6.10 They are dependent on different things and thus utilize the resources in a different way. Having 100% uptime on the processor would mean prioritizing CPU bound processes, running the most amount of processes would mean prioritizing I/O bound programs.

6.11 Increasing CPU utilization would mean lowering response time due to the lower amount of context switching.

Increasing turnaround time would most likely cause the average waiting time to increase because the programs with the shortest processing time would be queued before the longer processes.

I/O device utilization and CPU utilization are maximized by two differently bound programs. Decreasing one would inherently increase the other.

6.16

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | FCFS | RR | SJF | Priority |
| P1 |  |  |  |  |
| P2 |  |  |  |  |
| P3 |  |  |  |  |
| P4 |  |  |  |  |
| P5 |  |  |  |  |

6.21 91%

94%

5.3 Busy waiting is when a process is waiting or blocking for some reason but it is inside of an (almost) empty loop, meaning it does nothing while keeping control of the cpu. The process could give up control and when it was rescued check for whatever it is waiting for.

5.4 The only way to leave a spin lock would be to have another process change something that would knock you out. On a single processor the process would be in the spin lock and not give up control of the CPU.

5.10 Disabling interrupts could have unforeseen malicious uses. A process could disable interrupts and then never get kicked off the CPU when its quantum expired.

5.11 On a multi-processor system you would need to disable interrupts on all of the processors for it to have the desired effect.

5.23

int lock = 0;

int semVal = 0;

wait()

{

while (TestAndSet(&lock) == 1);

if (semVal == 0) {

processWaiting.add();

} else {

semVal--;

lock = 0;

}

}

signal()

{

while (TestAndSet(&lock) == 1);

if (semVal == 0 && processWaiting)

processWaiting.pop();

else

semVal++;

lock = 0;

}

5.28 You could increase throughput by giving multiple readers higher priorities than writers. This could cause some problems with the writers never getting a chance to write if there are a lot of readers. This could be somewhat mitigated by having an increasing priority based on how long someone has been waiting.

5.29 Once signals are sent if there is nothing there to acknowledge them then they are simple forgotten, while semaphores don’t send a signal out they increment a counter so that even if nothing is listening when the semaphore is used if at some point it is going to be accessed again the other process will know about each other.

5.32