Published: 10/13/66
(Supersedes: BE.8.00, 05/4/66;
BE.8.00, 06/23/66)

Identification

Overview of the Pseudo-supervisor
D. H. Johnson

Purpose

The pseudo-supervisor is a collection of procedures which provide basic supervisory functions during the execution of a process in the GE645 simulation system. The pseudo-supervisor consists of four modules which are described in detail in the following subsections of this manual:

BE.8.01 Linker for the Pseudo-supervisor
BE.8.02 Page Management for the Pseudo-supervisor
BE.8.03 Segment Management for the Pseudo-supervisor
BE.8.04 Linkage building for ordinary slave procedures in the Pseudo-supervisor

Introduction

The pseudo-supervisor consists of four modules: 1) linker, 2) page management, 3) segment management, and 4) linkage building.

The linker is the module which establishes intersegment references at execution time. Page management keeps track of pages of GE645 simulated memory which are available for assignment to segments as requested. Segment management allows a process to create, release, grow, truncate, or ask questions about a segment. Finally, linkage building gives a process the facilities to construct and interrogate linkage section information.

All ordinary processes using the simulation system will have intersegment referencing. (See the description of special segments INIT and .INIT in MSPM Section BE.7.07., Loader). There is need then for a standard linker module which will be automatically invoked whenever a linkage fault occurs and which will cope with all of the various possibilities in a linkage section. (See MSPM Section BD.7.01., Linkage Section, for the detailed structure of a linkage section.) The linker module also provides for dynamic loading of segments when first referenced. This relieves the programmer from declaring and having loaded all segments before execution. The linker module is also the point of departure for debugging programs (See BE.12.01).

A process begins in the simulation system with the entire GE645 memory allocated to it. This is accomplished when the 645 simulation loader first loads all user specified segments and then allocates the remaining pages of 645 memory to the stack segment. Since all of physical memory is then accounted for in the descriptor segment and page tables, routines may manipulate and have access to all of 645 simulated memory. During execution a process may have need to modify itself in one of the following ways. It may want to create a new segment, as an EPL procedure will do when the first STATIC storage class variable is referenced. It may need to grow existing segments, again as an EPL procedure will do whenever any unique STATIC storage class variable is first referenced. It is also desirable to be able to truncate parts of segments and to release entire segments when they are no longer useful to a process. This may become very desirable in the 645 simulation system where a process is limited to the size of 645 memory. These facilities are provided by the segment management module of the pseudo-supervisor. As the above functions are performed by the segment management module there often arise demands for additional pages of memory and the disposition of released pages. These requests are handled by the page management module, which maintains a pool of free pages of memory and obtains additional pages from the stack segment whenever the free page pool is exhausted.

The pseudo-supervisor also provides a module to build and query linkage section information. Procedures in this module will create linkage section segments, add linkage blocks to existing linkage section segments, insert linkage information into linkage blocks, and see if specified symbolic references are defined in linkage sections. Current known users of some of these routines include the SHELL (see MSPM Section BX.2.00, the SHELL), system option setting procedures (see MSPM Section BX.12.02, Creation of options), and EPL procedures which use STATIC storage (see EPL Design Journal 6, B0022, section titled Data Segments).

Figure 1 is a block diagram of the modules of the pseudo-supervisor. The solid lines represent flow of control through the use of formal calling sequences. Calls are in the direction of the arrows and formal returns are implied. The circles in the diagram represent the data bases. Dashed lines indicate the flow of data between modules and data bases. Figure 2 shows the procedures within the modules of the pseudo-supervisor and the inter-relationship between procedures. The procedures themselves

are explained in the following four subsections of this manual. In this paper, procedure entry names are associated with specific actions wherever possible by including the procedure entry name within parentheses near the description of the action.

Usage

The part of the pseudo-supervisor necessary for handling linkage faults and dynamic segment loading will be included in all processes. This is done by the special inclusion policy of the MRGEDT command in the 6.36 system and the 64.5 Driver in the 64.5 system. Other pseudo-supervisor procedures may be loaded in one of two ways. They may be loaded before execution by explicitly telling the loader to load them or they will be loaded dynamically when first referenced during execution. The pseudo-supervisor procedures all exist on the 645 segment library tape. The pseudo-supervisor consists of many segments. There is one segment, pseudo_ supervisor, which contains entries for all of the procedures. This segment provides the interface between EPL compiled procedures and the pseudo--supervisor. It must be used by EPL compiled programs to insure argument compatibility. The actual pseudo-supervisor procedures exist in separate segments. See the summary of the pseudo-supervisor segments at the end of this section.

