Identification

Shutting Down Multics After a System Crash
emergency_shutdown
N. I. Morris

Purpose

The design of the Multics Salvager can be greatly simplified
if the Salvager need not concern itself with the contents
of core storage after a system crash.　This can be done
if one is willing to abandon those pages which were in
core at the time of the crash and the latest copy of the
storage map, also in core at the time of the crash.　This
is not a satisfactory approach due to the large amount
of valuable information which will be lost.

The emergency_shutdown procedure is designed to re-enter
the Multics system after a system crash.　It will attempt
to gracefully shut the system down and thus salvage the
contents of core.　emergency_shutdown is designed to run
in a series of steps, each step relying on the integrity
of an increasingly larger portion of the Multics system.
A failure at any step of emergency_shutdown should have
no effect on the success of any previous step.

Implementation

The segment "emergency_shutdown" will be accessable in
all rings as a master mode procedure.　It may be entered
at any time by manually transferring to <emergency_shutdown>|10.
(10 is the first location after the master mode prologue.
Note that emergency_shutdown cannot be called by a slave
procedure.)　emergency_shutdown will perform the following
steps in an attempt to shut the system down:

1.　Establish a dbr value, pair base registers, and
establish an lp value.　The dbr value will be that of
ring 0 of any user process.　The hardcore processes may
not be used since they are incapable of supporting
page faults and they don't have a ring 0 pageable stack.

2.　Mask all interrupts.　Disable the traffic controller
and interrupt interceptor by turning off
tc_data$wait_enable.　Turn on ilock$ilock_ignore to
ignore the presence of locked data bases.　Unlock the
global page table lock.

3.   Establish a stack frame at the base of pds and set sp.

4.   Call free_store_epl$update to write out the rootbranch
     and the free storage map.

5.   Call core_man$flush to page out all of unwired core.

6.   Establish a stack frame at the base of stack_00 and set sp.
     The full paging mechanism must be intact in order to
     succeed at this step. pds must be abandoned at this point
     since the following steps of emergency_shutdown will invoke
     major portions of segment and directory control. The
     pds is not large enough to accommodate these procedures.

7.   Call shutdown$emergency. This will attempt to deactivate
     all user segments and then all hardcore segments as in
     a normal shutdown. (see MSPM Section BM.9). Note that
     most of segment control must be operational in order to
     accomplish this step. Processes will not be deleted and
     process directories will remain after this step.

8.   shutdown will finally call master_mode_ut$bos to enter
     BOS, completing the emergency shutdown.

Note that even if segments cannot be successfully deactivated,
all pages in core will have been written out. Thus certain
items in directories may not be completely up-to-date.
This, however, is non-fatal. The salvager will later
attempt to correct any inconsistencies in directory entries.