Published: 11/03/67

Identification

create_proc
R. L. Rappaport

Purpose

This document describes the program called in order to create a process.

Discussion

As is explained in BJ.2.01, creating a process means creating a group of segments. Procedure create_proc is the "main" program involved in the creation of these segments. Create_proc executes in the administrative ring and is only callable from that ring. The calling sequence is:

```
call create_proc (id, pit, l1, opt_stack, l2, restrict,
                  l3, wdt, l4, account, create_linker_
                  segs, user_id);
```

where:

id	is the process id which is returned to the caller.
pit	is the path name of the process initiation table.
l1	is equal to "1"b if a link should be established in the new process directory for pit and is equal to "0"b if a copy of pit should be put into this directory.
opt_stack	is the path name of the option stack segment.
l2	is similar to l1 but l2 refers to opt_stack.
restrict	is the path name of the restriction segment.
l3	similar to l1.
wdt	is the path name of the working directory table.
l4	similar to l1.
account	is the path name of the accounting segment (Ordinarily this argument is zero and the created process has the same account as the creator. Only privileged processes can specify something other than zero.) Note that a link to the appropriate account is always made.

create_linker_segs is the entry point of a procedure which constructs a linker package. That is, it creates the needed segments and builds the pre-linker driving table.

user_id is the user group to which this new process will belong. This argument is ordinarily zero and the new process is put in the same group as its creator. Only privileged processes can specify otherwise.

Create_proc is basically a driving program in that very little of the computing associated with process creation is done by create_proc. Instead create_proc calls a variety of subroutines which perform the actual tasks.

All the segments that are created at process creation time reside in the process directory of the new process. Therefore, the first task of create_proc is to create the new process directory. This is accomplished by calling the hardcore ring procedure estblproc (see BJ.8.02). Estblproc first establishes the id for the new process by calling get_proc_id (see BJ.7.03). Using this id, a directory is established in the process directory directory. The name of this new process directory is:

>process_dir_dir>fggxxbz

where fggxxbz is the guaranteed to be non-obscene character string representation of the process id created by get_proc_id.

With the new process directory available, create_proc can proceed with its tasks. In the calling sequence of create_proc a group of path names are specified along with a group of "switches". Depending upon the setting of a particular switch, create_proc either establishes a link in the new process directory to the given segment or it copies the contents of the given segment into a brand new segment established in the new directory. The linking, if appropriate, is accomplished by calling the file system primitive appendl (see Section BG.8.02), whereas the segment copying, if appropriate, is accomplished by calling subroutine copy_seg (see Section BJ.8.05).

Having established the links and/or segments specified in the calling sequence, create_proc performs a call to the entry create_linker_segs (see Section BJ.8.03) specified in the calling sequence. Create_proc passes the path name of the new process directory on the call. This procedure

creates, in the new process directory, the segments that the new process will need in order to be able to dynamically link itself. Among these segments are a Segment Name Table (SNT) (see Section BD.3.01), a linkage section for the linker (i.e. linker.link), and a linkage section for the segment management module (i.e. SMM.link). Create_linker_segs then compiles a table, similar in format to the Segment Loading Table (SLT, see Section BL.2.01), which lists the path names of the segments that the new process will have to pre-link in order to initialize the linker in the address space of the new process. This table, known as the pre-linker driving table lists all the segments created by create_linker_segs and in addition procedure segments such as the linker and the segment management module. This table is returned to create_proc.

At this point all the administrative ring segments of the new process have been created and create_proc calls to a hardcore ring procedure to create the hardcore ring segments that will be needed by the new process. The hardcore ring procedure, create_hardcore_segs (see BJ.8.04), is passed the id of the new process, the user id (or the null character string, see below) of the new process. The specification of the null character string as the argument "user_id", is used to indicate that the new process should be put into the creator's group. In fact most processes may only create new processes in their own group. This restriction is implemented by having estblproc ignore this argument unless the calling process is one of the privileged few that may specify otherwise. Create_hardcore_segs creates a Known Segment Table (KST, see Section BG.1), a Process Data Segment (pds, see Section BJ.1.03), and a process definition segment (pdf, see Section BJ.1.06) for the new process.

