Solutions to exercises on Scheduling Spring 2022

1. Explain how the following pairs of scheduling criteria can conflict:

   (a) Maximizing CPU utilization while minimizing response time

   sol: CPU utilization is increased if the overheads 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) Minimizing average turnaround time while minimizing waiting time

   sol: Average turnaround time is minimized by executing the shortest tasks first. Such a scheduling policy could, however, starve long-running tasks and thereby increase their waiting time.

   (c) Maximizing I/O device utilization and maximizing CPU utilization

   sol: 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.

2. "Lottery scheduling" is a scheduling algorithm that works by assigning lottery tickets to threads, which are used for allocating CPU time. Whenever the scheduler needs to schedule a thread, a lottery ticket is chosen at random, and the thread holding that ticket gets the CPU. Assume we have a scheduler that implements the lottery scheduling by holding a lottery 50 times each second, with each lottery winner getting 20 milliseconds of CPU time (20 milliseconds $\times$ 50 = 1 second). Describe how this scheduler can ensure that higher-priority threads receive more attention from the CPU than lower-priority threads.

   sol: By assigning more lottery tickets to higher-priority processes.

3. A variation of the round-robin scheduling algorithm is the "regressive round-robin" algorithm This algorithm assigns each thread a time quantum and a priority. The initial value of a time quantum is 50 milliseconds. However, every time a thread has been allocated the CPU and uses its entire time quantum (does not block for I/O), 10 milliseconds is added to its time quantum, and its priority level is boosted. (The time quantum for a process can be increased to a maximum of 100 milliseconds.) When a process blocks before using its entire time quantum, its time quantum is reduced by 5 milliseconds, but its priority remains the same. What type of thread (CPU-bound or I/O-bound) does the regressive round-robin scheduling algorithm favor? Explain.

   sol: favor CPU-bound threads as they are rewarded with a longer time quantum as well as priority boost whenever they consume an entire time quantum. This scheduler does not penalize I/O-bound processes as they are likely to block for I/O before consuming their entire time quantum, but their priority remains the same.

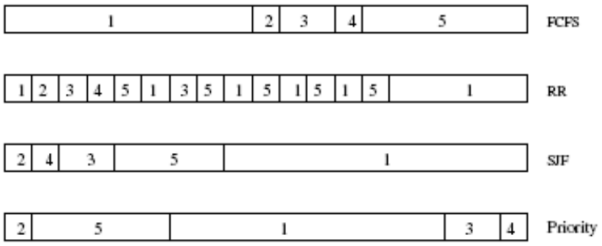
4. Consider the following set of processes, with the length of the CPU burst time given in milliseconds

Process	Burst Time	Priority
P_1	2	2
P_2	1	1
P_3	8	4
P_4	4	2
P_5	5	3

The processes are assumed to have arrived in the order P_1, P_2, P_3, P_4, P_5 all at time 0.

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

sol:



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

	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) What is the waiting time of each process for each of these scheduling algorithms?

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) Which of the algorithms results in the minimum average waiting time (over all processes)?

sol: Shortest Job First

5. Which of the following scheduling algorithms could result in starvation? a) First-come, first-served; b) Shortest job first; c) Round robin; d) Priority

   sol: Shortest job first and priority-based scheduling algorithms could result in starvation.

6. Consider a preemptive priority scheduling algorithm based on dynamically changing priorities. Larger priority numbers imply higher priority. When a process is waiting for the CPU (in the ready queue, but not running), its priority changes at a rate α; when it is running, its priority changes at a rate β. All processes are given a priority of 0 when they enter the ready queue. The parameters α and β can be set to give many different scheduling algorithms.

   (a) What is the algorithm that results from $\beta > \alpha > 0$? sol: FCFS

   (b) What is the algorithm that results from $\alpha < \beta < 0$? LIFO

7. Explain the differences in how much the following scheduling algorithms discriminate in favor or against short processes (sum of CPU and I?O bursts is short):

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

   (b) RR: sol: 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.

   (c) Multilevel feedback queues: sol: Multilevel feedback queues work similar to the RR algorithm they discriminate favorably toward short jobs.

8. Consider the scheduling algorithm in the Solaris operating system for time-sharing threads:

   (a) What is the time quantum (in milliseconds) for a thread with priority 10? With priority 55? sol: 160 and 40

(b) Assume a thread with priority 35 has used its entire time quantum without blocking. What new priority will the scheduler assign this thread? sol: 35

(c) Assume a thread with priority 35 blocks for I/O before its time quantum has expired. What new priority will the scheduler assign this thread? sol: 54

9. The following processes are being scheduled using a preemptive, round-robin scheduling algorithm. Each process is assigned a numerical priority, with a higher number indicating a higher priority. In addition to the processes listed in the table below, the system has also an "idle process" identified as P_{idle} which consumes no CPU resources. Process P_{idle} has priority 0 and is scheduled whenever the system has no other available process to run.

