see BS.22.02

Published: 10/29/69

Identification

Multics Segment List Utility Procedure
msl_util
Edwin W. Meyer, Jr.

Purpose

The procedure msl_util contains routines common to several different modules that reference and modify a Multics Segment List (MSL).

$get msl

This entry attempts to initiate the MSL segment specified by "dir_path" and "msl_name" and return its base pointer in "msl_pr". If the segment can not be found and "init_create" = 0, "msl_pr" is returned as null. If "init_create" = 1 and if the MSL could not be found, the message "creating virgin msl" is sent to user_output and an empty MSL is created and its base pointer returned in "msl_pr".

```
     call msl_util$get_msl (dir_path, msl_name, init_create,
                            msl_pr);
```

1) dir_path(char(*))	pathname of directory containing the MSL
2) msl_name(char(*))	entry name of the MSL
3) init_create(fixed bin(1))	initiate/create switch = 0 for null ptr
4) msl_pr(ptr)	pointer to base of MSL segment (output)

$name entry

The MSL with base pointer "msl_pr" is searched for the entry name and operation continues according to "rcd_sw".

```
     call msl_util$name_entry (bpr, rcd_sw, name, node);
```

1) bpr(ptr)	MSL base pointer
2) rcd_sw(fixed bin(2))	entry read/create/delete switch =1 for add
3) name(char(*))	entry name to be searched [illegible]
4) node(fixed bin(34))	node of entry list (Output)

rcd_sw = 0	The node of entry list for name is returned if the entry is found. Else node is set to null (0).
rcd_sw = 1	Same as above, except that if an entry for name is not found, one will be created with an empty list except for the name item.
rcd_sw = 2	The entry for name is deleted from the MSL, if found; and also from the association lists of entries which reference this entry.

$get item

The data pointed to by element "item_index" of entry list "entry_node" is returned in "item_pr" and item_len". In the case of character string data items, "item_pr" points to the string base and "item_len" is the character count. The only exception occurs if "item_index" = 1 (type_code item, a binary block). In this case, a one or two character type abbreviation corresponding to the binary type_code is returned in "item_pr" and "item_len".

```
      call msl_util$get_item (bpr, entry_node, item_index,
                              item_pr, item_len);
```

1) bpr(ptr)	MSL base pointer
2) entry_node(fixed bin(34))	node of entry item list
3) item_index(fixed bin(17))	index to entry item
4) item_pr(ptr)	ptr to item data (output)
5) item_len(fixed bin(17))	length of item (output)

$set item

A character string data block containing "item_string" is created and its node is inserted into the "item_index"th element of entry list "entry_node". If item_index = 1, the binary type_code corresponding to the supplied type abbreviation is inserted.

```
     call msl_util$set_item (bpr, entry_node, item_index,
                             item_string);
```

1) bpr(ptr)	MSL base pointer
2) entry_node(fixed bin(34))	node of entry item list
→ 3) item_index(fixed bin(17))	index to entry item
4) item_string(char(*))	character string item to be inserted into item list (use substr to pass chars w/o blanks)

$assoc_list

This entry operates on the superior/inferior list of entry item list "entry_node" as follows:

si_ad = 0	add to superior list
si_ad = 1	add to inferior list

The MSL is searched for an entry for name. If it is not found, one is created, empty except for the name item. Next the superior/inferior list of "entry_node" is searched for the entry name. If it is found, no action takes place, as name is already part of the list. Otherwise, an associative block is created and threaded alphabetically into the superior/inferior list of "entry_node" and also threaded into the inferior/superior list of the entry for name. (Note the converse.)

si_ad = 4	delete from superior list
si_ad = 5	delete from inferior list

The superior/inferior list of "entry_node" is searched for an associative block for name. If one is found, it is deleted from the list and also from the inferior/superior list of the entry for name.

```
     call msl__util$assoc_list (bpr, entry_node, si_ad,
                                name);
```

1) bpr(ptr)	MSL base pointer
2) entry_node(fixed bin(34))	node of entry item list
3) si_ad(fixed bin(3))	superior/inferior and add/delete switch
4) name(char(*))	name of entry to be added to or deleted from the superior/inferior list of "entry_node"

$get_assoc_name

msl_util$get_assoc_name returns the name of the superior/inferior entry of the association block at "blk_node" as pointer "item_pr" to the base of the name character string and character count "item_len".

```
     call msl_util$get_assoc_name(bpr, blk_node, si_sw,
                                  item_pr, item_len);
```

1) bpr(ptr)	MSL base pointer
2) blk_node(fixed bin(34))	associative block node
3) si_sw(fixed bin(1))	superior/inferior switch 0 = superior 1 = inferior
4) item_pr(ptr)	pointer to superior/inferior entry name character string (output)
5) item_len(fixed bin(17))	character count (output)

$set_path

A character string block containing "path_string" is created and its node is inserted into the "path_index"th element of the path_list item of entry list "entry_node".

```
     call msl_util$set_path(bpr, entry_node, path_index,
                            path_string);
```

1) bpr(ptr)	MSL base pointer
2) entry_node(fixed bin(34))	node of entry item list
3) path_index(fixed bin(17))	index to a path_list element of "entry_node"
4) path_string(char(*))	character string to be added to the "path_index"th element of the path_list of "entry_node"

$get_path

The character string pathname in the "path_index"th element of the path_list item of entry list "entry_node" is returned as a pointer "path_pr" to the base of the string and character count "path_len".

```
     call msl_util$get_path(bpr, entry_node, path_index,
                             path_pr, path_len);
```

1) bpr(ptr)	MSL base pointer
2) entry_node(fixed bin(34))	node of entry item list
3) path_index(fixed bin(17))	index to a path_list element of "entry_node"
4) path_pr(ptr)	pointer to character string pathname (output)
5) path_len(fixed bin(17))	character count (output)