To: Distribution

From: C. D. Erickson

Date: 03/17/77

Subject: Changes and Additions to help

Introduction

The format and contents of info segments were originally described in MTB-297. That MTB proposed a separate segment for each entry point of a subroutine, ignoring an obvious problem: what to do with names that exceed 32 characters when the info suffix is added. (For command names that are too long (e.g., print_translator_search_rules.info), help should use only the first 27 characters.) For subroutines, having help use only the first 27 characters is no good because the name could very likely NOT be unique.

Propose the following:

1. Document all entry points of a subroutine, alphabetically, in one info segment named <subroutine_name_>.info.

2. Modify the help command to make it easy for users to get information out of a subroutine info segment; e.g., get just one entry point by typing:

   help <subroutine_name_>$<entry_point_name>

In addition, this MTB describes other suggestions for modifying the help command to offer new features and improve existing ones.

The information in this MTB plus suggestions will be the subject of a future design review. Based on the results of that review, help, validate_info_segs, and possibly check_info_segs will be modified.

Changes to General Format

1. Reverse the order of the function and syntax sections. Many users have complained about help saying "1 line titled "<whatever>" follows. More help?" and syntax will be one line more often than function. Also, most users want the syntax line when they invoke help and it seems to make more sense to show that to them first.

2. Precede the first section with two blank lines to match all other section titles.

Changes to Control Arguments

1. Change the abbreviations for the -section and -search control arguments to reflect the new standard approved by MCR8.

   -section, -sc becomes -section, -scn
   -search, -sh becomes -search, -srh

   The -sc and -sh abbreviations will continue to be implemented, but will not be documented.

2. Change -section to accept a substring of the title being sought, rather than requiring the full title. This facilitates finding a section title that includes spaces or an unknown date, and it makes -section more like -search, which also accepts substrings.

3. Change -title to print the titles, then ask for more help about the first section. Currently, -title prints the titles and exits the info segment. The user must be listing the titles to be able to print one or more of the titled sections. The user should not have to invoke help again to do this.

4. Add a new control argument-

   -control_arg name, -ca name

   which prints the description for the named control argument and exits the info segment. The name will NOT include the hyphen (-), which precedes control arguments, to avoid conflict with help control arguments. For example,

   help help -ca header

   would describe the -header control argument of help.

5. Add a new control argument -

   -brief

   which prints the "Syntax:" section and lists the control arguments and their short forms (in a form similar to the pocket guide), then exits the info segment. For info segments that do not have a "Syntax:" section, help prints:

   help: No Syntax section for <name>.

Changes_To_Answers_For_help_Questions

1. Change the abbreviations for the search and section answers to agree with their corresponding control arguments; search is abbreviated as srh, section as scn. The old abbreviations (sh and sc) will be implemented for compatibility, but will not be documented.

2. Change the title answer to print all titles from the current position to the bottom of the info segment and then repeat the question which was previously answered by title, rather than exiting the info segment. Then the user can print one of the titled sections without having to reinvoke help. The title answer will not affect the current position.

3. Change the title answer to accept a -top control argument which lists all titles from the top down, rather than those from the current position down. The question at which title -top was answered is then repeated and the current location remains unchanged.

4. Change the section answer to accept a substring of the title, rather than requiring the complete title. Searching from the current position, the first section whose title contains the substring is printed. This facilitates finding a section with a long title, or with a title containing an unknown date (e.g., a title in a changes info segment), etc.

5. When no substring is given in a section answer, change section to find the next section that matches the substring given in the most recent section answer. This facilitates searching for a section when the first match wasn't the desired section, and it makes the operation of the section answer similar to that of the search answer. If the section answer or -section control argument was not previously used, print an error and repeat the question that was answered by section. Currently section with no title skips to the next section.

6. Change the section answer to accept a -top control argument, which searches for a matching section title starting from the top of the info segment, rather than from the current position. This allows the user to reposition to a section that may have been skipped, perhaps after he has seen its section title in a "title -top" answer.

7. Change the rest answer to accept a -section, -scn control argument, which causes the rest of the current section to be printed. Often the user wants the rest of the control arguments without the "Notes, Examples:", etc.

8. Change the skip answer to accept a -section, (or -scn) control argument, which skips to the next section. This replaces the function of the section answer with no arguments, which was changed in item 5 above.