Linker

The linker module provides automatic linking of symbolic intersegment references during execution. A linkage fault will occur upon the first execution of an instruction whose address points to another segment indirectly through the linkage section. At this time the linker module is invoked and takes the following steps to accomplish the linking of the reference:

1. Save the state of the process when the fault occurred. (f2catc)
2. Determine where the fault occurred from the saved machine conditions. (linker)
3. Determine what type of intersegment reference is being attempted. (linker)
4. If requested, trap to a user specified procedure before completing the link (linker)
5. If necessary, call the segment management module (segman) to obtain the number of the referenced segment. (linker)
6. If the segment is not in the segment name table, dynamically load the segment from the 645 segment library tape. (search)

7. If an external symbol is referenced, call the segment management module (segman) to obtain the number of the linkage section segment containing the external symbol definition. (linker)
8. Search the linkage section for the definition of the external symbol. (linker)
9. Change the fault to the desired address. (linker)
10. Modify enough of the machine conditions so that the fault will not occur again. (linker)
11. Restore the process to the state it was in before the fault happened. (f2catc)

The entry search scans the library dictionary for a given segment name and if present, creates the segment, allocates memory to the segment, and causes it to be read into memory from the 645 segment library. If present, the linkage section and symbol table segments will also be read into memory from the library. In addition to the functions described previously, the linker module contains an entry (link_) through which a user may force the linking of an intersegment reference. This entry was provided for implementation of the INITIAL attribute of EPL.

Page Management

The page management module provides primitive tools for manipulating pages in the simulation system. It is usually called by the segment management module to obtain a new page (newpag) or to release a page (relpag) no longer needed by a segment. Page management has two sources for satisfying requests for a page. The primary source is the free page pool which contains lists of absolute 645 addresses of 64 and 1024 word pages that are not currently assigned to any segment in the process. If the free page pool is exhausted, page management examines the stack segment to see if it has unused pages that will satisfy the request. If so, the stack is truncated and the trunated page(s) are placed in the free page pool. The operation on the stack and the number and type of resultant additions to the free page pool depend on the page sizes of the stack and the request. It is possible to remove from 1 to 31 pages of the stack to satisfy the request for 1 page. A page released from a segment is simply appended to the appropriate list in the free page pool.

Segment Management

Segment management provides the tools necessary for manipulating segments in the simulation system. It is called by the other modules of the pseudo-supervisor and may also be called by user procedures. Segment management

provides the following functions:

1. Return the number of a segment specified by name. (segman)
2. Determine whether a segment is known to the process, i.e., if the segment name exists in the segment name table. (segpr_)
3. Return the current and maximum lengths of a segment. The current length is the offset of the last word in the segment actually used. It may vary as the segment grows or truncates words. The maximum length is the offset of the last word possible for the segment. It must not be greater than the largest offset allowed by the total number of entries in the page table. (length)
4. Create a new segment in the process. This creates entries in the segment name table, segment length table, and descriptor segment but does not allocate any pages to the segment. If a paged segment is created, its page table is also created automatically. The descriptor segment, segment name table, and segment length table may have to be given more pages when new entries are added. (newseg)
5. Add a fixed number of words to a segment in the process. This request may add pages to the segment. (grow)
6. Release a segment, indicating that it is no longer known to the process. (relseg)
   This request releases all pages allocated to the segment and its page table. References to the segment in various system tables are removed.
7. Truncate a segment by a fixed number of words. If any pages are released they are returned to the free page pool. (trunct)

Linkage building

There are instances where linkage information is generated during execution. The SHELL must generate linkage during execution because it does not know until then what references it has to make. Linkage for STATIC variables in EPL procedures is created when the variable is first referenced. The linkage building module performs the following functions:

1. Determine whether a particular symbol is defined in the linkage section of a specified segment. (sympr_)
2. Generate a normal link pair and link definition in the linkage section of a specified segment and return a pointer so that the user can complete the reference at a later time through the normal linking mechanism of the linker module. (linkmk)
3. In addition to generating a normal link pair and link definition, also generate the necessary information in a linkage section which will cause a trap to a specified procedure to occur when the reference is linked later. (trapmk)

4. Add the necessary external symbol definition to a specified linkage section so that other segments may successfully reference the symbol name. (defmak)
5. Add a fixed number of words to a segment and then generate an external symbol definition for a given symbol so that intersegment references to the symbol will point to the first word added to the segment. (symmk_)

In performing functions 2-5 it may be necessary for the linkage building module to create linkage section segments, add linkage blocks to existing linkage section segments, and add words to existing linkage blocks.

