Exercise 4 – CPU Scheduling

Questions are taken from Stallings, Operating Systems Internals and Design Principles, fifth edition and Silberschatz et al., Operating System Concepts, seventh edition.

1 – Silberschatz 5.4

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

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

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, non-preemptive priority (a smaller priority number implies a higher priority), and RR (quantum = 1).

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

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

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

2

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

Process	Burst Time	Arrival time
P_1	8	0
P_2	4	0.4
P_3	1	1

a) Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, Clairvoyant SJF (the algorithm can look into the future and wait for a shorter process that will arrive).

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

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

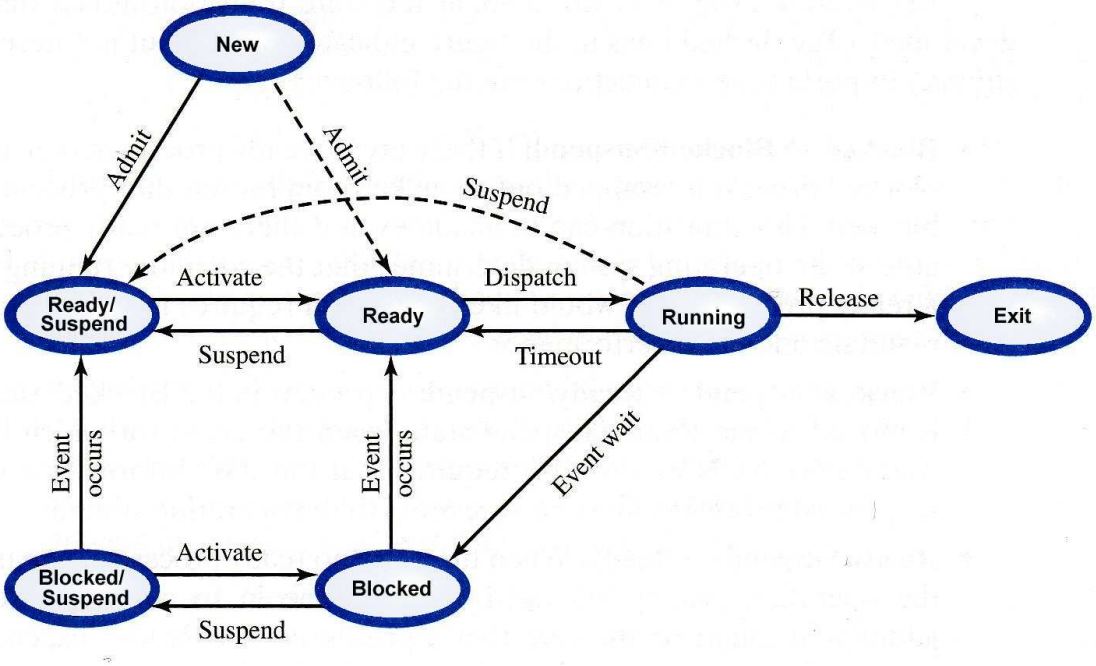
d) Which of the algorithms om part a results in the minimum average waiting time (over all processes)?

3 – Stallings 3.4

Consider the state transition diagram in the figure. Suppose that it is time for the operating system to dispatch a process and that there are processes in both the Ready state and the Ready/Suspend state, and that at least one process in the Ready/Suspend state has higher scheduling priority than any of the processes in the Ready state. Two extreme policies are:

1. Always dispatch from a process in the Ready state, to minimize swapping.
2. Always give preference to the highest-priority process, even though that may mean swapping when swapping is not necessary.

Suggest an intermediate policy that tries to balance the concerns of priority and performance.



- Ready: The process is in main memory and available for execution.
- Blocked: The process is in main memory and awaiting an event.
- Blocked/Suspend: The process is in secondary memory and awaiting an event.
- Ready/Suspend: The process is in secondary memory but is available for execution as soon as it is loaded into main memory.

4 – Stallings 3.9

In a number of early computers, an interrupt caused the register values to be stored in fixed locations associated with the given interrupt signal. Under what circumstances is this a practical technique? Explain why it is inconvenient in general.

5 – Stallings 4.7

A multiprocessor with eight processors has 20 attached tape drives. There is a large number of jobs submitted to the system that each require a maximum of four tape drives to complete execution. Assume that each job starts running with only three tape drives for a long period before requiring the fourth tape drive for a short period toward the end of its operation. Also assume an endless supply of such jobs.

a) Assume the scheduler in the OS will not start a job unless there are four tape drives available. When a job is started, four drives are assigned immediately and are not released until the job finishes. What is the maximum number of jobs that can be in progress at once? What is the maximum and minimum number of tape drives that may be left idle as a result of this policy?

b) Suggest an alternative policy to improve tape drive utilization and at the same time avoid system deadlock. What is the maximum number of jobs that can be in progress at once? What are the bounds on the number of idling tape drives?

6 – Stallings 10.3

Consider the Rate Monotonic scheduling that schedules periodic tasks with priority based on length of the periods. Short period, high priority and vice versa.

$$\frac{C_1}{T_1} + \frac{C_1}{T_1} + ... + \frac{C_n}{T_n} \leq n(2^{1/n} - 1) \tag{1}$$

This problem demonstrates that, although the following equation is a sufficient condition for successful scheduling, it is not a necessary condition. Sometimes successful scheduling is possible even if the equation is not satisfied.

a) Consider a task set with the following independent periodic tasks:

   - Task P_1: $C_1 = 20$: $T_1 = 100$
   - Task P_2: $C_2 = 30$: $T_2 = 145$

   Can these tasks be successfully scheduled using rate monotonic scheduling'?

b) Now add the following task to the set:

   - Task P_3: $C_3 = 65$; $T_3 = 150$

   Is the equation satisfied?

c) Suppose that the first instance of the preceding three tasks arrives at time t = 0. Assume that the first deadline for each task is the following:

$$D_1 = 100; D_2 = 145; D_3 = 150$$

   Using rate monotonic scheduling, will all three deadlines be met? What about deadlines for future repetitions of each task?

7 – Silberschatz 5.6

Consider a variant of the Round Robin (RR) scheduling algorithm in which 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 major advantages and disadvantages of this scheme?

c) How would you modify the basic RR algorithm to achieve the same effect without the duplicate pointers?

8 – Silberschatz 5.10

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

a) First Come First Served

b) Round Robin

c) Multilevel feedback queues