9. When there is more than one text block in a section, do not repeat the section title when asking if the user wants more help (i.e., have help say "N lines follow. More help?" rather than "N lines titled 'Crud' follow. More help?").

10. Have help space down one line before printing the next block of text (after the user gives answer). This way the number of lines printed matches the number of lines in the info segment (eliminating several trouble reports). Also, it looks better.

Provide_Info_Segments_For_Subroutines

1. A subroutine info segment will consist of a "common" part followed by separate parts for each of the entry points. Each one of these parts is a logical info segment. The common info segment consists of the date and name header, brief functional statement, any notes that apply to all the entry points, and, as the last section, an alphabetical list of all the entry points. Each entry point info segment consists of a date and name header, syntax section (i.e., call and declare lines), and, in rare cases, notes that apply to only this entry point.

   Thus, the segment named <subroutine_name_>.info would have the following format:

```
.ll 79
.na
.tr !
<date>  <subroutine_name_>
.sp
Function:!!<short phrase>
.sp 2
Notes for <subroutine_name_>:
.br
<anything applying to all entry points of the subroutine>
.sp 2
Entry points in <subroutine_name_>:
<help automatically generates the list of entry points>
.sp 2
Entry!!<date>!!<subroutine_name_>$<entry_point_name>:
.sp
Syntax:
.in 3
.un
call subroutine_$entry (...);
.sp
.un
dcl subroutine_$entry entry (char(*)!aligned,
...,fixed!bin(35));
.in 0
.sp 2
Notes:
.br
<special instructions, name of include segments for
structure declarations, etc. for this entry point only>
.sp 2
Entry!!<date>!!<subroutine_name_>$<entry_point_name>:
    .
    .
    .
    .
```

2. The "Entry..." titles divide the segment into several logical info segments. The "Entry" part of these titles is not printed so each entry point title looks like the header of a separate info segment (i.e., <date> <name>). The first logical info segment (the common portion) may require titles in place of or in addition to "Notes for..." to describe general use of the subroutine (e.g., ioa_ needs a "Control strings:" title). These titles must be placed before the "Entry points..." title.

   The rest of the segment consists of other logical info segments that describe one of the subroutine entry points. Without this division into logical info segments, info segments such as hcs_.info would be too long and would have too many titles (not to mention duplicate titles) to be handled easily by the help mechanisms. Imagine the output of

```
help hcs_ -title
```

   if it weren't divided into logical info segments.

   The following guidelines apply to each "Syntax:" section:

   Call line(s) should contain the "standard" value of items if standard values exist. These are put in braces immediately following the item (e.g., ref_name{""}, seg_sw{0}, ...).

   Declare line(s) should use ! as a translate character between things like "fixed" and "bin" so they will not end up on different lines.

3. Add a new control argument to select the info segment for <subroutine_name_>$<subroutine_name_>-

```
-entry_point, -ep
```

   For example, the user could type either of the following to get the logical info segment for term_$term_

```
help term_ -ep
help term_$term_
```

   This control argument is more obviously needed when the user wants an entry point with a longer name, for example convert_date_to_binary_$convert_date_to_binary_.

   Remember that typing "help <subroutine_name_>" such as

```
help term_
```

   selects the common info segment.

4. Change help to generate a list of entry points from the "Entry..." dividers when the user prints the "Entry points..." section of the common info segment. These are printed in alphabetical order 20 entry points to the text block, with the question

NN more entry points follow. More help?

following each text block. The user can give the following answers:

yes the next text block of entry points is listed.

rest the rest of the entry points are given. Since "Entry points..." is the last section of the common info segment, nothing else is given.

rest -scn
same as "rest" because "Entry points..." is the last section.

skip the next text block of entry points is skipped, and the user is asked the question again.

skip -ep
skips to the logical info segment of the first entry point.

skip -scn
remaining entry points are not listed, and the user is asked if he wants to see the logical info segment of the first entry point (see item 5 below).

no the info segment is exited.

quit help is exited.

title
there are no more titles in the common info segment since the "Entry points..." section is last so the user is told this and asked the question again.

title -top
the titles in the common info segment are listed and the user is asked the question again.

section {name}
the user is told there are no more sections in this logical info segment, and the question is asked again.

section {name} -top
the desired common info segment section is found and printed.

ep {name}
the logical info segment for the named entry point is entered, and its first section (Syntax) is printed. If name is omitted, then the logical info segment for <current_subroutine_name_>$<current_subroutine_name_> is entered.

