COSC1112/1114: Operating Systems Principles

Tutorial 04 (week 05)

1. **Define the difference between preemptive and nonpreemptive scheduling.**

Answer:
Preemptive scheduling allows a process to be interrupted in the midst of its execution, taking the CPU away and allocating it to another process. Nonpreemptive scheduling ensures that a process relinquishes control of the CPU only when it finishes with its current CPU burst.

2. **Why is it important for the scheduler to distinguish I/O-bound programs from CPU-bound programs?**

Answer:
I/O-bound programs have the property of performing only a small amount of computation before performing I/O. Such programs typically do not use up their entire CPU quantum. CPU-bound programs, on the other hand, use their entire quantum without performing any blocking I/O operations. Consequently, one could make better use of the computers resources by giving higher priority to I/O-bound programs and allow them to execute ahead of the CPU-bound programs.

3. **Discuss how the following pairs of scheduling criteria conflict in certain settings.**
   a) **CPU utilization and response time**
   b) **Average turnaround time and maximum waiting time**
   c) **I/O device utilization and CPU utilization**

Answer:
a) CPU utilization is increased if the overhead associated with context switching is minimized. The context switching overheads could be lowered by performing context switches infrequently. This could, however, result in increasing the response time for processes.
b) Average turnaround time is minimized by executing the shortest job first. Such a scheduling policy could, however, starve long-running tasks and thereby increase their waiting time.
c) CPU utilization is maximized by running long-running CPU-bound tasks without performing context switches. I/O device utilization is maximized by scheduling I/O-bound jobs as soon as they become ready to run, thereby incurring the overheads of context switches.

4. Consider the following set of processes, with the length of the CPU-burst time given in milliseconds:

process	Burst Time	Priority
P1	**10**	**3**
P2	**1**	**1**
P3	**2**	**3**
P4	**1**	**4**
P5	**5**	**2**

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.

a) **Draw four Gantt charts illustrating the execution of these processes using FCFS, SJF, a nonpreemptive priority (a smaller priority number implies a higher priority), and RR (*quantum* = 1) scheduling.**

b) **What is the turnaround time of each process for each of the scheduling algorithms in question a?**

c) **What is the waiting time of each process for each of the scheduling algorithms in question a?**

d) **Which of the schedules in question a results in the minimal average waiting time (over all processes)?**

Answer:

a) The four Gantt charts are

FCFS

1	2	3	4	5

RR

1	2	3	4	5	1	3	5	1	5	1	5	1	5	1	1	1	1	1

SJF

2	4	3	5	1

Priority

2	5	1	3	4

b) Turnaround time

	FCFS	RR	SJF	Priority
P1	10	19	19	16
P2	11	2	1	1
P3	13	7	4	18
P4	14	4	2	19
P5	19	14	9	6

c) Waiting time (turnaround time minus burst time)

	FCFS	RR	SJF	Priority
P1	0	9	9	6
P2	10	1	0	0
P3	11	5	2	16
P4	13	3	1	18
P5	14	9	4	1

d) Shortest Job First

5. Consider a variant of the RR scheduling algorithm where the entries in the ready queue are pointers to the PCBs

a) **What would be the effect of putting two pointers to the same process in the ready queue?**

b) **What would be the major advantages and disadvantages of this scheme?**

Answer:

a) In effect, that process will have increased its priority since by getting time more often it is receiving preferential treatment.

b) The advantage is that more important jobs could be given more time, in other words, higher priority in treatment. The consequence, of course, is that shorter jobs will suffer.

6. Explain the differences in the degree to which the following scheduling algorithms discriminate in favor of short processes:

a) **FCFS**

b) **RR**

Answer:

a) FCFS — discriminates against short jobs since any short jobs arriving after long jobs will have a longer waiting time.

b) RR — treats all jobs equally (giving them equal bursts of CPU time) so short jobs will be able to leave the system faster since they will finish first.