Published: 06/09/67

Identification

Overview of the pre-Multics System
G. S. Stoller

0. Purpose

This system allows one to compile (with EPL) and assemble (with EPLBSA) one or more MULTICS segments, and to run a pseudo-process. All of this is done on a GE645 or GE635 under GECOS.

This system resides on the 645.LIBRARY tape; its input is generated either by the 6.36 system (see BE.5.02) or the 64.5 system (see BE.6.01).

1. Composition of a Job

The GECOS jobstream in this system is composed as follows:

a.	Initialization activity	BE.7.01
b.	EPL compilation activities (to object code)	BE.2 (and BN)
c.	EPLBSA assembly activities (to object code)	BE.7.04 (and BN.8)
d.	PACKER, MARKER activity (normally present)	BE.7.05, BE.7.06
e.	Pseudo-process activity (only if a pseudo-process is to be run)	BE.7 : .07,.08,.10,.12,.13
f.	Return Tape activity (only for jobs made on 6.36 system)	BE.5.05
g.	64.5 Dumper activity (only if a pseudo-process has been run and a 64.5 Dumper core dump is requested, or a binary deck from a EPLBSA assembly is to be punched.)	BE.6.02

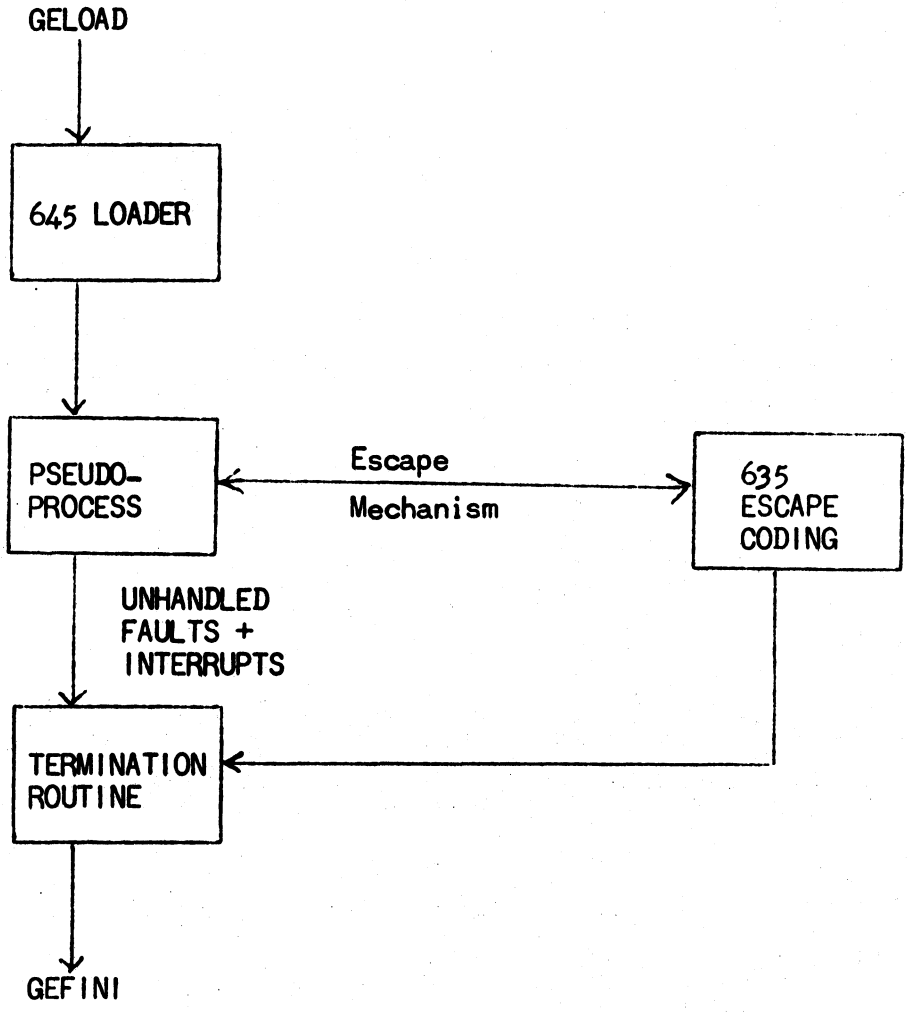
A GECOS job consists of no more than 35 activities.

