Lista 2

Chapter 5

5.2 A CPU-scheduling algorithm determines an order for the execution of its scheduled processes. Given *n* processes to be scheduled on one processor, how many different schedules are possible? Give a formula in terms of *n*.

5.3 Consider a system running ten I/O-bound tasks and one CPU-bound task. Assume that the I/O-bound tasks issue an I/O operation once for every millisecond of CPU computing and that each I/0 operation takes 10 milliseconds to complete. Also assume that the context-switching overhead is 0.1 millisecond and that all processes are long-running tasks. Describe the CPU utilization for a round-robin scheduler when:

a)The time quantum is 1 millisecond

b) The time quantum is 10 milliseconds

5.9 Which of the following scheduling algorithms could result in starvation?

1. First-come, first-served
2. Shortest job first
3. Round robin
4. Priority

5.10 Suppose that a scheduling algorithm (at the level of short-term CPU scheduling) favors those processes that have used the least processor time in the recent past. Why will this algorithm favor I/O-bound programs and yet not permanently starve CPU-bound programs?

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

1. What would be the effect of putting two pointers to the same process in the ready queue?
2. What would be two major advantages and two disadvantages of this scheme?
3. How would you modify the basic RR algorithm to achieve the same effect without the duplicate pointers?

5.13 Consider the following set of processes, with the length of the CPU burst given in milliseconds:
Process Burst Time Priority ----

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 P1, P2, P3, P4, P5, all at time 0.

1. 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).
2. What is the turnaround time of each process for each of the scheduling algorithms in part a?

3. What is the waiting tim of each process for each of these scheduling algorithms?

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

5.14 The traditional UNIX scheduler enforces an inverse relationship between priority numbers and priorities: the higher the the lower the priority. The scheduler recalculates process priorities once per second using the following function:

Priority = (recent CPU usage *I* 2) + base

where base = 60 and *recent* CPU *usage* refers to a value indicating how often a process has used the CPU since priorities were last recalculated.

Assume that recent CPU usage for process P1 is 40, for process P2 is 18, and for process P3 is 10. What will be the new priorities for these three processes when priorities are recalculated? Based on this information, does the traditional UNIX scheduler raise or lower the relative priority of a CPU-bound process?

5.15 Discuss how the following pairs of scheduling criteria conflict in certain settings.

1. CPU utilization and response time
2. Average turnaround time and maximum waiting time
3. I/0 device utilization and CPU utilization

Chapter 6

6.9 Servers can be designed to limit the number of open connections. For example, a server may wish to have only *N* socket connections at any point in time. As soon as *N* connections are made, the server will not accept another incoming connection until an existing connection is released. Explain how semaphores can be used by a server to limit the number of concurrent connections.
Describe a mechanism for enforcing memory protection in order to prevent a program from modifying the memory associated with other programs.

6.10 Why do Solaris, Lil1.ux, and Windows XP use spinlocks as a syn- chronization mechanism only on multiprocessor systems and not on single-processor systems?

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

6.12 Show how to implement the wait() and signal() semaphore operations in multiprocessor environments using the TestAndSet () instruction. The solution should exhibit minimal busy waiting.

6.17 Explain why implementing synchronization primitives by disabling interrupts is not appropriate in a single-processor system if the synchronization primitives are to be used in user-level programs.

Chapter 7

7.3 Consider a system consisting of four resources of the same type that are shared by three processes, each of which needs at most two resources. Show that the system is deadlock free.

7.7 Consider the following resource-allocation policy. Requests for and releases of resources are allowed at any time. If a request for resources cannot be satisfied because the resources are not available, then we check any processes that are blocked waiting for resources. If a blocked process has the desired resources, then these resources are taken away from it and are given to the requesting process. The vector of resources for which the blocked process is waiting is increased to include the resources that were taken away.

For example, consider a system with three resource types and the vector Available initialized to (4,2,2). If process P0 asks for (2,2,1), it gets them. If P1 asks for (1,0,1), it gets them. Then, if P0 asks for (0,0,1), it is blocked (resource not available). If P2 now asks for (2,0,0), it gets the available one (1,0,0) and one that was

allocated to P0 (since P0 is blocked). P0's Allocation vector goes down to (1,2,1), and its Need vector goes up to (1,0,1). a. Can deadlock occur? If you answer "yes," give an example. If you answer "no," specify which necessary condition cannot occur. b. Can indefinite blocking occur? Explain your answer.

7.9 Compare the circular-wait scheme with the deadlock-avoidance schemes (like the banker's algorithm) with respect to the following issues:

a. Runtime overheads

b. System throughput

7.10 Consider the following snapshot of a system:

	Allocation	Max	Available
	A B C D	A B C D	A B C D
P0	0 0 1 2	0 0 1 2	1 5 2 0
P1	1 0 0 0	1 7 5 0	
P2	1 3 5 4	2 3 5 6	
P3	0 6 3 2	0 6 5 2	
P4	0 0 1 4	0 6 5 6	

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 process P1 arrives for (0,4,2,0), can the request be granted immediately?

7.12 Consider a computer system that runs 5,000 jobs per month with no deadlock-prevention or deadlock-avoidance scheme. Deadlocks occur about twice per month, and the operator must terminate and rerun about 10 jobs per deadlock. Each job is worth about $2 (in CPU time), and the jobs terminated tend to be about half-done when they are aborted. A systems programmer has estimated that a deadlock-avoidance algorithm (like the banker's algorithm) could be installed in the system with an increase in the average execution time per job of about 10 percent. Since the machine currently has 30-percent idle time, all 5,000 jobs per month could still be run, although turnaround time would increase by about 20 percent on average. a. What are the arguments for installing the deadlock-avoidance algorithm? b. What are the arguments against installing the deadlock-avoidance algorithm?

7.13 Consider the deadlock situation that could occur in the dining philosophers problem when the philosophers obtain the chopsticks one at a time. Discuss how the four necessary conditions for deadlock indeed hold in this setting. Discuss how deadlocks could be avoided by eliminating any one of the four conditions

7.14 What is the optimistic assumption made in the deadlock-detection algorithm? How can this assumption be violated?

7.16 Is it possible to have a deadlock involving only a single process? Explain your answer.