Name: ______________________________

CS 4/53201

Exam #2

Operating Systems

Friday 13 November 1998

---

1. Define each of the following terms. (5 points each = 25 points)

a. mutual exclusion

Allowing only one thread at a time to perform a particular operation or execute a particular section of code (the critical section)

b. atomic operation

An operation that must be completed indivisibly — without interruption once it starts

c. busy waiting

Waiting in a loop doing nothing until some particular condition is met

d. CPU burst

Time spent by a thread executing on the CPU before it blocks or terminates (note that this is not the same as the thread's total execution time)

e. hold and wait

One of the four necessary and sufficient conditions for deadlock; it refers to a process holding one resource and waiting to acquire another

2. Consider this definition of semaphores, which we will call version 1:

```
wait(s):                                 signal(s):
s = s – 1                                s = s + 1
if (s < 0)                               if (s <= 0)
   block the thread that called wait        wake up a waiting thread
else
   continue into critical section
```

Also consider this definition, which we will call version 2:

```
wait(s):                                 signal(s):
if (s <= 0)                              if (a thread is waiting)
   block the thread that called wait        wake up a waiting thread
s = s – 1                                s = s + 1
continue into critical section
```

How do these two definitions compare? Assuming they're implemented correctly (as atomic operations, etc.), do they both work? If not, which one doesn't work, and why? If they do both work, is there any difference between them? (10 points)

Yes, both work. The main difference is that in version 1 the semaphore can take on negative values, and the absolute value of a negative value tells the number of threads blocked. In version 2, the semaphore can't go below 0, so to know the number of threads blocked the OS would have to look at the length of the queue.

3. **Lock ACQUIRE and RELEASE operations can be implemented fairly directly using semaphore WAIT and SIGNAL operations, respectively. Can condition variable WAIT and SIGNAL operations also be implemented using semaphore WAIT and SIGNAL operations, respectively? Explain your answer. (15 points)**

   No, for two main reasons. First, condition variable operations are designed to work within locks, so a condition variable WAIT operation releases the lock before it sleeps; a semaphore WAIT operation would not do this. Second, condition variables do not have a value, whereas semaphores do, so if a semaphore SIGNAL occurs before a semaphore WAIT, the WAIT does not wait, but if a condition variable SIGNAL occurs before a condition variable WAIT, the WAIT still waits (it always waits!).

4. **For each of the following scheduling algorithms, (i) clearly indicate <u>when</u> the CPU scheduler is invoked, and (ii) briefly explain the criteria used by the CPU scheduler to pick the next process to execute. (5 points each =15 points)**

   a. **FCFS scheduling**

      Scheduler is invoked when the running process blocks or terminates.

      CPU scheduler picks the process at the head of the ready queue.

   b. **Nachos scheduler**

      Scheduler is invoked when the running process blocks (sleeps), terminates, or yields.

      CPU scheduler picks the process at the head of the ready queue.

   c. **SRT scheduling**

      Scheduler is invoked when the running process blocks or terminates, or when a new or blocked process enters the ready queue.

      CPU scheduler picks the process with the shortest remaining CPU time (predicted).

5. **One method for dealing with deadlock is detection and recovery.**

   a. **Explain how deadlock can be detected in a system with <u>single</u> instances of each resource type. (15 points)**

      Sketch of answer: you should explain either a RAG or WFG, how to search it for cycles, and why a cycle indicates deadlock. See Lecture 20 for details.

Name: ______________________________

b. Explain how deadlock can be detected in a system with multiple instances of each resource type. (15 points)

Sketch of answer: you should explain about keeping track of available resources, and current allocation and request for each process, and marking processes that can be completed. See Lecture 20 for details.

Or: you could explain about knots, and why a knot indicates deadlock. However, we didn't examine any algorithms in class for searching for knots, so this is a slightly less-satisfactory answer.