Published: 10/25/68
(Supersedes: BD.3.02, 05/05/67)

Identification

Segment Management Primitives
John W. Gintell

Purpose

The BFS provides segment management primitives available both to system modules such as the linker and to the general user. The primitives are used to establish segment name-segment number correspondences and to perform various tasks necessary to utilize a segmented system.

Primitives

The following is a list of the new segment namagement primitives of the BFS. Although they are available to the general user as well as to system programs, outside of ring 0 they must be called via hcs_. Following this list is a detailed discussion of each primitive. (Note that declarations of all arguments follow the separate discussions.)

1. initiate (hcs_$initiate)
2. seg_man$get_segment (hcs_$get_segment)
3. seg_man$get_rel_segment (hcs_$get_rel_segment)
4. seg_man$get_seg_ptr (hcs_$get_seg_ptr)
5. seg_man$terminate (hcs_$terminate)
6. seg_man$make_seg (hcs_$make_seg)
7. seg_man$set_wdir (hcs_$set_wdir)
8. seg_man$get_wdir (hcs_$get_wdir)
9. seg_man$get_path_name (hcs_$get_path_name)
10. seg_man$get_name (hcs_$get_name)
11. seg_man$get_seg_status (hcs_$get_seg_status)

1. initiate

The following call is provided to initiate a segment and to obtain a pointer to the initiated segment.

call initiate (dirname, name, callname, copy, segptr);

dirname is the pathname of the directory in which the segment named name is to be found. callname is the reference name of the segment. copy is used as follows:

If copy = 0 then a copy of the segment is initiated if the copy bit in the branch is on; the original segment is initiated if the copy bit in the branch is off.

If copy = 1 then the original segment is initiated regardless of the value of the copy bit in the branch.

If copy = 2 then a copy of the original segment is initiated regardless of the value of the copy bit in the branch.

segptr is a pointer to the initiated segment and is set to null if the segment could not be initiated.

By convention, if dirname is the null character string, initiate returns a non-null segptr only if this segment was already initiated by the reference name callname in the ring corresponding to the validation level. (This convention is established to allow simple implementation of get_seg_ptr.)

Implementation of initiate is as follows:

1. A name unique to the calling ring is constructed by concatenating the validation ring number with the actual name: nn_ _ callname, where nn is the ascii representation of the decimal value of the validation ring number.

2. The pathname is checked. If it is not a null string go to step 3. If it is null the KST is searched for an entry with this name by calling sum$nsrchkst. If found, segptr is constructed from the returned segment number and a return is made. If not found a null pointer is returned in segptr.

3. Findbranch is called with the pathname and name to see if they can be found in the specified directory. If not found, a null pointer is constructed and a return is made.

4. Effmode is called to obtain the effective mode of the branch and the mode attributes in the branch are checked for at least some permission if the initiated segment is not a directory and for execute permission if it is a directory segment. A null segptr is returned if these checks fail.

5. Makeknown is called at makeknown$add_callname to make the segment known and to add the name to the KST. If makeknown returns with an error code (which may be not fatal) segptr as returned by makeknown is returned. segptr will be null if the error was fatal and will point to the known segment if it is not fatal.

6. The copy switch in the branch is now checked in conjunction with the copy option and if a copy is to be made:

   a. seg_man$make_seg is used to generate a segment of the same size as the original segment and with mode "rewa".

   b. move is used to copy the contents of the segment into the generated segment after checking that there was read permission.

   c. makeunknown is used to make the original segment unknown.

7. The branch is unlocked, and a return is made with segptr pointing to the initiated segment or set to null if the segment was not initiated.

2. seg_man$get_segment

The following call is provided to obtain a pointer to the segment corresponding to a given call name.

```
call seg_man$get_segment (callerptr, callname,
     copy, segptr)
```

Initiate is called with a null pathname to find whether a segment with a reference name callname has already been initiated. Initiate returns seqptr null if callname is not initiated and seqptr pointing to the base of the segment if it has been initiated.

If seqptr is returned not null by initiate then get_segment can return. If it is null get_segment must proceed to search, in turn, each of the directories prescribed by the Standard Search rules, beginning with the directory corresponding to the caller segment. callerptr is a pointer to a segment whose immediately superior directory is to be checked for a segment by the name callname before the normal search is performed. The segment number contained in callerptr is used to index into the KST entry from which is taken the segment number of the immediately superior directory from whose entry is taken the pathname. Initiate is then called to attempt to initiate the segment named callname in the directory whose pathname was just obtained. The argument copy is passed on to initiate to indicate whether a copy is desired.

If the call to initiate returns a null seqptr indicating that the segment was not initiated then sequential attempts are made to initiate the segment in the following directories (according to the current Search Rules):

the working directory (if there is one)

system_library

system_library_1

system_library_2

system_library_3

system_library_4

system_library_5

the process directory

If the segment cannot be initiated seqptr is returned with null value.

3. seg_man$get_rel_segment

This call is provided to obtain a pointer to a segment whose reference name is composed of the reference name of another initiated segment concatenated with some more characters. (E.g., an internal portion of the linker wants the linkage segment corresponding to some segment whose number is known but whose name is not known.)

```
call seg_man$get_rel_segment (relptr, relname, copy,
     segptr);
```

relptr is a pointer to a segment whose related segment is desired. relptr is used to index to the KST and pick out the last name of the corresponding segment. This name is then concatenated with relname (e.g., ".link") and seg_man$get_segment is called to obtain a pointer to a segment with the concatenated name.

copy is passed on to get_segment. segptr is returned by initiate as a pointer to the initiated segment and is null if the segment could not be initiated. segptr will also be set to null if the segment corresponding to relptr is not known.

