SNEHAL KUMAR
2019101003

OS NW

QUIZ - 4

Q.1. Suppose that a multiprogrammed system has a load of N processes with individual total execution times.

A: A multi programmed system, if it has multi cores can execute the processes in time $T > \max(t_1, t_2, t_3 \dots t_n)$. If one process is involved in I/O execution, then reading input, use CPU, finish the process. By alternating among the I/O and CPU using a multilevel feedback and hyperthreading on the cores will enable each process to run simultaneously while the process with highest execution time will be given highest priority and will continue execution. Thus if we decide priorities based on execution times and hyperthreading these processes, we can use multiprogramming and adequately execute all processes in time $T > \max(t_1, t_2, t_3 \dots t_N)$ through concurrent processes. One of the largest process & will go between I/O and CPU bursts while other smaller process will use the CPU and I/O when the largest isn't using it.

Q.2 Identify whether the following statement is TRUE or FALSE. If the statement

A: The statement is false. In FCFS scheduling algorithm suffers from convoy effect since the first ~~ass~~ process to enter could be a heavy CPU bound process and if other I/O bound process come after, they would have to wait long times before CPU bound process is completed. SJF doesn't suffer as it schedules fastest process for execution.

Q.3. The kernel of a multiprogramming system classifies a program as CPU bound or I/O bound

A: In case programs are wrongly classified, the scheduling algorithms will cause major problems for processors.
Typically interactive processes are given good response times and CPU bound processes are given good throughput.
If I/O process is given good throughput instead, it will be given lower priority lower time slice and will thus take a longer time to execute which is not desirable in real time. Similarly CPU bound will be given higher priority instead.
This kind of unbalance will effectively slow the average waiting time of processes greatly and might even cause load unbalance and starvation of processes. In a multiprogramming system, this can be fatal since number of context switches can also incur lot of overhead.

Q.4. Why preemption points are included to system calls?

A: In an operating system, it is critical for processes to be executed in real-time. For this reason, some processes are given higher priorities. ~~and~~ When these process, typically I/O processes, are have when called through system calls, have preemption points to interrupt current running tasks and execute them. If kernels can process tasks on behalf of other processes, and thus if preempted in between it can cause chaos and thus requires mutex locks to prevent this.
~~If kernel is non preemptive, kernel will execute~~
When a low priority process makes a system call, high priority process will wait until the system call completes.