TO: MSPM Distribution
FROM: Karolyn Martin
DATE: July 10, 1969
SUBJECT: Standards for Command Writers, BX.0.01

Section BX.0.01 is being re-issued to reflect a number of policy changes on Multics during the past two years.

1. Restricted EPL is to be used in all command coding.

2. The new file system provides some improved facilities for initiating segments.

3. Rules now exist for deciding what directory to use when creating a segment, depending on the intended use of the segment.

4. Access control settings are standardized for different types of segments.

5. Error handling is no longer done by signalling.

Published: 07/10/69
(Supersedes: BX.0.01, 12/30/65
BX.0.01, 01/06/67)

Identification

Standards for Command Writers

G. Schroeder, K.J. Martin

Purpose

Through the mechanism of the shell any procedure--system library, system commands, user procedures--can be invoked from the console. Although system commands are procedures, they are a special class of procedures in that they are programmed with the console user in mind. They must check carefully for argument validity; they must warn the user of possible misunderstandings; they must be *very* reliable.

Rules

The following is a set of rules considered mandatory for the writing of the system commands.

Multics Standards

1. They must be pure procedures

2. They must return to the calling procedure under all circumstances

3. They must be written in a high-level language in a straight-forward manner. Explicit permission is required to use any language except EPL or PL/1. They must be well commented, including a statement at the top of the source listing giving command purpose, a description of the calling sequence, and name of author and date of original coding and all subsequent modifications.

Efficency

4. All commands written in EPL should use restricted EPL (REPL) whenever possible. This is particularly appropriate in calls to non-user-oriented procedures. In general, however, convenience to users is primary. Efficiency is important, but is secondary to user (not command-writer) convenience.

5. They should avoid calling write-arounds and, rather, should call the desired primitive directly. Write-around procedures are provided for the casual user, not for the command writer.

6. The most common route through a command should be as efficient as possible. More exotic facilities which are inherently expensive should be separated from the simple facilities so that a casual user need not pay for the exotic every time he uses the simple.

Good Citizenship

7. Commands must leave system data bases in a consistent state; e.g., a command which changes the contents of a segment should reset the bit count when finished. Commands should make any period of inconsistency as short as possible. They must also clean up after themselves; e.g., free storage should be released.

8. Commands should, in general, use one of the two standard I/O streams, user_input and user_output. Only special I/O service commands should issue I/O attach or detach calls for these streams.

9. Commands should do thorough validity checking on arguments. Under no circumstances should incorrect use of arguments cause unexpected, random behavior. Binary arguments (value yes or no, on or off) should be explicitly checked for each possible setting unless a default is clearly stated in the documentation (see no. 20).

10. Commands should use system library procedures if one exists which performs a required function. Conversely, if a new function is required which seems generally useful but has not been coded, it should be made a system library procedure. The new procedure should be as general as possible.

11. Commands should initiate the segments they access by a null call name (a facility of the new file system), and should subsequently access those segments via a pointer. Accessing segments by call name is less efficient and is discouraged. However, if a command does find a call name necessary, the segment must be initiated by a unique name rather than by the entry name. In general, segments initiated by a command should be terminated by that command. Exceptions, such as compilers, which wish to leave temporary segments initiated for future use must truncate the segments to zero-length.

12. Any command which creates new segments should put them into the user's current working directory unless the command explicitly makes provision for the user to provide a target directory. (The movebranch and copy commands fall into this latter category; most compilers choose a directory by default, so should choose the working directory.) The aim of this rule is to avoid "messing up" another directory, such as the directory from which a source segment was obtained.

13. The process directory is often used for temporary segments. It should not be used to create segments which will be later moved to the working directory. Commands should clean up after themselves by deleting and terminating process directory segments when they have finished their work. (Note the exceptions in rule 11.) Segments which are no longer needed should be deleted or truncated to zero-length. It is desireable that segments in the process directory should have unique names to avoid name conflicts (see BY.15.01). However, it is also desireable that a user be able to tell these uniquely-named segments apart. The convention to be followed is a name with a unique first component and some identifying second component such as "epl" for temporary segments created by the EPL compiler. Unique names should be qualified in general to avoid confusion.

14. Commands which create new segments must set access control lists according to the conventions enumerated below. If a segment is being replaced instead of being newly created, the command must leave the access control list as it was before the command acted. For instance, a compiler finds that an object segment already exists with effective access RE for this user, with other access for other users. The compiler must obviously add W access to change the segment contents, but should restore the entire access control list to its former value when the compilation is completed. The access to be given to the user creating a segment is:

Segment type	access	ring brackets
directory segment	REWA	48,48,48
object segment segment	RE	v, 48, 48
data segment	RWA	v, 48,48

where V is the current validation level of the user.

In addition, the CACL of every directory should give REWA access to *.sys_control.* and *.Daemon.* with ring brackets of 48, 48, 48. It will not be necessary to put *.sys_control.* on the CACL after the special access control list (SPACL) has been inplemented.

15. All commands must handle their errors as described in BD.8.00. Cryptic comments to users, such as "error 100" are not acceptable. The com_err procedure (BY.11.07) may be used to simplify and standardize user messages. Any pertinent additional information should be included in error comments such as the argument being processed.

User Convenience

16. Commands should allow a variable number of arguments whenever the function of the command makes this meaningful. Most of the file system commands, for instance, should be able to operate on more than one segment or pair of segments.

17. Commands should follow already established conventions, implicit as well as explicit ones. In particular, any use of "*" as a special character should be analogous to its use in the file system commands. Also, a path name as an argument should be interpreted via the setpath (entryarg) procedure and should not generally be restricted

a) to be an entry in a particular directory or
b) to be a full path name (relative to the root).

18. All commands should check whatever system options apply to their actions.

19. When a command performs several related tasks, several entry points (with descriptive names) are preferable to a long argument list with complicated conventions and defaults, For example, it is easier to remember and use a command which pairs segment names and numbers with the two entries:

a) seg_match$name segno
   which prints out the segment name
b) seg_match$segno name
   which prints out the segment number,

than with the one entry

c) seg_match segno name
   which requires that one argument be the null string and the other be specified.

Documentation

20. Every command available on Multics must be documented in the following ways within a reasonable time after its installation.

a) a full MSPM section describing purpose, usage, special considerations in its design, restrictions and implementation.

b) an MSPM abstract in the form described in BS.0

c) an entry in the Multics Condensed Guide

d) a segment on Multics suitable for inclusion in the on-line documentation scheme described in BX.11.02.