Operating Systems, 2nd Term Exam, Chapter 5~8, 6:30pm, Dec. 19, 2003/12/18
(A total of 110 points)

Part I. Write down the full name of the following abbreviated terminology. (5%)

1. FCFS, 2. RR (in scheduling), 3. FIFO, 4.SJF 5.LWP (hint: in Threads)

Part II. True or False. (If your answer is false, please write down your reason.)(10%)

1. ( ) User threads are supported by the kernel and are implemented by a kernel system call interface.
2. ( ) Each thread will have its own copy of code. It will not be shared with other threads.
3. ( ) Although the SJF algorithm is optimal, it cannot be implemented at the level of short-term CPU scheduling.
4. ( ) RR scheduling is specially designed for time-sharing systems.
5. ( ) Suppose that mutex is a semaphore and a process's synchronization code is written as:

   { mutex.P(); critical section; mutex.P();}

   A deadlock will occur.
6. ( ) The monitor construct allows concurrent access to all procedures defined within the monitor.
7. ( ) Only one thread (or process) at a time can be active within the monitor at any time.
8. ( ) When there is a cycle in the resource allocation graph, there must be a deadlock.
9. ( ) Operating system designers often choose to eliminate deadlocks using resource preemption. They preempt some resources from processes and give these resources to other processes until the deadlock cycle is broken.
10. ( ) A P() operation on semaphores is also called a 'signal' operation and a V() is also called a 'wait' operation.

Part III. Choose appropriate answer(s) to fill out the blanks. (Note: Single or multiple choices a re possible.) (10%)

(a) P(S) (b)V(S) (c) while $S \leq 0$; (d) S--; (e) S ++; (f) add this process to list
(g) block (h)test-and-set (i) lock.set(false) (j) lock.set(true)

1. The definitions of P(S) and V(S) are:

   P(S): {___(1)_____}; V(S) {___(2)_____}

Ïhý uú

2. The general strategy for using a binary semaphore to control access to a critical section is:

```
Semaphore S;
.....
_____(3)_____; //enter critical section
critical section();
_____(4)_____;//leave critical section
```

3. Given a Java program segment which shows a thread using test-and-set lock as follows:

```
HardwareData lock = new HardwareData(false);
While (true) {
        While(HardwareSolution.TestAndSet(lock))
            Thread.yield();
        CriticalSection();
        ___(5)_____;
        nonCriticalSection();
}
```

Part IV. Explain the following terminology briefly.(20%)

1. thread pool, 2. preemptive scheduling, 3. CPU utilization
4. response time, 5. starvation, 6. load sharing (in homogeneous multiprocessors)
7. test-and-set instruction, 8.bust waiting 9. safe state (Hint : in 'Deadlock')
10. mutual exclusion

Part V. Answer the following questions briefly.(65%)

1. Why kernel-level threads are better for a multiprocessor environment? (5%)
2. A solution to the problem of starvation of low-priority process is 'aging'. How does 'aging' work? Can you show an example? (5%)
3. Please show an example for the 'race condition'. (5%)
4. What is 'signal-and-wait' and 'signal-and-continue' in a monitor?(5%)
5. Can you describe the bakery algorithm?(5%)
6. Consider the following set of processes, with the length of the CPU-burst time given in milliseconds:

   P1 (burst time 10ms, priority 3); P2(burst time 1ms, priority 1);
   P3 (burst time 2ms, priority 3); P4(burst time 1ms, priority 4)
   P5 (burst time 5ms, priority 2)
   The processes are assumed to have arrived in the order P1,P2,P3,P4,P5 all at time 0.

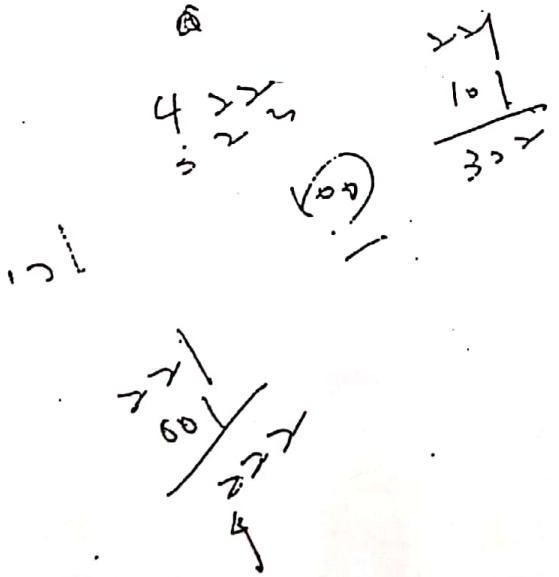
(a) Draw four Gantt charts that illustrate the execution of these processes using SJF, a nonpreemptive priority (a smaller priority number implies a higher priority), and RR (quantum =1) scheduling. (5%)

(b) What is the waiting time of each process for each of the scheduling algorithms in part (a).(5%)

7. What is 'atomic transaction'? Suppose that an on-line transaction is failed. Can you show an example that such a transaction could be recovered by the atomic transaction? (10%)
8. What is 'priority inversion' in a priority-scheduling system?(5%) How to solve this problem? (5%)
9. Consider the following resource-allocation policy. Requests and releases from 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 they have the desired resources, then these resources are taken away from them and are given the requesting process. The vector of resources for which the 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 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. (5%)

(b) Can indefinite blocking occur? Explain your answer. (5%)



Ïhý uú