Published: 04/05/67

Identification

Segment Management Module (SMM) - Overview
S. L. Rosenbaum

Purpose

In the Multics world of segments, the Segment Management Module (SMM) is an interface for translating symbolic call names into segment pointers. (A call name is a symbolic name by which a process knows a segment, i.e., a name by which a procedure or group of procedures within a process references a segment. A segment pointer is an ITS pair pointing to the base of a segment.) The SMM is composed of a group of administrative ring slave procedures and is a major interface to the basic file system segment manipulation mechanisms (BG.3.00). As such, ordinary user procedures may call the SMM as well as the primary intended caller, the Linker (BD.7.04).

The main function of the SMM is to translate a call name into a segment pointer for the Linker. To facilitate the translation, the SMM records all the "name-into-segment pointer" translations for a process in a table, the Segment Name Table (SNT). Each process has its own SNT. Both the SMM and the SNT reside in the administrative ring and hence, serve as a logical bridge between the user ring's use of symbolic call names and the basic file system's use of segment pointers. The basic file system resides in the protected (or hard-core) supervisor ring. (See Figure I.)

In addition to translating a call name into a segment pointer for the Linker, the SMM provides primitives for adding entries to the SNT (normally called by the Linker), primitives for removing entries from the SNT and primitives for obtaining information about entries in the SNT. Although all the primitives are available to both system programs and user programs it is suggested that the primitives which alter the SNT should be invoked only for handling special situations, e.g., situations which the Linker does not handle satisfactorily and situations for which neither library procedures nor commands are available.

This section describes the interfaces of the SMM and discusses the general features of "initiating", "terminating" and

"relating" call names as provided by the SMM. Section BD.3.01 describes the structure of the Segment Name Table in detail; section BD.3.02 describes the SMM primitives.

The reader should be familiar with section BG.00 - "Overview of the Basic File System" and the description of directories, path names and entry names which is contained in section BX.8.00 - "Overview of the File System Commands".

Definition of Terms

An entry in the Segment Name Table (SNT) contains a call name, information about the segment associated with the call name, information about the relation of the entry to the current process and information about the availability of the entry to segments already known to the process.

At this point it is convenient to define some terms which are used throughout section BD.3.

> A call name is known if at least one entry for it appears in the SNT.
>
> A call name is initiated if it is a known name and the information about its associated segment includes a segment pointer.
>
> A call name is terminated if no entry for it in the SNT is associated with a segment pointer, i.e., a terminated name is a call name which is not initiated.

A call name is related if a specific entry in the SNT for the call name must be used by a specific segment when that segment references the call name. For example, assume there is an entry which translates the call name "x" into the segment pointer "i". If that entry is related to segment "n" (where "n" is the pointer to the segment and not a call name for it), then whenever segment "n" references the call name "x", segment "n" gets segment "i". Loosely speaking, the call name "x" (and its associated segment "i") are internal to segment "n" and segment "n" can get no segment other than segment "i" for the call name "x".

A related entry can be local or global. A related entry is local if it is available to only one segment - the segment

to which the entry is related. A related entry is global if it is available to any segment seeking the call name. (All entries which are not related entries are global entries. That is, they are available to all calling segments.)

"Relating" call names to segments enables the use of multiply-defined call names which, in turn, decreases the need to reserve special names for system programs; relating names also facilitates the use of sub-systems. That is, the user of a packaged system need not be concerned with the system's calls conflicting with his own calls if the packager of the sub-system has established all the necessary relationships within the sub-system. To aid the packager, the system command relate (section BX.8.13) enables the packager to establish relationships without re-programming his sub-system. The Linker automatically establishes a relationship between a procedure segment and its linkage section segment with an SMM primitive, getseg. In particular, when the Linker initiates a procedure's call name (causes an entry, i.e., a name-to-segment translation, to be made in the SNT for the call name of the procedure), the Linker also directs the SMM to relate the call name of the linkage section to the procedure segment (causes an entry to be made in the SNT for the call name of the linkage section which must be used by the procedure segment when it refers to the linkage section's call name).