4. seg_man$get_seg_ptr

The following call is provided to return a pointer corresponding to a reference name for a segment which is known.

```
call seg_man$get_seg_ptr (callerptr, callname, segptr);
```

callerptr is a pointer to the calling segment which desires a pointer. segptr is set to null if a segment has not been initiated by the name callname or to the base of the segment if it has been initiated. Implementation is accomplished by a call to initiate with a null directory name since initiate returns a non-null segptr only if the name is already known.

The value of callerptr is ignored in this implementation.

5. seg_man$terminate

The following call is provided to terminate a segment.

```
call seg_man$terminate (segptr, errcode);
```

segptr is a pointer to the segment to be terminated. Makeunknown is called to make the segment unknown. If successful errcode is set to 0, otherwise it is not zero.

6. seg_man$make_seg

The following call is provided to generate an empty segment with specified attributes.

```
call seg_man$make_seg (dirname, entryname, name, max1,
     bitcnt, trewa, segptr);
```

A segment whose maximum length is max1*1024 words, whose bit count is bitcnt bits, whose access attributes are trewa, whose ring brackets are {validation level, validation level, validation level} is created and initiated. Its reference name is name and it is placed in the directory whose pathname is dirname. If dirname is the null character string, "", then the pathname is the pathname of the process directory. entryname is used as the name of the segment in the given directory. If entryname is the null string, "", then the name in the directory of the generated segment is those unique characters generated by the unique id of the segment. If make_seg cannot successfully operate, segptr is returned as a null pointer, otherwise segptr is returned as a pointer to the base of the generated segment.

7. seg_man$set_wdir

The following call is provided to set the name of the current working directory into the KST.

```
call seg_man$set_wdir (wdir);
```

wdir is the name of the working directory to be placed in the KST. The name is stored as a KST name structure pointed to by a pointer in the KST header. If there is currently a working directory name stored in the KST, it is freed. Then space is allocated for the new name and it is placed in the KST.

8. seg_man$get_wdir

The following call is provided to get the name of the working directory.

```
call seg_man$get_wdir (wdir);
```

wdir is returned as the name of the working directory which has been placed in the KST by the set_wdir primitive.

9. seg_man$get_path_name

The following call is provided to obtain the pathname of a segment

```
call seg_man$get_path_name (segptr, dirname,
     name, errcode);
```

For this call segptr is a pointer to the segment whose pathname is requested. If the segment is not known, errcode is set to 1 and the other arguments are ignored. errcode is set to 2 if the segment number is greater than the highest known segment number. If the segment is known, name is set to the value of the last reference name added to the KST entry, the segment number of the superior directory is used to index to the KST entry for the directory containing the segment and dirname is set to the name taken from its KST entry, and errcode is set to 0.

Because the entryname of a segment is not placed in the KST, only if the last reference name of a segment is identical to the entryname will the pathname of the segment actually be dirname||">"||name.

10. seg_man$get_name

The following call is provided to obtain one of the many reference names to a segment.

```
call seg_man$get_name (segptr, namecount, name,
     errcode);
```

For this call segptr is a pointer to the segment whose name is requested; namecount is an integer indicating which of the many names is to be returned where 1 returns the last name, 2 returns the next to last name, etc; name is set to the value of the namecount-th name if a segment corresponding to segptr has been initiated. errcode is set to 0 if a name is returned and 1 if no name is returned, 2 if no name is returned because the segment number n segptr is higher than the highest known segment number, and 3 if the name returned was the first name in the entry but namecount is greater than the name's counted position in the list of names (i.e., namecount of 99 should almost always fetch the first name).

11. seg_man$get_seg_status

The following call is provided to obtain status information about a segment.

```
call seg_man$get_seg_status (segptr, trewa, rings,
     uid, errcode);
```

For this call segptr is a pointer to the segment for which status information is requested. If the segment is not known to the process, errcode is set to 1 and all other arguments are ignored. errcode is set to 2 if the segment number is higher than the highest known segment number. If the segment is known to the process, errcode is set to 0, and trewa, the mode of the segment, rings, the ring brackets of the segment, and uid, the unique id of the segment are set to the values found in the KST entry for the segment.

PL/I Declarations

The PL/I declarations for the parameters to the segment management module are given below.

```
declare

   dirname char(*),          /* pathname of directory */

   name char(*),             /* segment name */

   callname char(*),         /* segment's reference name */

   copy fixed bin(2),        /* copy switch

                                = 0 if value of copy switch
                                  in branch is to be used

                                = 1 if the original segment
                                  is to be used

                                = 2 if a copy is to be made */

   segptr ptr,               /* ptr to segment */

   callerptr ptr,            /* ptr to caller segment */

   errcode fixed bin(17),/* error code = 0 if no error */

   maxl fixed bin(9),        /* maximum length of segment in
                                blocks of 1024 words */
```

```
namecount fixed bin(17),      /* integer indicating which
                                 name is desired */

bitcnt fixed bin(24),         /* bitwise encoding of
                                 mode of segment in usual
                                 interpretation of Trap,
                                 Read, Execute, Write,
                                 Append */

wdir char(*),                 /* name of the working
                                 directory */

rings (3) fixed bin(6),       /* ring brackets */

uid bit(70),                  /* unique id of segment */

relptr ptr,                   /* pointer to segment for
                                 which a related segment
                                 is wanted */

relname char(*);              /* the portion of the segment
                                 name which is known */
```