Identification

Directory Supervisor, System Primitives
C.A. Cushing

Purpose

The directory supervisor provides the primitives for manipulating
directory entries and decides the permission needed to
carry out the requested operation from the intent of the
caller.

Primitive

Certain primitives of the directory supervisor are privileged,
i.e., they are callable only by procedures residing in
system rings, namely the hard-core ring and the administrative
ring. Some of these primitives are necessary to cleanup
and update certain file system tables during activation,
deactivation or reactivation of a segment (finddir, activinfo,
refindb) and another to set up certain values for variables
in system data bases (set_base_dir). Since these functions
are necessary for the well-being of the file system, the
question of whether or not the user has any attribute
(read, execute, write, append) ON with respect to the
segment being referenced to complete the function is irrelevant.
In the terminology of directory control, the user needs
a null access mode with respect to the affected segment.

In the case of estblseg and setusage, the access mode
of the user is important. In order to establish a non-directory
segment the user must have the execute attribute ON for
the directory containing the segment's branch but he doesn't
need any attribute ON for the segment itself. Since,
for example, the user may be expecting a mode change with
respect to this segment at any moment, he is allowed to
establish a non-directory segment to which he has no access.
In order to establish a directory segment the user may
have a null access mode with respect to the directory
containing this segment's branch but he needs at least
one attribute ON with respect to the segment itself.
This is used to restrict a user to a section of the file
hierarchy beneath a particular node (directory).

In order to set the usage status of a particular segment
the user needs the execute attribute ON in the directory

containing the segment's branch. He also needs that attribute ON with respect to the segment which corresponds to the usage being set: the read attribute if setting read usage, write or append attribute if setting write usage, or read, write or append attrbute if setting data-share usage.

The following is a list of the primitives with the access mode needed in the directory containing the segment's branch and possibly the access mode needed in the branch (for the segment) itself.

1. estblseg   (non-directory segment - execute
               directory segment   - at least one attribute
                  ON in branch)

2. setusage   (execute; read in branch for read usage, write
                  or append in branch for write usage, or
                  read, write or append in branch for data
                  share usage)

3. set_base_dir (null)

4. finddir (null)

5. refindb (null)

6. activinfo$rdbranch,activinfo$wrbranch (null)

1. __estblseg__

The primitive __estblseg__ is privileged and is called by the Segment Management Module after it has received an initiate call from a user. This primitive finds the branch to which a given entry points and, through a call to segment control at makeknown makes the segment known to the current process.

        call estblseg (dir, entry, segsw, segptr, uid, optsw,
            slotlist, code);

          dcl dir char(*), /*path name of a directory*/

            entry char(*), /*name of entry in dir which points
                        to the requested segment*/

            segsw fixed bin(1), /*switch indicating whether the
                        user does (=1) or doesn't
                        (=0) want a special segment
                        number given to this segment*/

segptr ptr, /*pointer to the base of the established
            segment, returned by directory supervisor
            or given by caller if segsw is ON*/

uid bit(70), /*unique identifier of the segment
             found in the branch pointed to by
             entry*/

optsw bit(2), /*value of the option switches in the
              branch pointed to by entry, returned
              by directory supervisor*/

slotlist(*)fixed bin(17), /*list of slot numbers for
                          each of the branches in
                          the path name of the branch
                          pointed to by entry,
                          returned by directory
                          supervisor*/

code fixed bin(17); /*if non-zero, it represents the
                    code of an error detected by the
                    file system*/

If the branch to which <u>entry</u> points is a non-directory
branch then the user needs the execute permission in the
directory containing this branch. If it is a directory
branch, the user needs no permission in the directory
which contains it.

The findbranch primitive is called to find the branch
to which entry points. The effmode primitive is called
to determine the effective mode of the user with respect
to the segment. If the segment is a directory segment
then the user must have at least one permission (read,
execute, write or append). Segment control is called
at makeknown to make the segment known and to return the
segment pointer and <u>slot list</u> for the segment. The branch
is then unlocked and control is returned to the caller.

