'2025 Final-Term Report

5.3 Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use nonpreemptive scheduling, and base all decisions on the information you have at the time the decision must be made.

Process	Arrival Time	Burst Time
P_1	0.0	8
P_2	0.4	4
P_3	1.0	1

a. What is the average turnaround time for these processes with the FCFS scheduling algorithm?

b. What is the average turnaround time for these processes with the SJF scheduling algorithm?

c. The SJF algorithm is supposed to improve performance, but notice that we chose to run process P_1 at time 0 because we did not know that two shorter processes would arrive soon. Compute what the average turnaround time will be if the CPU is left idle for the first 1 unit and then SJF scheduling is used. Remember that processes P_1 and P_2 are waiting during this idle time, so their waiting time may increase. This algorithm could be known as *future-knowledge scheduling*.

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

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

The processes are assumed to have arrived in the order P1 , P2 , P3 , P4 , P5 , 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 larger priority number implies a higher priority), and RR (quantum = 2).

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 these scheduling algorithms?

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

5.13 Consider the exponential average formula used to predict the length of the next CPU burst. What are the implications of assigning the following values to the parameters used by the

algorithm?

a. $\alpha = 0$ and $\tau_0 = 100$ milliseconds

b. $\alpha = 0.99$ and $\tau_0 = 10$ milliseconds

5.15 The following processes are being scheduled using a preemptive, priority-based, round-robin scheduling algorithm.

Process	Priority	Burst	Arrival
P_1	8	15	0
P_2	3	20	0
P_3	4	20	20
P_4	4	20	25
P_5	5	5	45
P_6	5	15	55

Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. The scheduler will execute the highest priority process. For processes with the same priority, a round-robin scheduler will be used with a time quantum of 10 units. If a process is preempted by a higher-priority process, the preempted process is placed at the end of the queue.

a. Show the scheduling order of the processes using a Gantt chart.
b. What is the turnaround time for each process?
c. What is the waiting time for each process?

6.2 What is the meaning of the term busy waiting? What other kinds of waiting are there in an operating system? Can busy waiting be avoided altogether? Explain your answer.

6.4 Show that, if the wait() and signal() semaphore operations are not executed atomically, then mutual exclusion may be violated.

6.5 Illustrate how a binary semaphore can be used to implement mutual exclusion among n processes.

6.7 The pseudocode of Figure 6.15 illustrates the basic push() and pop() operations of an array-based stack. Assuming that this algorithm could be used in a concurrent environment, answer the following questions:

a. What data have a race condition?
b. How could the race condition be fixed?

```
push(item) {
   acquire();
   if (top < SIZE) {
      stack[top] = item;
      top++;
   }
   else
      ERROR
   release();
}

pop() {
   acquire();
   if (!is_empty()) {
      top--;
      item = stack[top];
      release();
      return item;
   }
   else
      ERROR
   release();
}

is_empty() {
   if (top == 0)
      return true;
   else
      return false;
}
```

Figure 6.15 Array-based stack for Exercise 6.7.

7.3 Describe what changes would be necessary to the producer and consumer processes in Figure 7.1 and Figure 7.2 so that a mutex lock could be used instead of a binary semaphore.

7.4 Describe how deadlock is possible with the dining-philosophers problem.

8.2 Suppose that a system is in an unsafe state. Show that it is possible for the threads to complete their execution without entering a deadlocked state.

8.3 Consider the following snapshot of a system:

Answer the following questions using the banker 's algorithm:

a. What is the content of the matrix Need?
b. Is the system in a safe state?
c. If a request from thread T1 arrives for (0,4,2,0), can the request be granted immediately?

	Allocation	Burst	Arrival
	A B C D	A B C D	A B C D
T_0	0 0 1 2	0 0 1 2	1 5 2 0
T_1	1 0 0 0	1 7 5 0	
T_2	1 3 5 4	2 3 5 6	
T_3	0 6 3 2	0 6 5 2	
T_4	0 0 1 4	0 6 5 6	

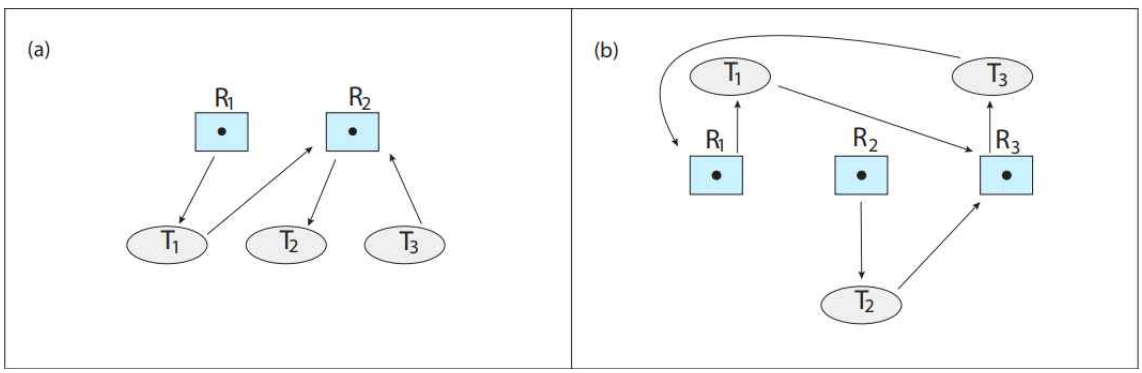
8.9 Consider the following snapshot of a system:

