Identification

stop_proc
Michael J. Spier

Purpose

This section defines the overall strategy applied when
stopping a process. A "stopped" process does not, basically,
differ from a "blocked" process; both processes have given
their processors away for an undetermined length of time.
However, the process that calls block does so knowing
that some other process will wake it up, whereas a stopped
process is more often than not "doomed", on its way to
saving or destruction, and has been made to put itself
in the stopped state by another process (e.g., the universal
overseer process).

In stopping a process, we distinguish between two states
of execution for that process:

1.   The process is executing in behalf of the user.

2.   The process is executing in behalf of the system.

A process must be stopped in a way such as to insure that
no damage will be inflicted on the system as a result
of that action.

Discussion

When a process is to be stopped, it is in either one of
the two above-mentioned states of execution. Without,
at this point, going into a precise definition of those
states, the stopping policy is as follows:

1.   A process that is executing in behalf of the user
      is interrupted and made to call pxss$i_stop, which
      puts it in a "stopped" state and gives the processor
      away.

2.   A process that is executing in behalf of the
      system is allowed to "run itself out" of that
      state; it is forced to stop itself as soon as
      it executes in behalf of the user.

By convention, a process is said to be executing in behalf
of the system whenever it executes in the hardcore ring.

Implementation

When process A wants to quit process B it calls

        hcs_$stop_proc(B)

which, after validation of the call invokes pxss$stop(B) which in turn sends a stop interrupt to process B.

Process B gets interrupted (provided that it executes outside of ring 0), and the interrupt handler calls pxss$i_stop to put the process in a stopped state and give the processor away.

This strategy insures that a stopped process has always a standard and reconstructable ring 0 stack history.

A stopped process can be restarted by calling

        hcs_$start_proc(B)

which sets that process' wakeup switch to "on" and calls for it pxss$start(B) to put it in a "ready" state.