Identification

General specifications for segment formats

C. Garman

The uses of segments in Multics fall into three broad categories:

1. Procedure segments which provide a source of instruction words for execution by a processor attached to a Multics configuration;

2. Data segments which are to be manipulated by the procedures in (1) above; and

3. Temporary or "scratch" segments, used in the production of new editions of (1) and (2) above, as well as for storage of variables required to properly perform the processing functions required, or for the maintenance of particular Multics Standards (e.g., the (per-process) stack and linkage segments used to assist the implementation of pure-procedure segments.).

Various attributes are attached to segments found in the Multics hierarchy, of which the following are the most important:

1. location (directory path-name)
2. identifier (entry name(s))
3. length (in multiples of 1024 words)
4. Access ("TREWA" and ring brackets)
5. Segment number (when a segment has been made known in a process)

(Many other attributes serve to more rigorously describe individual segments, but their range of usage or interest is not so universal (e.g., the unique ID for an entry in a directory, or whether or not a segment has any pages in core.)

Location of Segments in Directory Hierarchy

Each segment in Multics is a branch in some directory (at least nominally, in the case of hardcore system segments) which is itself a branch in a directory, etc, until the chain stops at the root directory (See BG.0, Overview of the Basic File System, and BD.6.02, System Skeleton). Of most importance, perhaps, are (1) user_dir_dir, a directory containing the various directories for user-associated segments, (2) process_dir_dir, which contains one directory per process known to Multics, and (3) the system library directories, which contain the segments generally available to user processes when required for the purposes of searching for dynamically linkable procedures and data bases.

Entry name(s) of Segments

Each segment in a directory has at least one name associated with it to be used as an identifier. This identifier may be any string of bits of length 288, but by convention the bits are ~~considered as any~~ restricted to be combinations of the ASCII graphic characters, of length 32 or less, left justified, with the ASCII character SP used as padding on the right.

Length

The Basic File Systems maintains information about the length of a segment in terms of the number of 1024-word blocks which it occupies.

Since this number is slightly on the coarse side, (36864 bits, or 4096 9-bit ASCII characters) the Basic File System maintains an item in each branch, the bit-count, which by convention tells precisely how much information is of value in the segment when it is treated as data. All system procedures set or rely on these bit-counts, e.g., a context editor would set the bit count to 9 times the number of legitimate characters in the segment, while a command to provide print-out of that segment would examine only bit-count/9 characters.

Procedure segments, while intrinsically self-identifying, still would have their bit-counts set for the cases in which they themselves would be treated as data (e.g., for inclusion in an "archive" segment).

Segments in the process directory for a given process need not, in general, have their bit-counts set; thus the length of the stack segment, as determined by the forward pointer of the current stack frame, may change 200 times per second, so it would be useless to try to maintain the bit-count. Likewise the next available location in a combined linkage segment is at a known place in that segment, such that the single program which needs the information may always easily find it. A third example might be a scratch segment used only by a compiler, which knows the structure and format of the segment and thus does not need to know the bit-count.

Note that any segment whose bit count is not word aligned is not immediately usable as input to many commands.