2. EPL Compilation (to object code)

An EPL compilation (to object code) consists of several activities. There is one activity per pass of EPL and one activity for an EPLBSA assembly.

3. Pseudo-Process activity

This activity consists of two logically distinct parts. Its raison-de-etre is the pseudo-process part; the other part is called the "635 support package" and is made up of 635 subroutines that support the pseudo-process.



Schematic of the pseudo-process activity

All but the pseudo-process are 635 subroutines. They are loaded and bound by GELOAD, and all faults that they generate are passed to GECOS.

3.1 635 Support Package

A subset of the 635 support package is called 635 escape coding. These subroutines are called upon by the pseudo-process to interface with GECOS. A subset of the pseudo-process interfaces with these subroutines; it is called the pseudo-process escape coding. Together they compose the escape coding.

The 635 support package acts as an extension of GECOS for the pseudo-process. (On the other hand, the simulator acts as hardware for the pseudo-process.) Its main functions are:

(a)	Loading segments and initializing,	BE.7.07, BE.7.12, BE.8.05
(b)	Interfacing with GECOS, (to provide I/O for the pseudo-process)	BE.7.10
(c)	Providing a clean exit from the pseudo-process activity.	BE.7.10, BE.7.08, BE.7.12

3.2 <memory>

This segment is present in every pseudo-process run. A fixed segment number (5) is reserved for this segment. The communication area between the pseudo-process and its support package is set up in <memory>.

<memory> is an unpaged segment consisting of 1024 word blocks. It is a sub-space of the 635 support package; i.e., its address space is a subset (with the same contents in each location) of the address space of the 635 support package. When the pseudo-process is being simulated, it is a proper subspace (i.e., smaller). When the pseudo-process is being executed, it is an improper subspace (i.e., equal).

<memory>'s SDW holds a DF3 (directed fault 3) while the pseudo-process is running; but it is SLVPRC, SLVACC, WPERMT ("slave-procedure, slave-access, write-permit") when the 635 support package is running. Segment <escape> flips the SDW back and forth. See BE.7.10 for details.

3.3 Pseudo-Process Environment

Major differences exist between the simulation environment of a pseudo-process and the execution environment of a pseudo-process. These are:

3.3.1. instruction-by-instruction trace

Availability of an instruction-by-instruction trace of the pseudo-process under simulation (in effect, a trace by hardware). This trace output is provided on the 645 printer only; it is not put on the return tape.

3.3.2. Clean exit guarantee

Under simulation, a clean exit from the pseudo-process activity is guaranteed.

3.3.2.1. Core memory configuration

Under simulation of the pseudo-process, there is a totally different configuration of core memory than exists under execution of the pseudo-process. Under execution, all of 645 core memory is available to the pseudo-process, it need only insert appropriate SDW's in <dseg>. Under simulation, only the memory allocated to the pseudo-process by the 645 loader exists; any references outside of this area result in an "op-not-complete" fault; thus, the pseudo-process cannot alter the support package, GECOS, or the supplement.

3.3.2.2. Pseudo-Process Loop Problem

If the pseudo-process gets into the loop, the simulator can terminate it when "TIME" runs out. Under execution, only a GECOS or RAID termination is provided.

3.3.3. Pseudo-Process: Running Time and Hardware Checkout

The pseudo-process running time under simulation is a multiple (by a factor of > 100) of its running time under execution.

Under simulation, the 645 hardware is not being tested.

Reason (3.3.3) dictates running the pseudo-process under execution at all times unless the guarantee of a clean exit from the pseudo-process activity is not provided under execution, or an instruction-by-instruction trace is needed.