The name string may be either:
<current_subroutine_name_>$<entry_point_name>
or
<entry_point_name>

5. When the end of the common info segment is reached, help prints

```
Entry point <subroutine_name_>$<first_entry_point_name>.
More help?
```

The following answers may be given:

yes the logical info segment for the first entry point is entered, and its first section is printed.

rest the logical info segment for the first entry point is entered and all sections are printed without further questions.

rest -scn
the entire first section (i.e., "Syntax:") of the logical info segment for the first entry point is printed.

skip skips next text block.

skip -ep
skips to the next logical info segment.

skip -scn
skips to next section, if one exists; if there are no other sections, this is the same as "skip -ep".

no
quit
title
title -top
section {name}
section {name} -top
ep {name}

} same as item 4 above

6. Each logical info segment includes all sections between its bounding "Entry..." dividers, plus all sections of the common info segment that apply to all entry points (those after the "Function:" section, and before the first "Entry..." divider). The help command performs this concatenation automatically, rather than having this information duplicated in each entry point's logical info segment.

7. When the end of an entry point logical info segment is reached, help prints

```
Entry point <subroutine_name_>$<next_entry_point_name>.
More help?
```

The same answers listed in item 5 above may be given (merely substitute "next entry point" for "first entry point" in the explanations).

The following pages contain the revised help command description for the MPM Commands.

Name: help

The help command assists users in obtaining online information about the Multics system. This online information, in the form of documents called info segments, consists of command descriptions and subroutine descriptions, plus miscellaneous information about system status and system changes.

Info segments can be manipulated from command level by using control arguments; from within, by using info segment requests. Control arguments and requests are described below under "Usage" and "Requests," respectively.

Format of Info Segments

All info segments have a heading line consisting of a brief title and the date the segment was last modified. For command and subroutine descriptions, the program names are used for the title.

All info segments are divided into one or more sections, each of which may be composed of one or more text blocks. A text block may be a paragraph, two short paragraphs, a list of items, a partial list; i.e., it is a logical "block" of information. A section consists of one or more text blocks preceded by a section title. Section titles are short -- usually one or two words -- and are always followed by a colon (:).

For commands, section titles in their standard order are:

<date> <command name>, <short name>
header line.

Syntax:
how to invoke the program.

Function:
description of the command.

Arguments:
description of each argument.

Control Arguments:
description of each control argument.

Notes:
comments, clarifications, or any special case information.

Examples:
sample invocations of the program.

For subroutines, section titles in their standard order are:

<date> <subroutine_name>
header line.

Function:
brief description of what the subroutine does.

Notes for <subroutine_name>:
explanation of those items that apply to the entire subroutine.

Entry points in <subroutine_name>:
list of entry points.

<date> <subroutine_name_$entry_point_name>
header line for first entry point.

Syntax:
call and declare lines for first entry point.

Notes:
explanation of any items that apply to this entry point only.

<date> <subroutine_name_$entry_point_name>
header line for next entry point.

.
.
.
.

An info segment may contain section titles other than those listed above. A user can issue the help command with the -title control argument to see all the section titles in a particular info segment.

For a description on how to write info segments, refer to the MPM Subsystem Writers' Guide.

Usage

```
help {names} {-control_args}
```

where:

1. names

   specify the info segments that the user wishes to read. If name$_i$ contains a greater-than (>) or less-than (<) character, it is the pathname of an info segment. Otherwise, the command searches for name$_i$ first in the site-dependent information directory, then in the system information directory. If name$_i$ does not have the info suffix, one is assumed.

   For subroutine info segments, name$_i$ must be of the form:

   ```
   subroutine_name_$entry_point_name
   ```

   For using help with the star convention, see "Notes" below.

2. control_args

   may be chosen from the following:

   -header, -he

   prints only an expanded heading line (consisting of a title, segment modification date, and pathname) of the info segment. (This control argument may be used to obtain the pathname of an info segment so it can be printed with the dprint command.)

