

Homework 1 – System Programming

CS241, Spring 2007

Deadline: Friday 03/02/2007, 4pm, 3120 SC

Submission Steps:

- (1) YOUR NETID MUST BE AT THE TOP OF EVERY PAGE;**
- (2) PRINT THE SOLUTIONS – SUBMISSION IN PAPER FORMAT;**
- (3) BRING THE PRINTED HW1 SOLUTION TO ANDA OHLSSON'S OFFICE, 3120 SIEBEL CENTER BY 4PM**

Problem 1 (10 Points)

Consider two processes running on two processors sharing a variable (critical section) in shared memory. Specify the two flag and turn solution (pseudo-code) for synchronizing two processes that satisfies mutual exclusion, progress and bounded wait synchronization conditions. Furthermore, explain (a) one assumption for the software to run correctly, and (b) how each variable contributes to the mutual exclusion, progress and/or bounded waiting requirements.

Problem 2: (20 Points)

- (2 Points) In a multithreaded process, explain why is there normally one stack per thread instead of one stack per process?
- (6 Points) What is a race condition? Explain and give an example of two processes (order of instructions) where a race condition occurs.
- (8 Points) The simplest state diagram for process execution has three states. What are the three states? Explain the meaning of each state and meaning of the allowed transitions. In the corresponding queuing diagram explain one difference and one similarity between the ready queue and the I/O queue.
- (4 Points) Explain why user-mode threads in a single process can use a user-mode semaphore but cannot use a kernel semaphore to perform synchronization.

Problem 3 (30 Points):

Consider the following functions that are shared by multiple processes:

```
static int count = 0;
int increment(void)
{
    count++;
    if (count > 5)
        { printf("counter %d reached value > 5", count);
          return 0; }
    return 1;
}

int decrement(void)
{
    while (count >5)
        { printf("counter %d is > 5:", count);
          count --;
        }
    if (count ==0) return 0;
    else return 1;
}
```

Use any synchronization primitives (semaphores, mutex) to make the two functions atomic.

Problem 4 (10 Points):

- (6 Points) Give a sketch of how an operating system that can disable interrupts could use this to implement semaphores. Describe at which points in the implementation of the two semaphore operations (wait & post) would interrupts be enabled and disabled?
- (4 Points) Does the busy waiting synchronization solution using 'turn' variable (see strict alternation solution) satisfy mutual exclusion and bounded wait when the two processes are running on a shared-memory multiprocessor? Why or why not?

Problem 5 (30 Points)

Consider the workload in Table 1:

Process	Burst Time	Priority	Arrival Time
P1	50ms	4	0 ms
P2	20ms	1	20 ms
P3	100ms	3	40 ms
P4	40ms	2	60ms

- (18 Points) Provide schedule using preemptive Shortest Job First, non-preemptive Priority (a smaller priority number implies higher priority) and Round Robin with quantum 30 ms. Use time scale diagram as shown below for the FCFS example to show the schedule for each requested scheduling policy.

Example for FCFS:

P1	P1	P1	P1	P1	P2	P2	P3	P3	P3	P3	
0	10	20	30	40	50	60	70	80	90	100	110
P3	P3	P3	P3	P3	P3	P4	P4	P4	P4		
120	130	140	140	150	160	170	180	190	200	210	220

- (12 Points) What is the average waiting time of the above scheduling policies?