# Fall 2014
# CSE 421/521 – Operating Systems
# Homework Assignment #2

***** SOLUTION KEY *****

**Problem 1:**

Consider a preemptive priority scheduling algorithm based on dynamically changing priorities. Larger priority numbers imply higher priority. When a process is waiting for the CPU (in the ready queue, but not running), its priority changes at a rate $\alpha$; when it is running, its priority changes at a rate $\beta$. All processes are given a priority of 0 when they enter the ready queue. The parameters $\alpha$ and $\beta$ can be set to give many different scheduling algorithms.

a. What is the algorithm that results from $\beta > \alpha > 0$?

b. What is the algorithm that results from $\alpha < \beta < 0$?

**Answer:**

a. FCFS

b. LIFO

**Problem 2:**

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/O 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. What is the CPU utilization for a round-robin scheduler when:

a. The time quantum is 1 millisecond

b. The time quantum is 10 milliseconds

**Answer:**

- The time quantum is 1 millisecond: Irrespective of which process is scheduled, the scheduler incurs a 0.1 millisecond context-switching cost for every context-switch. This results in a CPU utilization of $1/1.1 * 100 = 91\%$.
- The time quantum is 10 milliseconds: The I/O-bound tasks incur a context switch after using up only 1 millisecond of the time quantum. The time required to cycle through all the processes is therefore $10*1.1 + 10.1$ (as each I/O-bound task executes for 1 millisecond and then incur the context switch task, whereas the CPU-bound task executes for 10 milliseconds before incurring a context switch). The CPU utilization is therefore $20/21.1 * 100 = 94\%$.

**Problem 3:**

Which of the following scheduling algorithms could result in starvation? Explain the reasoning.

a. First-come, first-served
b. Shortest job first
c. Round robin
d. Priority

**Problem 4:**

Write a bounded-buffer monitor in which the buffers (portions) are embedded within the monitor itself.

**Answer:**

```
monitor bounded_buffer {
    int items[MAX_ITEMS];
    int numItems = 0;
    condition full, empty;

    void produce(int v) {
        while (numItems == MAX_ITEMS)
            full.wait();
        items[numItems++] = v;
        empty.signal();
    }

    int consume() {
        int retVal;
        while (numItems == 0)
```

```
            empty.wait();
retVal = items[--numItems];
        full.signal();
        return retVal;
    }
}
```

**Problem 5:**

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.

**Answer:** Deadlock is possible because the four necessary conditions hold in the following manner: 1) mutual exclusion is required for chopsticks, 2) the philosophers tend to hold onto the chopstick in hand while they wait for the other chopstick, 3) there is no preemption of chopsticks in the sense that a chopstick allocated to a philosopher cannot be forcibly taken away, and 4) there is a possibility of circular wait. Deadlocks could be avoided by overcoming the conditions in the following manner: 1) allow simultaneous sharing of chopsticks, 2) have the philosophers relinquish the first chopstick if they are unable to obtain the other chopstick, 3) allow for chopsticks to be forcibly taken away if a philosopher has had a chopstick for a long period of time, and 4) enforce a numbering of the chopsticks and always obtain the lower numbered chopstick before obtaining the higher numbered one.

**Problem 6:**

Consider the dining-philosophers problem where the chopsticks are placed at the center of the table and any two of them could be used by a philosopher. Assume that requests for chopsticks are made one at a time. Describe a simple rule for determining whether a particular

**Answer:** The following rule prevents deadlock: when a philosopher makes a request for the first chopstick, do not satisfy the request only if there is no other philosopher with two chopsticks and if there is only one chopstick remaining.

**Problem 7:**

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

**Answer:** A wait operation atomically decrements the value associated with a semaphore. If two wait operations are executed on a semaphore when its value is 1, if the two operations are not performed atomically, then it is possible that both operations might proceed to decrement the semaphore value thereby violating mutual exclusion.

**Problem 8:**

Write a monitor that implements an *alarm clock* that enables a calling program to delay itself for a specified number of time units (*ticks*). You may assume the existence of a real hardware clock that invokes a procedure *tick* in your monitor at regular intervals.

**Answer:**

```
monitor alarm {
    condition c;

    void delay(int ticks) {
       int begin_time = read_clock();
       while (read_clock() < begin_time + ticks)
       c.wait();
    }

    void tick() {
       c.broadcast();
    }
}
```