Although the implementation of related names and their use is relatively straightforward, the concept of relationships is more easily explained by example rather than in the abstract. Let it suffice for now to say that relationships enable two segments which reference the same call name to co-exist within the same process even though they neither want (nor do they get) the same segment for the call name. From the SMM's point of view, this means that the SNT may have several entries for a single call name; of course, the SMM may need to know what segment is referencing the call name in order to decide which entry to use for the call name. Information attached to each entry in the SNT indicates:

1. if the entry is related and if so to which segment.

2. if the entry is local, i.e., the entry is available only to one segment (the one to which it is related), or if the entry is global, i.e., the entry is available to all segments seeking the call name.

Obtaining Segment Pointers for the Linker

When a procedure symbolically references an external segment for the first time, a linkage fault occurs. (Section BD.7.00 discusses dynamic linking.) The Linker, the system module which handles linkage faults, calls the SMM for a pointer to the base of the segment associated with the symbolic call name. The Linker assumes that the SMM can obtain the pointer on the basis of the symbolic call name which caused the linkage fault and knowledge about the faulting procedure, i.e., the identity and location of the procedure which referenced the segment by call name. The Linker passes the call name (for which a segment pointer is wanted) and a pointer to the faulting segment (the procedure segment which wants a segment for the call name) to the SMM. The SMM checks the SNT to see if there is an entry for the call name which is available to the faulting segment. An entry for a call name is available to a faulting segment if

1. the entry in the SNT is associated with the ring in which the faulting segment resides (that is, the entry was created by a call from the same ring in which the faulting segment resides) and

2. the entry is local to the faulting segment, i.e., the entry is directly related to the faulting segment, or the entry is global.

The above conditions are established for an entry when the SMM originally creates the entry in the SNT as is described later in this section. A global entry is used only if there is no local entry available - either type of entry must meet the first condition.

The first time that a procedure in the process references a call name, the SMM will probably not find any entry for the call name in the SNT. The SMM must then initiate the name for a segment, i.e., create an entry for the call name and its associated segment which the faulting segment can use. To accomplish this, the SMM invokes the Search Module (described in sections BD.4.00 and BX.13.00) at its entry search. The SMM passes (to the Search Module) the information it received from the Linker, i.e., the symbolic call name and the pointer to the faulting segment.

The SMM expects the Search Module to return the location of the desired segment in the file system hierarchy, i.e., the segment's path name. The Search Module returns the path name of the directory which contains the entry for the segment and the name of the entry in the directory. (So if the path name of the directory is "> a > b" and the name of the entry is "x", then the path name of the segment is ">a >b >x".) Now the SMM calls the Directory Control primitive estblseg, giving estblseg the segment's path name and getting back the pointer to the segment. (The interface between the SMM and estblseg is more fully described later in this section.)

Now with both ends of the entry and the bridge between the extremities, i.e., the call name from the Linker, the segment pointer from the Directory Control and the segment's path name from the Search Module, the SMM takes the following steps to establish the availability of the entry.

1. The SMM gets the ring number of the faulting segment and makes the entry available to the ring in which the faulting segment resides.
   (NOTE: the SMM is responsible for the availability of the SNT entry to the faulting segment; the basic file system (Access Control) is responsible for the availability of the segment associated with the SNT entry to the faulting segment.)

2. The SMM makes the entry global.

Finally, the SMM updates the SNT to include the entry and then returns the segment pointer to the Linker.

On subsequent calls to the SMM (for that same call name - wanted by segments faulting in the ring to which the entry is available), the SMM finds that the name is initiated and available and simply returns the segment pointer to the Linker.

If the Linker chooses to, it can ask the SMM for two segment pointers: one to the procedure segment and one to the linkage section segment. In this case, the SMM proceeds as above but takes the following steps after having found or made an initiated entry for the procedure segment and its call name.

1. The SMM checks the SNT to see if there is an entry related to the procedure segment for the linkage section segment with the same ring of availability as the entry for the procedure segment. In addition the call name for the entry is comprised of the entry name of the procedure segment concatenated with a character string (normally ".link") supplied by the Linker. If the SMM finds such an entry it goes immediately to step 6 below.

2. If there is no such entry, the SMM calls estblseg with the directory path name of the procedure segment and the call name of the linkage section. (For example, if the Linker wants segment pointers for the call name "x" and its linkage section - and the segment for "x" has the path name ">a >b", then the linkage section has the path name ">a >b.link".)