   The length of a time quantum is 10 milliseconds. If a process is preempted by a higher priority process, the preempted process is placed at the end of the queue. Burst times and arrival times in the table below are expressed in milliseconds.

Process	Priority	Burst	Arrival
P_1	40	20	0
P_2	30	25	25
P_3	30	25	30
P_4	35	15	60
P_5	5	10	100
P_6	10	10	105

   (a) Show the scheduling order of the processes using the table below. On the first row of this table write the identity of the process (for example P_i) and under process P_i on the second row, write the time the process has been de-scheduled or has completed. For example, if process P_i starts at time 0 and is de-schduled after 10ms, write $\begin{array}{|c|}\hline P_i \\ \hline 10 \\ \hline \end{array}$. If P_j starts at 55ms and complete at 60ms write $\begin{array}{|c|}\hline P_j \\ \hline 60 \\ \hline \end{array}$

P1	P1	idle	P2	P3	P2	P3	P4	P4	P2	P3	idle	P5	P6	P5
10	20	25	35	45	55	60	70	75	80	90	100	105	115	120

   (b) What is the average turnaround time of the processes (not including P_{idle})? (Turnaround is finishing time - arrival time)

   P1=20; P2 = 80-25 = 55; P3 = 90-30=60; P4 = 75-60=15; P5=20; P6 = 10 = 180/6 = 30

   (c) What is the average waiting time of the process (not including P_{idle})? (finishing time - running time - arrival time)

   P1=0; P2 = 80-25-25=30; P3=90-30-25=35; P4=0; P5=120-100-10=10; P6=0 = 75/6 = 12.5

10. For each sub-question choose a subset of correct answers. For example, in 1), is $\{b, c\}$ correct or is $\{a, b\}$ correct or just $\{a\}$ is correct or none of them $\{\emptyset\}$ is correct, etc.

(a) Possible next states for a process in the Ready state a) New, b) Ready, c) Running, d) Blocked, e)Terminated

Sol: a

(b) Possible next states for a process in the Running state

a) New, b) Ready, c) Running, d) Blocked, e)Terminated

Sol: b, d, e

11. Solaris like multilevel feedback queue scheduling. The table below describes a 5 levels feedback queue similar to the Solaris feedback queue for time-sharing and interactive threads, where 0 is the lowest priority. In this table "Time quantum expired" is the new priority of a process that did not complete its current CPU burst in the allocated time quantum of the queue, while "Return from I/O" is the new priority of a process that enters an I/O phase before the end of the allocated time quantum of the queue.

Priority	Quantum time	Time quantum expired	Return from I/O
0	10	0	2
1	8	0	2
2	6	0	3
3	4	1	4
4	2	1	4

We have 3 processes P_0, P_1 and P_2 which have execution cycles as described below. For example, the arrival time of P_2 is 2, its first CPU burst last 5ms, then enters into an I/O queue for 3ms, then a CPU burst of 8ms, an I/O of 4ms, CPU burst of 4ms and then exit.

P_0	CPU	I/O	CPU	exit
0	2	6	2	

P_1	CPU	I/O	CPU	exit
1	8	4	4	

P_2	CPU	I/O	CPU	I/O	CPU	exit
2	5	3	8	4	4	

There is one CPU. The possible **states** of a process are the following: 1) ready to execute: "RY-x" where x is the priority queue in which the process is waiting; 2) running: "RU-x" where x is the priority queue in which the process was before been scheduled to run; 3) waiting in an I/O queue: "I/O-x" where x refers to the priority queue in which the process will be placed after completing its I/O; 4) exited: "EX"

All the processes start in priority queue 2. Arrival times of process P_0 is 0, process P_1 is 1 and process P_2 is 2. Fill the table below. For each column of current time and for each process, write the state of the corresponding process. For example, at current time 3, P_0 = I/O-3; P_1 = RU-2; and P_2 = RY-2.

Note, the scheduler always empty first the queue of higher priority before it starts to run processes in the next lower priority queue. However, the scheduler never preempt a currently running lower priority process.

	3	9	11	16	18	19	22	23	27	33	37
P_0	I/O-3	RU-3	EX	EX	EX	EX	EX	EX	EX	EX	EX
P_1	RU-2	RY-0	RY-0	RU-0	I/O-2	I/O-2	RY-2	RU-2	EX	EX	EX
P_2	RY-2	RY-2	RU-2	I/O-3	I/O-3	RU-3	RU-3	RY-1	RU-1	I/O-2	RU-2