   -title

   prints the heading line and section titles of the info segment and then goes to the first section and asks if the user wants more help. If the user asks for more help, the command begins printing the first section. (See "Requests" below for information about the query and possible user responses to it.)

-section STR, -scn STR
searches the info segment for a section title containing the STR string, prints the section title and asks, "More help?". If the user asks for more help, the command begins printing the section; printing continues only to the end of the first text block of that section. Then the user is asked if he wants more help.

If the STR string is more than one word, it must be enclosed in quotes. If the -section and -search control arguments are given, the -section search is done first.

-search STRs, -srh STRs
searches the info segment for a text block that contains the specified character string(s). The character strings may be anywhere within a text block and in any order but only within one text block. The command prints the first text block that contains the specified character strings and then asks the user if he wants more help. Everything in the command line after the -search control argument is taken as part of the character string, so the -search control argument must be the last control argument given.

If no text block containing the requested strings is found, the message:

search failed: search STRs

is returned and the command goes to the first section and asks if the user wants more help.

-control_arg STR, -ca STR
searches the info segment for the control argument named STR and prints its description. The STR string must be either the long or short name of a control argument without the hyphen.

-brief, -bf
prints a brief summary of a command or subroutine info segment, consisting of the syntax section and, for commands, a list of control arguments and their short forms. For info segments that do not have a syntax section, help prints:

help: No Syntax section for <name>.

-pathname path, -pn path
specifies the info segment path that the user wishes to read. The command does not search in the site-dependent or system information directories. This control argument is useful for specifying the name of an info segment in the user's working directory. If path does not have the info suffix, one is assumed. For use with the star convention, refer to "Notes" below.

For subroutine info segments, path must be of the form:
subroutine_name_$entry_point_name

Requests

After printing a text block of an info segment, the help command asks the user if he wants more help before it prints the next text block. The query is of the form:

N lines titled "XXX" follow. More help?

where N is the number of lines and XXX is the title of the next section or:

N lines follow. More help?

if the text block to be printed does not begin a new section. The user may respond to either query with any of the requests listed below.

yes
prints the next text block of information and then asks the user is he wants more help.

no
exits from the current info segment. If no other info segment remains, this request is the same as the quit request.

quit
terminates the command, returning user to command level.

rest {-scn}
prints the remaining text of either the current info segment ("rest") or the current section ("rest -scn").

skip {-scn} {-ep}
skips the text block referenced by the query (or remainder of the current section if "skip -scn" is typed or entry point if "skip -ep" is typed) and asks the user if he wants to see the following text block (or section or entry point). The -scn and -ep control arguments to the skip request conflict -- only one may be given.

title {-top}
prints section titles. If the -top argument is given, help prints all section titles. If the title request is given without any arguments, help prints the section titles of all sections remaining in the info segment.

This request does not change the user's current location within the info segment.

section {STR} {-top}, scn {STR} {-top}
searches the info segment for a section title containing the STR string, prints the title, and asks the user if he wants more help. If the -top argument is given, help searches from the top of the info segment and prints the first section title containing STR. If the user asks for more help, the command starts printing from that section. Printing continues only to the end of the first text block of that section, then the user is asked if he wants more help. If no section title containing STR is found, the message:

Section containing "STR" not found.

is returned and the query is repeated. If no STR argument is given in the request, help uses the STR given in the last search request; if no other search request has been given, help uses the STR given in the -search control argument. If no strings are specified and a section request or the -section control argument has not been issued previously, the help command returns an error message and the query is repeated.

search {STRs}, srh {STRs}
: searches the info segment for a text block that contains the specified character strings. The character strings may be anywhere and in any order but only within one text block. Printing starts at the beginning of the text block and terminates at the end. Then the user is asked if he wants more help. If no text block containing the requested strings is found, the message:

```
search failed: search STRs
```

is returned and the query is repeated. If no STR argument is given in the request, help uses the STR given in the last search request; if no other search request has been given, help uses the STR given in the -search control argument. If no STRs are specified and a search request or the -search control argument has not been issued previously, the help command returns an error message and the query is repeated.

entry_point {STR}, ep {STR}
: begins printing the description of the STR entry point where STR must be one of the following:

```
entry_point_name
subroutine_name_$entry_point_name
```

If STR is not given, the help command begins printing the subroutine_name_$subroutine_name_ entry point description.