3. The SMM makes a copy of the linkage section segment and puts the copy in the Process Directory and calls makeunknown in Directory Control to release the original linkage section segment. (The Linker operates on copies of linkage sections and not the original.) "Making a copy" consists of calling Directory Control's appendb to make a branch in the Process Directory for the new segment, calling estblseg to get the segment pointer for this new segment and copying the original segment into the new segment.

4. Having all the pieces for the entry for the linkage section segment, the SMM relates the new entry to the procedure segment and makes it global.

5. The SMM updates the SNT.

6. The SMM returns both the pointer to the prccedure segment and the pointer to the linkage section segment to the Linker.

Related Call Names

When a user wants to ensure that a procedure uses a specific sub-procedure regardless of the process in which it is running, the user (or packager) can "relate" names with

the relate command (section BX.8.13). To understand how the SMM handles these relationships, a brief discussion of the relate command and its immediate effects seems pertinent here.

The relate command expects its caller (the packager) to supply information about the desired relationship for the procedure. The command puts the information into the file system hierarchy for the SMM to use when the procedure is actually invoked in a process - in any process by any user.

To be more explicit, the packager supplies the call name of the procedure segment and its location in the hierarchy (its path name), the call names of all its sub-procedure and/or data segments to be related and directions to be used by the SMM to get the related segments when they are needed by the procedure segment. The packager can indicate that the segment for a related name should be created, is located in the hierarchy by a specific path name and/or can be searched for. In addition, the packager can indicate whether a related call name should be initiated or should simply be made known at the time its caller (the procedure segment) is initiated.

The relate command creates a segment in the file system hierarchy with all the relationship information supplied by the packager; this created segment is called a relationship segment and its location in the hierarchy is determined by the packager when he invokes the relate command. In normal use, the packager will probably tell users of the package about the location (path name) of the relationship segment and mention neither the path name of the procedure segment nor the path names of the procedure's related segments.

(Note: With the relate command, the packager relates call names to entry names and not to segment pointers. This is necessary since a segment pointer is not assigned to a segment until the segment is part of a process and then only has meaning within the scope of the process.)

The SMM's Use of Related Names

In the course of obtaining a segment pointer for the Linker, the SMM reaches the stage where it has a path name for

the segment and wants estblseg to return the segment pointer. If estblseg indicates that the segment is a relationship segment (the relate command sets this vital piece of news in the form of a switch attached to the branch in the hierarchy when the command creates a relationship segment), then the SMM must expend a little more effort to obtain the segment pointer wanted by the Linker. Basically, the SMM breaks the relationship segment down and makes all the related names known (following the directions specified in the relationship segment) in addition to initiating the procedure via estblseg. (Remember - the path name of the procedure segment was given to the relate command and stored in the relationship segment.) The SMM puts the instructions for getting the segments for the related names with their respective known entries and then frees the relationship segment by calling makeunknown. The entries for the related names have the same ring of availability as the entry for the procedure segment to which they are available.
Now when the SMM is invoked to get a segment pointer and the SMM finds an entry in the SNT for a related name which meets all the requirements (name, ring, and caller), the SMM uses the information associated with the known entry to get the segment pointer. The information may direct the SMM to create a new segment, to use a specific segment already in the file system hierarchy and/or to search in the file system hierarchy for a segment.

To create a new segment for the call name, the SMM calls Directory Control's primitive appendb to create a new entry in a directory which exists in the file system hierarchy. (Besides indicating that the SMM should create a segment, the information stored with the known entry specifies by path name the directory into which the new entry should be put and the name for the entry.) The SMM obtains the segment pointer for the new segment by calling Directory Control at estblseg. The SMM updates the SNT so that the name is initiated and then returns the segment pointer to the Linker.

When the information indicates that a specific segment in the file system hierarchy should be associated with the call name, the SMM immediately calls estblseg to obtain

the segment pointer. The SMM locates the desired segment in the hierarchy by the path name specified by the information stored with the known entry. If there is no segment with the given path name (estblseg generates an error) and searching is requested - or if the information associated with the calling name initially directed the SMM to search - the SMM invokes the Search Module as previously discussed. Obtaining the desired segment pointer in one way or the other, the SMM records the initiation in the SNT and returns the segment pointer to the Linker. (See figure III.)

