Vincent Maggioli

Lab 3 Answers

3(f)

Under nonuniform process priorities, XINU has potential to starve out a process. I tested multiple scenarios that all had one thing in common – one process had a far lower priority than the rest. Whether the higher processes were equivalent priority, even priority, or a mix of the two, the lowest priority process always executed last. With they hypothetical scenario that processes with a high priority keep getting made while the other high priority processes are executing, a low priority or an intermittently sleeping process can easily get starved.

4.3

CPU Bound Only

CPU bound processes will run the full length of the *QUANTUM* and then be slated for a context switch. For every 100 milliseconds, not only is each process only running for 20 milliseconds, but each process is also only running once per 100 milliseconds. Therefore, fair scheduling is being satisfied.

IO Bound Only

Unlike CPU processes these processes are not running to the entirety of *QUANTUM*. In a time frame of 25 milliseconds, each process runs once and is running for only 5 milliseconds. Since the processes are running evenly in a time frame, and are running in the same order for each time frame, fair scheduling is achieved.

CPU and IO Bound

Equal sharing between three CPU bound and three IO bound processes is being achieved. All the IO processes wound up with the same priorities, as did the CPU processes, and all the like processes finished within a few cycles of each other. Not only did the IO processes take up SIGNIFICANTLY less CPU time (367 vs. 613 where UP1=10 and UP2=10, so small values), but they finished much sooner as well. Finishing sooner the CPU processes is essentially entirely due to the minimal amount of CPU time they need to perform their operations. This is why the common strategy in operating systems is to give IO processes priority over CPU processes. The IO processes also had a shorter preempt overall than the CPU bound processes.

Vincent Maggioli

Bonus

For this implementation I would effectively split the priority range in half. Whatever MAXPRIO/2 – 1 is, that and below will be the only priorities that a TS process can have. If a RT process is made, it starts with the actual MAXPRIO – 1 value. As far as checking if the user is root, you can check the process field prparent to see if it is 0 (the nulluser). If they aren't the null user, then XINU should not allow a process with priority above MAXPRIO/2 – 1 to be created. This implementation also assumes that we are keeping the assumption that any process's total CPU total run time will not be above MAXPRIO's maximum value.