Date: 10/15/76

From: Cec Erickson

Subject: Documentation Format Changes

This MTB shows the new format for command and subroutine descriptions that will be used in Multics documents beginning with MR5.0. The "Usage" portion of the descriptions is being changed.

COMMANDS

The usage line will now have braces ({}) around any optional argument rather than hyphens. For example, the usage line for delete_acl will change from:

```
     delete_acl -path- -User_ids- -control_args-
```

to:

```
     delete_acl {path} {User_ids} {-control_args}
```

In addition, the indenting used in the list of argument descriptions following the usage line will no longer vary according to the longest argument. This method not only added a lot of unnecessary white space but also was a big headache for the documentation group. In the new format, the description of each argument will begin on the line immediately beneath the argument itself at character position 13 (i.e., use .in 12). The runoff control words used in this list will be rigidly controlled; that is, although several different combinations of runoff control words can produce the same runout, one sequence of runoff control words has been chosen as the standard. By having a "standard" we will know how the runoff segment is structured and be able to use various macros to locate and possibly change things.

The basic rules for the "standard" are:

--keep the arguments in the same (runout) positions they had in the old format; only the descriptions of the arguments are changing position
--control the positioning of items by changing the undents, i.e., no leading spaces are allowed

The following pages show the "Usage" portion of the delete_acl command in 5.0 format (first the runout, then the runoff).

Usage

```
     delete_acl {path} {User_ids} {-control_args}
```

where:

1. path
   is the pathname of a segment, multisegment file, or directory. If it is -wd, -working_directory, or omitted, the working directory is assumed. If path is omitted, no User_id can be specified. The star convention can be used.

2. User_ids
   are access control names that must be of the form Person_id.Project_id.tag. All ACL entries with matching names are deleted. (For a description of the matching strategy, refer to the set_acl command.) If no User_id is given, the user's Person_id and current Project_id are assumed.

3. control_args
   can be chosen from the following:

   -all, -a
      causes the entire ACL to be deleted with the exception of an entry for *.SysDaemon.*.

   -directory, -dr
      specifies that only directories are affected. The default is segments, multisegment files, and directories.

   -segment, -sm
      specifies that only segments and multisegment files are affected.

   -brief, -bf
      suppresses the message "User name not on ACL."

```
.if 12h "Usage"
     delete_acl {path} {User_ids} {-control_args}
.sp 2
where:
.sp 1
.in 12
.un 12
1.   path
.br
is the pathname of a segment, multisegment file, or directory.
If it is -wd, -working_directory, or omitted, the working
directory is assumed.  If path is omitted, no User_id can be
specified.  The star convention can be used.
.sp 1
.un 12
2.   User_ids
.br
are access control names that must be of the form
Person_id.Project_id.tag.  All ACL entries with matching names
are deleted.  (For a description of the matching strategy, refer
to the set_acl command.)  If no User_id is given, the user's
Person_id and current Project_id are assumed.
.sp 1
.un 12
3.   control_args
.br
can be chosen from the following:
.sp
.un 5
-all, -a
.br
causes the entire ACL to be deleted with the exception of an
entry for *.SysDaemon.*.
.sp
.un 5
-directory, -dr
.br
specifies that only directories are affected.
The default is segments, multisegment files, and directories.
.sp
.un 5
-segment, -sm
.br
specifies that only segments and multisegment files are affected.
.sp
.un 5
-brief, -bf
.br
suppresses the message "User name not on ACL."
.in 0
```

SUBROUTINES

Although the MPM Subroutines will not be revised until after 5.0, new formatting rules have been established, which can be used in other manuals immediately. This new format is similar to the new format being used for commands. That is, the descriptions of items in the "Usage" portion of the subroutine will be beneath the item they describe and begin at character position 13 (i.e., use .in 12). No leading spaces are allowed; control the positioning of text by changing the undents.

Also, the declare and call lines should be done in no adjust (.na) and a translate character should be used between things like "fixed" and "bin" so they will not appear in different lines in the runout.

The following pages show the "Usage" portion of set_lock_$lock in the new format (first the runout, then the runoff).

<u>Usage</u>

```
declare set_lock_$lock entry (bit(36) aligned, fixed bin,
     fixed bin);

call set_lock_$lock (lock_word, wait_time, code);
```

where:

1. lock_word
   is the word to be locked. (Input)

2. wait_time
   indicates the length of real time, in seconds, that the set_lock_$lock entry point should wait for a validly locked lock word to be unlocked before returning unsuccessfully. A value of -1 indicates no time limit. (Input)

3. code
   is a standard status code. (Output) It may be one of the following:

   0
   indicates that the lock word was successfully locked because the lock word was previously unlocked

   error_table_$invalid_lock_reset
   indicates that the lock word was successfully locked, but the lock word previously contained an invalid lock identifier that was overwritten

   error_table_$locked_by_this_process
   indicates that the lock word already contained the lock identifier of the calling process and was not modified

   error_table_$lock_wait_time_exceeded
   indicates that the lock word contained a valid lock identifier of another process and could not be locked in the given time limit

```
.if l2h "Usage"
.in 10
.na
.un 5
declare set_lock_$lock entry (bit(36)!aligned, fixed!bin, fixed!bin);
.sp
.un 5
call set_lock_$lock (lock_word, wait_time, code);
.ad
.in 0
.sp 2
where:
.sp
.in 12
.un
1.   lock_word
.br
is the word to be locked.  (Input)
.sp
.un
2.   wait_time
.br
indicates the length of real time, in seconds, that the
set_lock_$lock entry point should wait for
a validly locked lock word
to be unlocked before returning unsuccessfully.
A value of -1 indicates no time limit.  (Input)
.sp
.un
3.   code
.br
is a standard status code.  (Output)  It may be one
of the following:
.sp
.un 5
0
.br
indicates that the lock word was successfully locked because
the lock word was previously unlocked
.sp
.un 5
error_table_$invalid_lock_reset
.br
indicates that the lock word was successfully locked, but
the lock word previously contained an invalid lock identifier
that was overwritten
.sp
.un 5
error_table_$locked_by_this_process
.br
indicates that the lock word already contained the
lock identifier of the calling process and was
not modified
```

```
.sp
.un 5
error_table_$lock_wait_time_exceeded
.br
indicates that the lock word contained a valid
lock identifier of another process and could not
be locked in the given time limit
```