The SMM terminates an entry by deleting it from the SNT.

When a segment is no longer needed (its segment pointer is not associated with any call name), the SMM invokes Segment Control's primitive makeunknown. If this segment's delete switch in the SNT is on, the SMM also calls Directory Control's primitive delentry. (Calls to the SMM primitive setdel change the setting of the delete switch; initially, the SMM sets a segment's delete switch off.) The call to delentry removes the segment from the file system hierarchy. If the segment has entries related to it, the SMM terminates all local entries and removes all the related information from the global entries, i.e., terminates all local related entries and unrelates all global related entries.

Primitives

The SMM's primitives are all unprivileged and designed for use by both system programs and the general user. These primitives perform two functions: they alter the SNT and they retrieve information from the SNT. (Relationships are internally established by a process for itself with the SMM primitive setnamestatus; relationships are established externally by a process for itself and other processes with the command relate described in section BX.8.13.)

Specific error conditions and error codes are described in section BD.3.02 along with the primitives. The SMM uses the standard error handling mechanism described in Section BY.11.00.

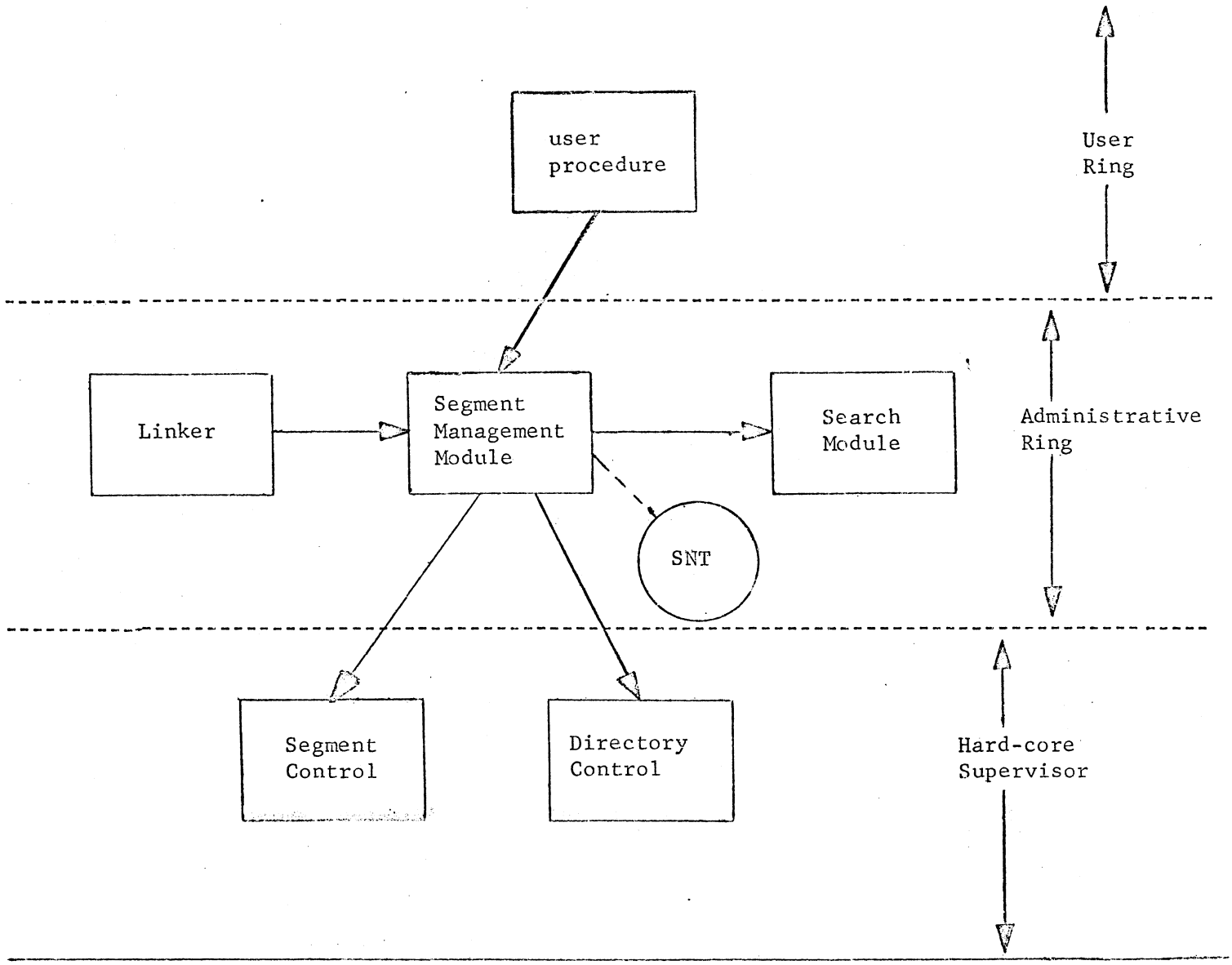
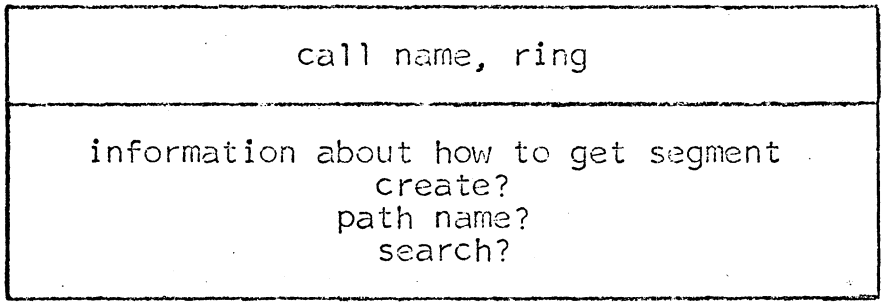
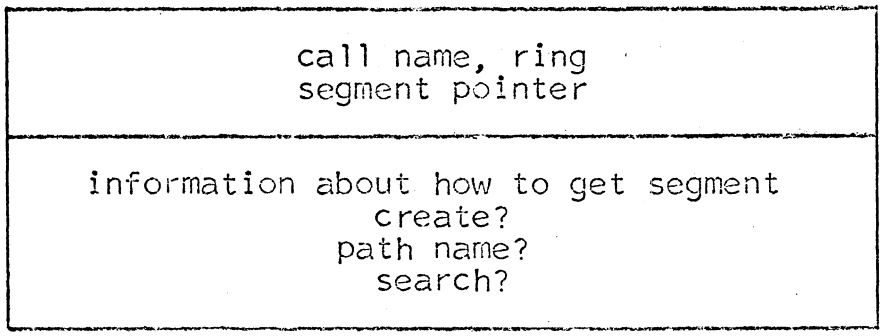