Summary of Pseudo-supervisor segments

segment name	descriptor class*
1. f2catc	MP,SA
2. linker	SP,SA
3. link_	SP,SA
4. linkmk	SP,SA
5. trapmk	SP,SA
6. defmak	SP,SA
7. symmk_	SP,SA
8. sympr_	SP,SA
9. newseg	SP,SA
10. relseg	SP,SA
11. grow	SP,SA
12. trunct	SP,SA
13. segpr_	SP,SA
14. segman	SP,SA
15. length	SP,SA
16. newpag	SP,SA
17. relpag	SP,SA
18. free_page_pool**	D,SA,WP
19. get_put***	MP,SA
20. search	SP,SA
21. pseudo_supervisor	SP,SA

* descriptor class code
MP - master procedure
SP - slave procedure
SA - slave access
D - data
WP - write permit

** free page pool and stack information
*** used by pseudo-supervisor procedures to read and write any segment, e.g., page tables and descriptor segment.

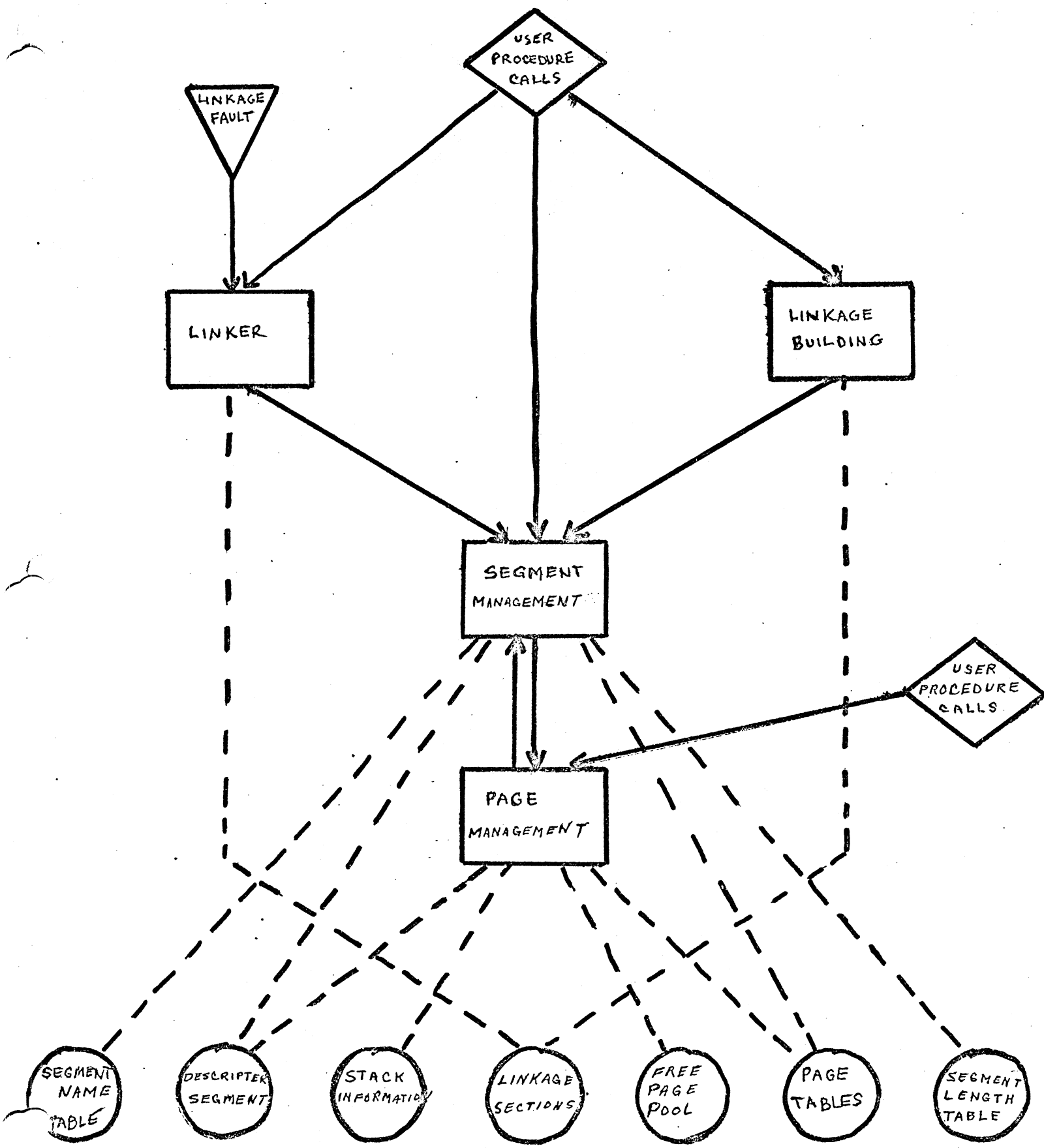

Figure 1. Pseudo-supervisor Modules

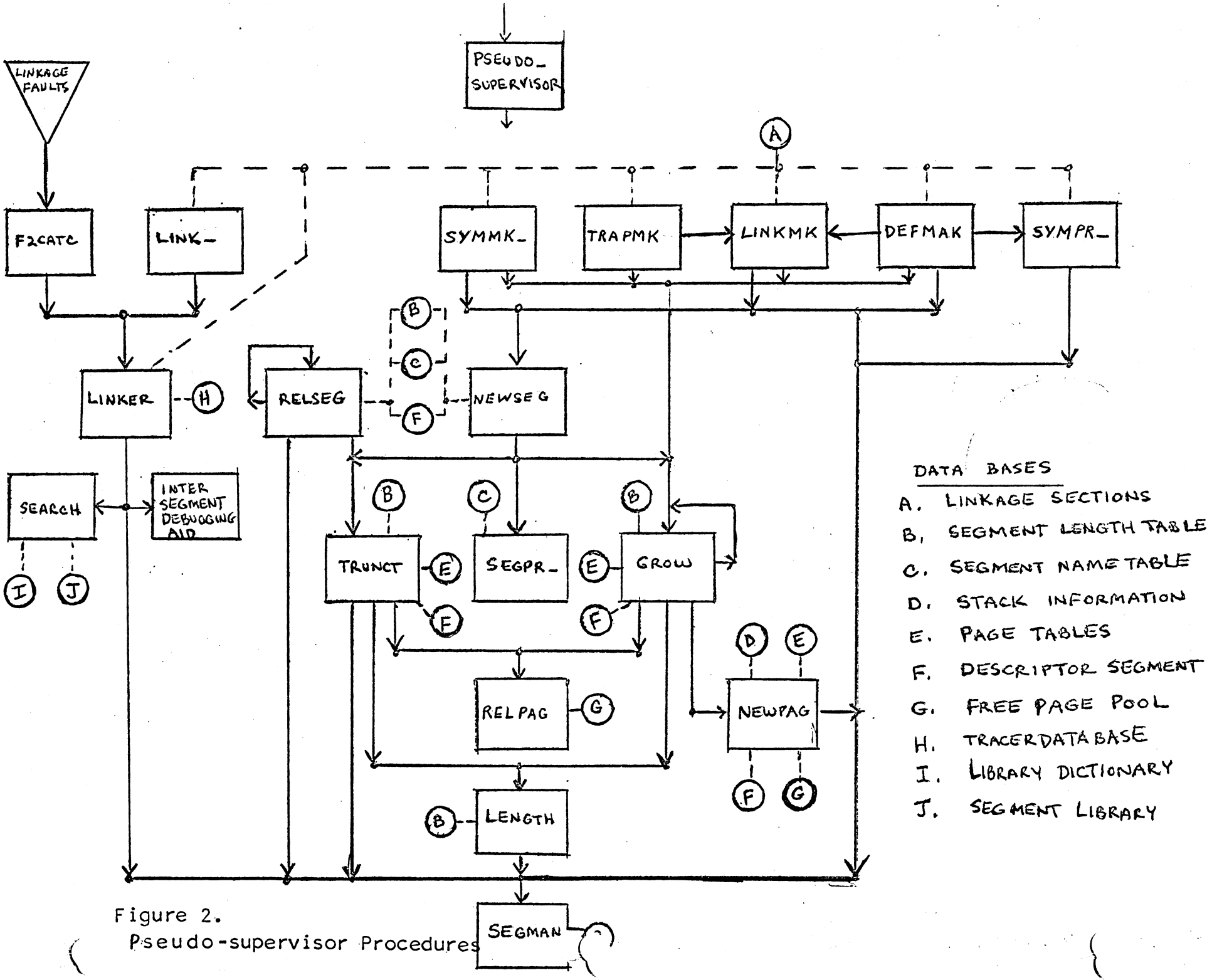

Figure 2.
Pseudo-supervisor Procedures