Notes

If the user issues a quit signal, printing of the current text block is aborted. The program_interrupt command can be used to reenter the help command; at reentry the user is queried about the next text block.

The star convention is not allowed with subroutine info segments. For those invocations in which the star convention is allowed, the following rules apply:

1. The info segments whose entryname matches the star name are alphabetized within the directory and scanned in that order.

2. Control arguments apply to all matching info segments; requests, to the current info segment. When control argument are given, help scans each info segment until the desired item is found, then questions the user as usual.

Typing the command "help" or "help help" prints information about the help command.

For a list of info segments in the site-dependent information directory, type:

```
help >doc>ils>** -he
```

For a list of info segments in the system information directory, type:

```
help >doc>info>** -he
```

Examples

In the examples below, lines (or partial lines) typed by the user are preceded by an exclamation mark (!).

```
! help copy -brief

Syntax: cp path1A {path2A ... path1N path2N} {-control_args}
-name, -nm          -all, -a
-acl                -brief, -bf
<ready>


! help hcs_ -brief
help: No Syntax section for hcs_.
<ready>
```

```
!  help convert_authorization_
   (6 lines follow; 15 lines before first entry point.)

   08/03/76  convert_authorization_

   Function:  converts an AIM authorization back and forth
   between its binary and character-string representation.

   9 lines titled "Entry points in convert_authorization_"
   follow.  More help? ! yes

   Entry points in convert_authorization_:
   convert_authorization_$decode
   convert_authorization_$encode
   convert_authorization_$from_string
   convert_authorization_$minimum
   convert_authorization_$to_string
   convert_authorization_$to_string_short

   Entry point convert_authorization_$decode.  More help?  ! yes
   (8 lines follow; 17 lines for entry point.)

   08/03/76  convert_authorization_$decode

   Syntax:
   call convert_authorization_$decode (....
            .
            .
            .
            .



!  help hcs_$initiate
   (10 lines follow; 132 lines for entry point.)

   08/10/76  hcs_$initiate

   Syntax:
   call hcs_$initiate (dir_name, entryname, ref_name{""},
      seg_sw{0} copy_ctl_sw{1}, seg_ptr, code);

   dcl hcs_$initiate entry (char(*), char(*), char(*),
      fixed bin(1), fixed bin(2), ptr, fixed bin(35));

   N lines titled "Notes for hcs_" follow. More help? ! skip -scn
```

```
   N lines titled "Entry points in hcs_" follow. More help?
   ! ep initiate_count
   (10 lines follow; 132 lines for entry point.)

   08/10/76  hcs_$initiate_count

   Syntax:
   call hcs_$initiate_count (....
                .
                .
                .
                .



!  help term_
   (6 lines follow; 20 lines before first entry point.)

   11/04/76  term_

   Function: removes reference names from segments; removes
   segments from address space.

   6 lines titled "Notes for term_" follow.  More help?  ! yes

   Notes for term_:
   term_ performs the same function as some hcs_ entry points,
   but in addition, it unsnaps links to the terminated reference
   names.  Use of term_ is preferred to the hcs_ entry points.

   8 lines titled "Entry points in term_" follow.  More help?
   ! yes

   Entry points in term_:
   term_$refname
   term_$seg_ptr
   term_$single_refname
   term_$term_
   term_$unsnap

   Entry point term_$refname.  More help?  ! yes
   (8 lines follow; 14 lines for entry point.)

   11/04/76  term_$refname:

   Syntax:
   call term_$refname (ref_name, code);

   dcl term_$refname entry (char(*) aligned, fixed bin(35));
```

```
6 lines titled "Notes for term_" follow.  More help?  ! yes

Notes for term_:
term_ performs the same function as some hcs_ entry points,
but in addition, it unsnaps links to the terminated reference
names.  Use of term_ is preferred to the hcs_ entry points.

Entry point term_$seg_ptr.  More help?  ! ep

Entry point term_$term_.  More help?  ! yes
(8 lines follow; 14 lines for entry point.)

11/04/76  term_$term_

Syntax:
call term_ (dir_path, entryname, code);

dcl term_ entry (char(*) aligned, fixed bin(35));

6 lines titled "Notes for term_" follow.  More help?  ! quit
```