2.   <u>setusage</u>

The primitive <u>setusage</u> is privileged and is called by
the Segment Housekeeping Module after it has received
a segment usage request from a process. This primitive
changes the usage of a given segment by this user from
a previous usage to a new usage status if this new status
is compatible with the usage of the segment by other processes.

        call setusage (dir, entry, newuse, olduse, blocksw,
            blockrtn, code);

```
dcl dir char(*), /*path name of a directory*/

     entry char(*), /*name of the entry in dir which points
                     to the requested segment*/

     newuse bit(2), /*designation of the new way in which
                 ,    the process wishes to use the segment;
                     it may designate read (01), write
                     (10), data-share (11) or no use (00)*/

     olduse bit(2), /*designation of the current usage of
                     the segment by this process*/

     blocksw fixed bin(1), /*a switch indicating whether
                     a process is (ON) or isn't
                     (OFF) willing to wait until
                     the value of newuse becomes
                     compatible with the usage
                     of the segment by other
                     processes*/

     blockrtn label; /*place to which control will be
                     returned if blocksw is ON and a
                     blocking situation occurs (see
                     below)*/
```

The user needs the permission implied in newuse. The
primitive _setusage_ first calls findbranch to find the
branch to which _entry_ points.

If olduse indicates that the process was using this segment
for reading, writing or data-sharing, then the effect
of the usage of this segment by this process must be removed
from the usage and usage-count items in the branch for
the segment. That is, the usage-count item is decreased
by one and, if it is still greater than zero, the usage
item is unchanged. Otherwise if the usage-count turned
zero, then the usage item in the branch is changed to
not-used and if the no-more users switch is ON, it is
turned OFF and a call is made to _notify_ in the process
wait and notify (PWN) module to wake up any processes
that are blocked waiting for this segment to become available.

If olduse indicates that the process was not using the
segment then the usage and usage-count items are not changed
with respect to this argument.

If newuse indicates that the process wishes to use the
segment for reading, writing or data-sharing, then the
usage and usage-count items in the branch must be updated
to reflect this. If blocksw and the no-more-users switch
are both ON, a call is made to addpwt in the PWN module
to put this process on the process-waiting table for the
correct event (so that it may later be notified if the
event has occurred) and control is returned to blockrtn.
If the no-more-users switch is OFF and

1.   if the current usage item indicates not-used,
      then it is set to newuse and the usage-count
      item is set to 1, or

2.   if the usage item equals newuse and is either read
      or data-share, then the usage-count item is increased
      by 1.

For any other combination, if blocksw is ON, the no-more-users
switch is turned ON, addpwt is called and control is returned
to blockrtn or, if blocksw if OFF, an error is reflected
to the calling procedure.

3.   set base dir

The primitive set base dir is privileged and is called
by the file system initializer procedure initialize_branches
(see BL.3) and certain administrative procedures. This
primitive sets all file system activity for a process
to a particular path from the root of the hierarchy.

        call set_base_dir (pathname, code);

        dcl pathname char(*); /*pathname of the particular
                               subtree to be used for file
                               system activity, given by
                               caller*/

No permission is needed to switch the subtree using set_base_dir
since this is a privileged primitive.

The contents of pathname which for example may be

1.   "root>system_root", the subtree used exclusively for
      segments of the trigger, hardcore supervisor and the
      hierarchy reconstruction process (see BH.3.01), or

2.   "root>multics_root", the subtree used for all other
      segments,

are stored in the process definition segment for the current process. The size of <u>pathname</u> is also stored in the process definition segment. Any subsequent call to directory supervisor which specifies the path name of a directory relative to the root (the normal type of path name handed to directory control, see BX.8.00) has <u>pathname</u> prefixed to it. Hence all file system activity after this call is switched to the hierarchy subtree defined by pathname.