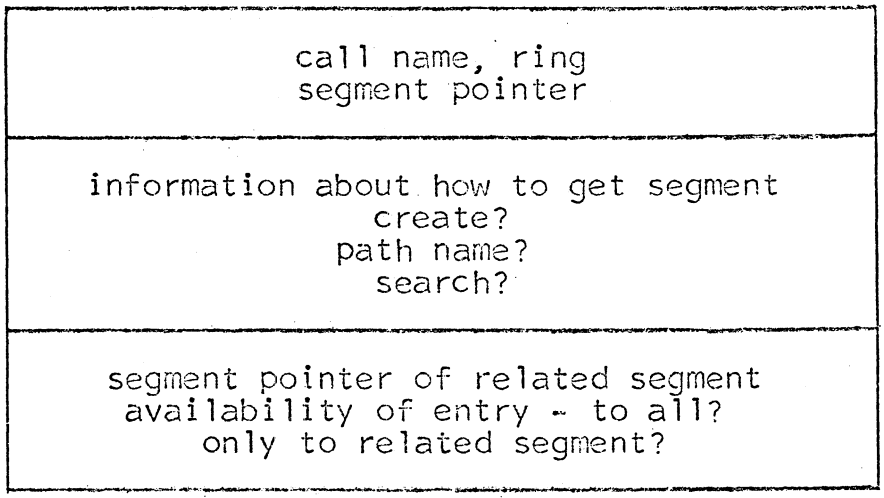
Figure I - The Segment Management Module's Interfaces



known and terminated name



known and initiated name



known, initiated and related name

Figure II - Representation of various types of entries in the Segment Name Table (SNT)

Figure III - Obtaining a segment pointer for the call name "x".

