do + 1;
```

the indent command interprets it to mean that the statement delimits a do group and does not indent correctly.

Structure level numbers greater than 99 do not indent correctly.

Ver. 3
741022

MULTICS CHANGE REQUEST

MCR 1619

TITLE: Fix bug in set_lock_ which sometimes causes initializer problems.
AUTHOR: Steve Webber

STATUS	DATE
Written	01/29/76
Status	A02/10/76
Expires	07/29/76

Planned for System: not applicable
Fixes Bug Number(s): unreported
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

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)
MOSN (sect)
PLMs (AN#)
Info Segs
Other
None (reason) no change
MPAM (sect)
MSAM (sect)

OBJECTIONS/COMMENTS:

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

SUMMARY:

Fix a bug in set_lock_ which can cause the loss of alarm clock wakeups in the calling process.

REASONS:

Better reliability.

Ver. 4
750508

Multics Change Request

MCR 1621
Page 1 of 1

TITLE: Fix pi bug in mail

AUTHOR: S. Herbst

STATUS	DATE
Written	February 2, 19
Status	A 02/10/76
Expires	08/10/76

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

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

DOCUMENTATION CHANGES

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

Objections/Comments:

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

SUMMARY:

Fix bug causing mail's program_interrupt handler to work incorrectly

REASON:

When reading mail, if program_interrupt is used before answering the query "Delete?", mail tries to delete a nonexistent message from the mailbox and gets an error.

Ver. 4
750508

Multics Change Request

MCR 1623
Page 1 of 1

TITLE: Fix bug in dmp355
AUTHOR: M. Grady

STATUS	DATE
Written	2 February 76
Status	A 02/10/76
Expires	08/10/76

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

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

DOCUMENTATION CHANGES

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

Objections/Comments:

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

SUMMARY:

Fix a bug in trace data copy loop in dmp355.

REASONS:

If more than 6 words of trace data are included in a trace message, dmp355 will fault and not print the dump.

DETAILED PROPOSAL:

Increase limit from 6 to 20 and add warning message if limit exceeded.

Ver. 4
750508

Multics Change Request

MCR 1624
Page 1 of 1

TITLE: Incorporate USGS Improvements in MCS
AUTHOR: M. Grady

STATUS	DATE
Written	2 February 76
Status	A 02/10/76
Expires	08/10/76

-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
X	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:

Meter command is required

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

SUMMARY:

Add performance improvements and meters developed during USGS benchmark, to standard system:

1) New scheduler priorities to allow fast response to hsla pre-tally runout interrupts, i.e., full input buffers.

2) Faster LRC checking code in interpreter.

3) Bug fix in scheduler timer management and queue sizes.

4) Fast test for complete sync output messages in interpreter.

5) New meters in dn355 to watch average free space in each 355 and tty_buf.

REASONS:

The changes showed large performance improvements during USGS testing and should be installed.

DETAILED PROPOSAL:

Coded in 355map.

Ver. 3
741022

MULTICS CHANGE REQUEST

MCR 1627

TITLE: Fix g115_ DIM and MCS to run remote printers properly
AUTHOR: Robert S. Coren

STATUS	DATE
Written	02/02/76
Status	A 02/10/76
Expires	08/02/76

Planned for System: MR3.1
Fixes Bug Number(s): not applicable
Documented in MTB: not applicable
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
(☒)tools & 355

DOCUMENTATION CHANGES (specify one or more)
MPM (vol,sect) MPAM (sect)
MOSN (sect) MSAM (sect)
PLMs (AN#)
Info Segs
None (reason) makes g115 interface work properly

OBJECTIONS/COMMENTS:

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

SUMMARY: Fix 355 control_tables and g115 DIM so that initialization sequence completes properly.

REASONS: Prior to BOS 1-47, 355 would crash if Multics sent it only part of a g115 message. Fix for this problem was incorrectly implemented, so that g115 initialization protocol never completed properly. Also, g115 DIM was not changed to retry only that part of the message which didn't get sent the first time.

IMPLICATIONS: Multics support for the g115-type device will finally be real.

NOTE: MCS portions of fix coded in 355map.

Ver. 3
741022

MULTICS CHANGE REQUEST

MCR 1628

TITLE: Fix two bugs in MCS

AUTHOR: Robert S. Coren

STATUS	DATE
Written	02/02/76
Status	A 02/10/76
Expires	08/02/76

Planned for System: MR3.1
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)
MOSN (sect)
PLMs (AN#) AN85
Info Segs
Other

MPAM (sect)
MSAM (sect)

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: 1. Fix bug which sometimes caused a tty channel to get one input message behind.
2. Remove code that tries to implement special LSLA option recognizing dataset leads separately.

REASONS: 1. Bug exists in dia_man which causes "last-buffer-in-chain" flag to be set erroneously when too many input buffers filled.
2. LSLA theoretically can be optioned to recognize carrier detect, data set ready, and clear to send separately; ordinarily it only recognizes carrier detect, which software maps into all three. Option does not work as expected, and current software assumes that once it sees data set ready or clear to send it can expect always to see them; this makes it impossible to dial up a line connected to the relevant LSLA. We propose to ignore the option, and thus ignore data set ready and clear to send.

IMPLICATIONS: Option to allow LSLA to report data set leads separately will not be supported by MCS.

NOTE: Coded in 355map.

Ver. 4
750508

Multics Change Request

MCR 1629
Page 1 of 1

TITLE: Fix bugs in list command

AUTHOR: T. Casey

STATUS	DATE
Written	January 30, 197
Status	A 02/10/76
Expires	08/10/76

-Coded in: ☒ PL/I ☐ AIM ☐ 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 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:

Fix bugs in the printing of information from the VTOC for msf's.

REASON:

Some code that tries to avoid unnecessary VTOC I/O avoids necessary VTOC I/O for multisegment files, printing directory information even when user has asked for VTOC information.

750508

Multics Change Request

MCR 1630
Page 1 of 1

TITLE: Drop reloader references to installation_parms
AUTHOR: A. Kobziar

STATUS	DATE
Written	2 February
Status	A 02/10/76
Expires	08/10/76

-Coded in: ☒ PL/I □ ALM □ other-explain in DETAILED PROPOSAL
-Planned for System MR
-Fixes Bug Number(s) C80564
-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
X	Ring One
	SysDaemon/Admin.
	Runtime
	User Cmmd/Subr.

DOCUMENTATION CHANGES

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

Objections/Comments:

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

SUMMARY:

Drop all reloader references to installation_parms.

REASON:

Reloader sometimes is required to replace installation_parms, causing seg faults on subsequent references.

IMPLICATIONS:

Access classifications will only be printed in octal rather than both octal and ascii, a small loss. Since calls are removed from the reloader, a theoretical speed increase results.