Upon return from create_hardcore_segs, the new process appears to be a normal blocked process. Therefore create_proc simply awakens it by calling wakeup (see Section BJ.3.02). Having done this create_proc returns to its caller. Figure 1 is a flow diagram of create_proc.

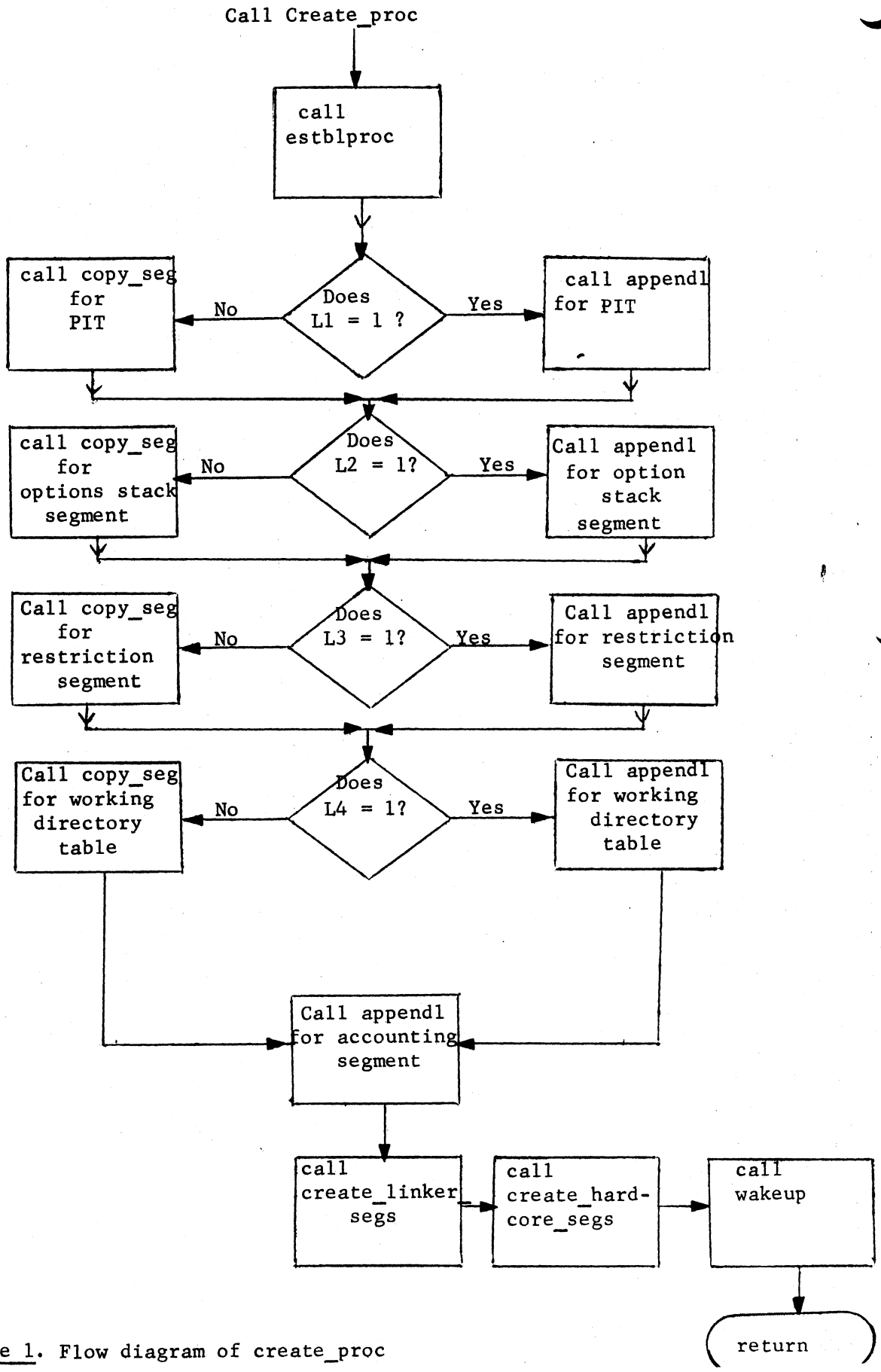

Figure 1. Flow diagram of create_proc