Published: 10/01/68

Identification

The Idle Process
Robert L. Rappaport, Michael J. Spier

Purpose

For reasons which are implementation dependent, processors are never temporarily stopped; moreover, processors must always execute in Multics processes (in other words, Multics does not tolerate the independent execution of a processor). As it is not guaranteed that there will always (at every discrete instant of time) be useful work for a processor to do, the Traffic Controller maintains for every processor in the system a "dummy" process whose purpose it is to soak up all surplus processor capacity. Such a process is known as an "idle process".

Discussion

The way the Traffic Controller is implemented, whenever a process abandons its processor, it gives that processor to the topmost eligible process on the ready list. In this implementation it is imperative that there always be some eligible process on the ready list, an "empty" ready list cannot be tolerated.

Taking advantage of Multics' multi-level scheduler, the following has been implemented: Just as priority level 1 is reserved for system processes, so is the lowest-priority level (tc_data$lowest_level) reserved for the exclusive use of the idle processes. It is guaranteed that no other process will ever be assigned a level number of equal or lower priority. Also, idle processes enjoy eternal eligibility. Consequently, whenever an idle process gets scheduled, it is automatically placed by the scheduler at the tail of the ready list. This insures that the ready list be never empty, as well as that an idle process be made to run only if no other process in the system is currently capable of running.

The idle process' function is to keep the processor "idling" until some other process is ready to run. The fact that some other process is now capable of running can be determined in the ready list only. Consequently, the idle process should loop around examining the ready list. This can be accomplished by the following program:

```
idle_process:  call reschedule;

               go to idle_process.
```

By calling the Traffic Controller entry point reschedule, the idle process gives up its processor and reschedules itself at the tail of the ready list. The only way in which it can be made to run again is by being the first eligible process on the ready list, or in other words when there is nothing else to do in the system.

It has been observed that the idle process is often made to run because all other processes are unable to proceed, waiting for pages to come into core. Therefore, for reasons of efficiency, before calling entry point reschedule the idle process invokes entry point device_control$run which services page faults.

It is also conceivable that the idle process might invoke some metering mechanism to determine the amount of idle processor time in the system.

In the above-mentioned scheme, an idle process need not be associated with any specific processor. All that is needed is to have as many idle processes as there are processors executing in the system. However, the implementation is such that every idle process is actually associated with one specific processor (in the idle process' APT entry, item processor_required=processor number). This insures that idle process p' will be chosen to execute on processor p only. This organization allows for possible control over specific processors. For example, if in a multi-processor system we wish to halt the execution of processor p without affecting the system, we can make idle process p' execute a DIS instruction, knowing that sooner or later processor p (and only processor p) will execute in that process.