CS 4/53201, Operating Systems, Spring 2006

Department of Computer Science
Kent State University
Take Home Assignment#2

All problem and page numbers refer to your text book (6th Edition, OS Concepts, Silberschartz).

1. (Problem 6.1) 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***.

2. (Problem 6.3) 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 process are ssumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.
   a. Draw four Grantt 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 part a?
   c. What is the waiting time of each process for each of the scheduling algorithms in part a?
   d. Which of the schedules in part a results in the minimal average waiting time (over all processes)?

3. (Problem 6.4) Suppose that the following processes arrive for execution at the times indicated. Each process will run the listed amount of time. 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
P1	0.0	8
P2	0.4	4
P3	1.0	1

   a. What is the average turnaround time of these processes with the FCFS scheduling algorithm?
   b. What is the average turnaround time of these processes with the SJF scheduling algorithm?
   c. The SJF algorithm is supposed to improve performance, but notice that we chose to run process P1 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 is used. .Remember that processes P1 and P2 are waiting during this idle time, so their waiting time may increase. This algorithm could be known as future-knoeledge scheduling.

4. (Problem 6.7) 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 effort of putting two pointers to the same process in the ready queue?
    b. What would be the 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?

5. (Problem 7.2) Explain why spinlocks are not appropriate for uniprocessor systems yet may be suitable for multiprocessor systems.

6. (Problem 7.4) The first known correct software solution to the critical-section problem for two processs was developed by Dekker. The two processes, P0 and P1, share the following variables:

```
Boolean flag[2];   /* initially false  */
Int turn;
```

    The structure of process Pi (i==0 or 1), with Pj (j == 1 or 0) being the other process, is shown in Figure 7.27.

    Prove that the algorithm satisfies all three requirements for the critical-section problem.

7. Show that message passing and semaphores have equivalent functionality by:
    a. Implementing a message passing using semaphores. Hint: make use of a shared buffer area to hold mailboxes, each one consisting of an array of message slots.
    b. Implementing a semaphore using message passing. Hint: introduce a separate synchronization process.

8. (Problem 8.3) People have said that proper spooling would eliminate deadlocks. Certainly, it eliminates from contention card readers, plotters, printers, and so on. It is even possible to spool tapes (called *staging* them), which would leave the resources of CPU time, memory, and disk space. Is it possible to have a deadlock involving these resources? If it is , how could such a deadlock occur? If it is not, why not? What deadlock scheme would seem best to eliminate these deadlocks (if any are possible), or what condition is violated (if they are not possible)?

9. (Problem 8.6) In a real computer system, neither the resources available nor the demands of processes for resources are consistent over long periods (months). Resources break or are replaced, new processes come and go, new resources are bought and added to the system. If deadlock is controlled by the banker's algorithm, which of the following changes can be made safely (without introducing the possibility of deadlock), and under what circumstances?

a. Increase *Available* (new resources added)
b. Decrease *Available* (resource permanently removed from system)
c. Increase *Max* for one process (the process needs more resources than allowed, it may want more)
d. Decrease *Max* for one process (the process decides it does not need that many resources)
e. Increase the number of processes
f. Decrease the number of processes

10. (Problem 8.13) Consider the following snapshort 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?