Chapter 5 Questions

Questions may be asked in class, on quizzes, and on exams

* + Briefly explain the difference between preemptive and non-preemptive scheduling

Preemptive : Process from Run to Ready, from Wait to Ready

Non-preemptive : Process from Run to wait, Process terminated

**preemptive scheduling** takes place when a process runs from running state to ready state

**non**-**preemptive scheduling** takes place when the process is terminating

* + What actions does the dispatcher need to conduct? Why?

Switch context from one process to another. Switch to user mode. Jump to the user program to resume that program.

* + Define the terms: CPU utilization, throughput, turnaround time, waiting time, and response time

CPU utilization : From 0 to 100%, measure of how CPU works

Throughput : When CPU is busy executing processes, throughput is the measure of number of process completed per time unit.

Turnaround time : Sum of waiting in ready queue, executing on CPU and doing I/O

Waiting time : Period spend on waiting in ready queue.

Response time : Time of starting response after request.

* + Explain how FCFS scheduling works, identify the benefit of using FCFS, and identify a problem with using FCFS

CPU would execute the first process come and won’t switch to another one until the first one is executed. Benefit : Easy to write and understand. Problem : Cost much time especially when the CPU get a really busy process, it would just waste the time of other processes waiting.

* + Explain how SJF scheduling works, identify the benefit of using SJF, and identify a problem with using SJF

When CPU is available, it is assigned to the process with shortest CPU burst.

Benefit : Save much wasting time for those processes which are not in execution.

Problem : Can’t know the exact length of next process CPU burst.

* + Explain how SJF would work if it was configured as a preemptive strategy

It may cause some troubles. When a process with shortest CPU burst comes to Ready, though the previous one was working, CPU would stop executing the current process in CPU and let the new-come one in and execute it.

* + Explain how RR scheduling works, identify the benefit of using RR, and identify a problem with using RR

RR is similar with FCFS, it has a time slice which is a small time interval. First, we treat ready queue as FIFO queue, allocate each process time slice, if the process can be executed in a time slice, the process would release CPU, or the process’s CPU burst is longer than time slice, it would be interrupted to next after being executed for a time slice.

Benefit : It dramatically improve the response time. Just a time slice.

Problems : If the time is big enough, the first process would wait a really really long time to be executed.

* + Identify and explain the key difficulty with attempting to implement SJF

You can’t know the length of next process exactly. To implement SJF, we need to approximately estimate the length of next process.

* + Briefly explain how Round Robin scheduling is different than FCFS

RR has a timer to interrupt process to next process. RR is preemptive while FCFS is non-preemptive.

* + Briefly explain how Round Robin scheduling is the same as FCFS

RR would pick the first process of the ready queue to execute, then to the second and so on. Which is the same order as FCFS. Additionally, if time is big enough, they are the same.

* + Identify and briefly explain the benefits and problems with using a priority-based scheduling system

Benefits : The order of priority is defined by you, which means you could choose which kind to be executed first. As the number of open files, response time and so on.

Problems : Infinite-blocking. Starvation. Due to the low-priority, some process may be delayed forever.

* + Identify and briefly explain the value of a multilevel queue scheduling strategy

Process would be distributed to different queue depends on its character. Different queue has different priority.

* + Briefly explain what memory stall is and how the use of multicore processors may suffer from memory stall or how it might effectively manage memory stall

Memory stall : When a processor access the memory, it would spend a significant time waiting for the data to be available.

Processors operate at a much faster speed than memory. They use multithreaded processing cores in which two (or more ) hardware threads are implemented to each core. Threads would prepare data in memory before processers access.

* + Identify and briefly explain the difference between coarse and fine grained multithreading

Coarse grained : the cost of switching is high while the cost of fine grained is small.

Coarse grained, when the delay happens due to some long-latency events, core needs to switch to another thread. Coarse grained : the instruction pipeline must be cleaned before other threads get to be executed in processor core. When the new thread begins execution, it would fill the pipeline with new instructions.

Fine grained : when core is switching, it doesn’t need to clean the whole pipeline, it could stand more fine grain than coarse grained. The cost is lower.

* + End of Chapter 5
    - Exercises: 5.3, 5.4, 5.5, 5.6, 5.8, 5.11, 5.12, 5.17, 5.18