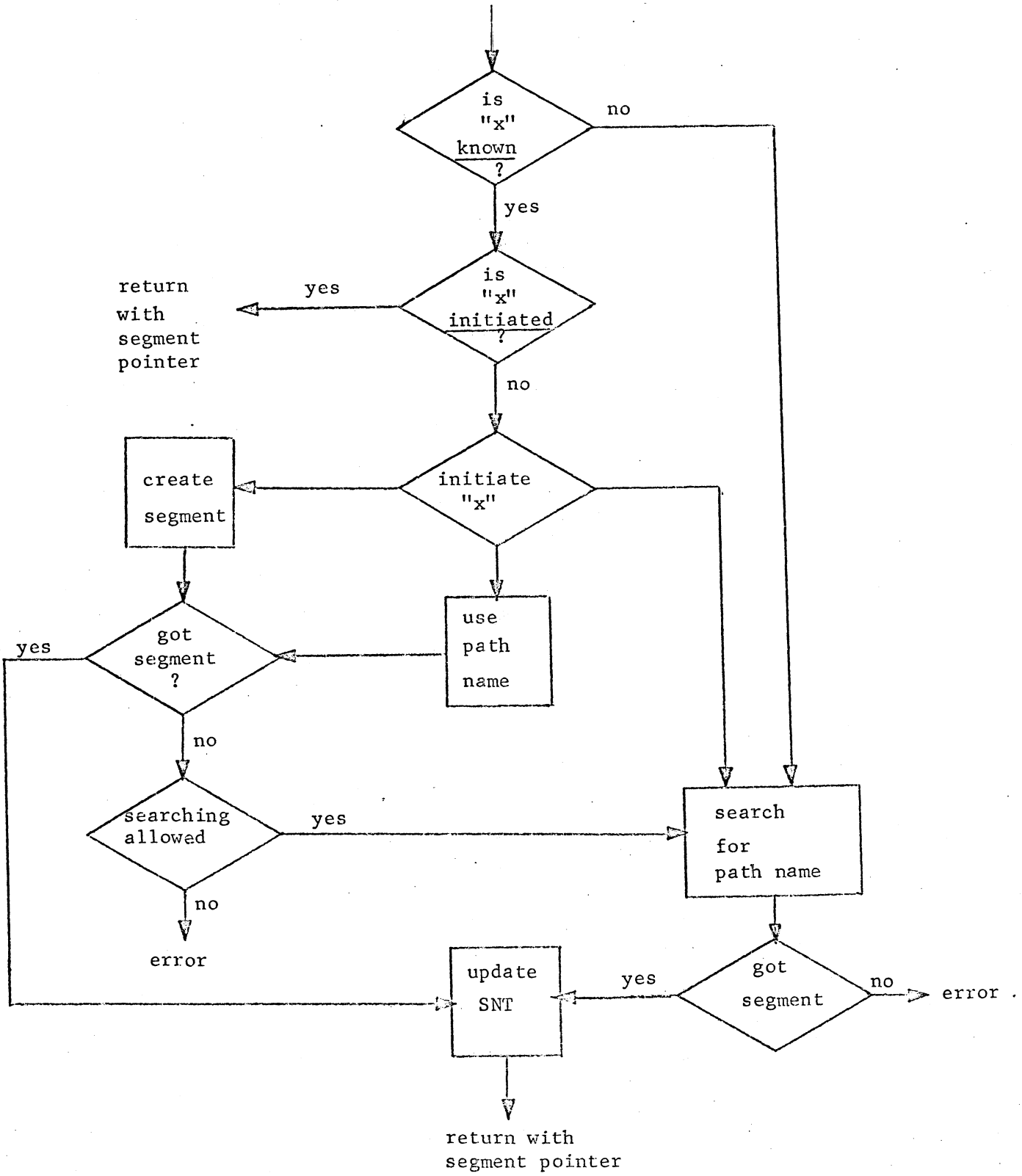


Figure IV - Terminating a call name