	Allocation	Max
	A B C D	A B C D
T_0	3 0 1 4	5 1 1 7
T_1	2 2 1 0	3 2 1 1
T_2	3 1 2 1	3 3 2 1
T_3	0 5 1 0	4 6 1 2
T_4	4 2 1 2	6 3 2 5

Using the banker's algorithm, determine whether or not each of the following states is unsafe. If the state is safe, illustrate the order in which the threads may complete. Otherwise, illustrate why the state is unsafe.

a. Available = (0, 3, 0, 1)
b. Available = (1, 0, 0, 2)

8.15 Which of the six resource-allocation graphs shown in Figure 8.12 illustrate deadlock? For those situations that are deadlocked, provide the cycle of threads and resources. Where there is not a deadlock situation, illustrate the order in which the threads may complete execution.



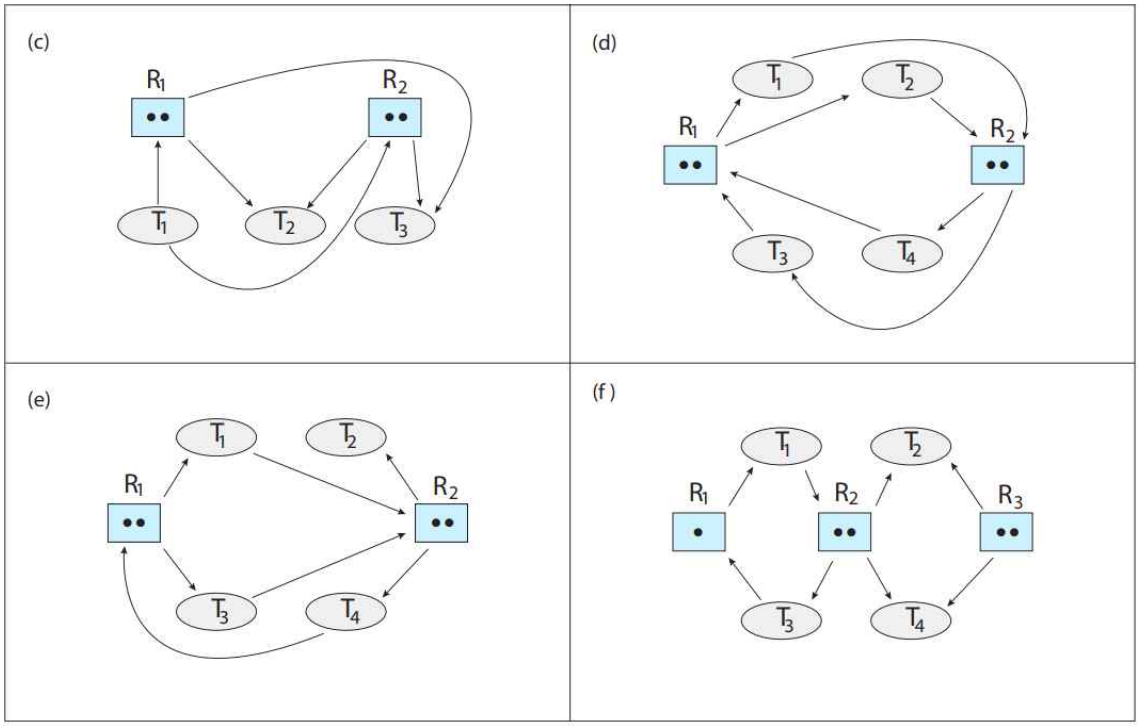


Figure 8.12 Resource-allocation graphs for Exercise 8.15.

8.18 Consider the following snapshot of a system:

	Allocation	Max
	A B C D	A B C D
T_0	1 2 0 2	4 3 1 6
T_1	0 1 1 2	2 4 2 4
T_2	1 2 4 0	3 6 5 1
T_3	1 2 0 1	2 6 2 3
T_4	1 0 0 1	3 1 1 2

Using the banker's algorithm, determine whether or not each of the following states is unsafe. If the state is safe, illustrate the order in which the threads may complete. Otherwise, illustrate why the state is unsafe.

a. Available = (2, 2, 2, 3)
b. Available = (4, 4, 1, 1)
c. Available = (3, 0, 1, 4)
d. Available = (1, 5, 2, 2)

9.4 Consider a logical address space of 64 pages of 1,024 words each, mapped onto a physical memory of 32 frames.

a. How many bits are there in the logical address?
b. How many bits are there in the physical address?

9.7 Assuming a 1-KB page size, what are the page numbers and offsets for the following address references (provided as decimal numbers):

a. 3085
b. 42095
c. 215201
d. 650000
e. 2000001

9.9 Consider a logical address space of 256 pages with a 4-KB page size, mapped onto a physical memory of 64 frames.

a. How many bits are required in the logical address?
b. How many bits are required in the physical address?