4.   <u>finddir</u>

The primitive <u>finddir</u> is called only by segment control. It locates the directory branch to which a given entry effectively points and makes the segment to which this branch points known to the current process. This primitive is used by segment control when it is asked (by directory supervisor or maintainer) to return a segment pointer to a directory which is not known to the process.

        call finddir (dir, segptr, code);

        dcl segptr ptr; /*pointer to the segment with name <u>dir</u>,
                          returned by directory supervisor*/

No permission is needed to find a directory and make it a known segment using this primitive. If dir does not denote the root directory, then findbranch is called to find and lock the branch which points to dir. If dir is not a directory, its branch is unlocked and an error is reflected to the caller.

If dir does denote the root directory (i.e., dir = "root") then the primitive findbranch need not be called because the branch which points to the root is known to the directory supervisor.

In either case, segment control is called at makeknown to make dir known to the process. The branch is unlocked and control is returned to the caller.

5.   <u>refindb</u>

The primitive <u>refindb</u> is privileged and used only by segment control. It returns to segment control selected information from a requested branch. This primitive is used by segment control when activating and reactivating known segments for a process.

```
dcl dp ptr, /*pointer to the base of the directory segment
            containing the desired branch, given by caller*/

    slot fixed bin(17), /*slot number of the branch when
                        originally found by estblseg, given
                        by the caller*/

    uid bit(70), /*unique identifier of the branch when
                 originally found by estblseg, given by
                 the caller*/

    dtbm bit(52), /*date and time this requested branch
                  was last modified, returned*/

    emode bit(5), /*effective mode of the current user with
                  respect to the branch, returned*/

    plist(plistmax)bit(18); /*list of protection numbers
                            defining access bracket, call
                            bracket and gates (see BG.9.00)
                            for the segment pointed to by
                            this branch (plistmax is a static
                            variable initialized to the
                            maximum size of any protection
                            list), returned*/
```

The user needs no permission to find a given branch using
this primitive since it is privileged. The primitive
refindb first checks the unique identification of the
branch defined by dp and slot to see if it matches the
given uid. If there is no match an error is reflected
to the caller. If there is a match the dtbm, emode and
plist items are taken from the branch and returned to
the caller. The effmode primitive of access control must
be called to compute the effective mode (emode) and the
protection list (plist) from the access control list for
the branch.

6.   activinfo$rdbranch, activinfo$wrbranch

The primitives rdbranch and wrbranch are privileged and
are used only by segment control. rdbranch is used by
segment control to obtain the information from a given
branch which is needed to activate a segment. wrbranch
is called to update this information when the segment
is deactivated or when segment control is asked specifically
to update it.

```
call activinfo$rdbranch (dp, slot, uid, actsw, itemsptr, fmp
     code);

call activinfo$wrbranch (dp, slot, uid, actsw, itemsptr, fmp
     code);

dcl dp ptr, /*pointer to a directory segment, given by
               caller*/

    slot fixed bin(17), /*slot number of a branch in the
                           directory, given by caller*/

    uid bit(70), /*unique identifier of the branch, given
                    by caller*/

    actsw bit(1), /*new setting for the file-active switch
                     in the branch, given by caller*/

    itemsptr ptr, /*pointer to a structure to be filled in
                     with certain items from the branch for
                     rdbranch or a structure containing new
                     values for these certain items for
                     wrbranch*/

    fmp ptr; /*pointer to the file map of the branch*/
```

The user does not need any permission to read from or
write into the branch using these primitives since they are
privileged.

If rdbranch was called, then the requested branch must be
locked. Before reading or writing the information which is
pertinent when a segment is to be activated or deactivated,
the secondary lock in the branch which locks only this subset
of information must be set. When this lock is set, this
information may be read or written. The file-active switch is
set to actsw, all locks which were set are removed and control
is returned to the caller.