CSCE 351 Operating System Kernels

Fall 2001
Steve Goddard

Homework 1, September 25

Due: 12:30pm October 4

---

1) (10 points) Consider the following program:

```
P1: {                              P2:{
  shared int x;                      shared int x;
  x = 10;                            x = 10;
  while (1)  {                       while ( 1 ) {
     x = x - 1;                         x = x - 1;
     x = x + 1;                         x = x + 1;
     if (x != 10)                       if (x!=10)
       printf(''x is %d'',x);             printf(''x is %d'',x);
     }                                  }
  }                                  }
}                                  }
```

Note that the scheduler in a uni-processor system would implement pseudo-parallel execution of these two concurrent processes by interleaving their instructions, without restriction on the order of the interleaving.

   a) Show how (i.e., trace the sequence of inter-leavings of statements) such that the statement ''x is 10" is printed.

   b) Show how the statement ''x is 8" is printed. You should remember that the increment/decrements at the source language level are not done atomically, i.e., the assembly language code:

```
LD     R0,X     /* load R0 from  memory location x */
INCR   R0       /* increment R0 */
STO    R0,X     /* store the incremented value back in X */
```

   implements the single C increment instruction (x=x+1).

2) (5 points) In view of the fact that existence of a super-user (as the user with administration privileges is often called in UNIX) can lead to all kinds of security problems, why does such a concept exist?

   What are the pros and cons of having a super-user account?

3) (5 points) In a system with threads, is there one stack per thread or one stack per process? Explain.

4) (10 points) Why is the process table needed in a timesharing system?

Is it also needed in personal computer systems in which only one process exists at a time (where that process takes over the entire machine until it is finished)?

Explain why or why not.

5) (10 points) Suppose that we have a message-passing system using mailboxes. When sending to a full mailbox or trying to receive from an empty one, a process does not block. Instead, it gets an error code back. The process responds to the error code by just trying again, over and over, until it succeeds. Does this scheme lead to race conditions? Explain.

6) (10 points) For each of the following instructions, explain why it should or should not be allowed only in Kernel mode.

   A) Disable all interrupts.

   B) Read the time-of-day clock.

   C) Set the time-of-day clock.

   D) Change the memory map.

7) (10 points) Suppose that you were to designing an advanced computer architecture that did process switching in hardware, instead of having interrupts. What information would the CPU need?

   Describe how the hardware process switching might work.

8) (5 points) The CDC 6600 computers could handle up to 10 I/O processes simultaneously using an interesting form of round-robin scheduling called *processor sharing*. A process switch occurred after each instruction, so instruction 1 came from process 1, instruction 2 came from process 2, etc. The process witching was done by special hardware, and the overhead was **zero**. If a process needed *T* sec to complete in the absence of competition, how much time would it need if processor sharing was used with *n* processes?

9) (10 points) During execution, MINIX maintains a variable proc_ptr that points to the process table entry for the current process. Why?

   Describe another way of achieving the same goal without using a special variable such as proc_ptr.

10) (10 points) The MINIX procedure mini_rec contains a loop. Explain what it is for.

    Is the loop bounded?

    Are interrupts disabled during the loop execution?

    Is this a problem?

11) (15 points) Explain how a context switch in MINIX differs from the (high level) way it was presented in class. Be as precise as you can in your description.