Name:

Operating Systems
Fall 2011
Test 1
September 29, 2011

Closed books, notes, cell phones, PDAs, iPods, laptops, etc. No headphones, please. No calculator needed.

You have 75 minutes to solve 7 problems. You get 10 points for writing your name on the top of this page. As with any exam, you should read through the questions first and start with those that you are most comfortable with. If you believe that you cannot answer a question without making some assumptions, state those assumptions in your answer.

Partial credit will be offered only for meaningful progress towards solving the problems.

Please read and sign below if you agree with the following statement:

In recognition of and in the spirit of the academic code of honor, I certify that I will neither give nor receive unpermitted aid on this exam.

Signature:________________________________

0	/10
1	/15
2	/15
3	/10
4	/10
5	/15
6	/10
7	/15
Total	/100

1. (15 points) **Short Answer Questions**:
   a. The __________ is a layer of software between the applications and the computer hardware that supports applications and utilities.

   b. _________ refers to the ability of an OS to support multiple, concurrent paths of execution within a single process.

   c. __________ was designed to keep the processor and I/O devices, including storage devices, simultaneously busy to achieve maximum efficiency.

   d. ___________ is a facility that allows programs to address memory from a logical point of view, without regard to the amount of main memory physically available.

   e. A situation in which two or more processes are unable to proceed because each is waiting for one of the others to do something is a _________.

   f. In the case of competing processes three control problems must be faced: mutual exclusion, deadlock, and _________.

   g. A situation in which multiple threads or processes read and write a shared data item and the final result depends on the relative timing of their execution is a _________.

   h. __________ is a function or action implemented as a sequence of one or more instructions that appears to be indivisible; no other process can see an intermediate state or interrupt the operations.

   i. A significant point about the ___________ is that it contains sufficient information so that it is possible to interrupt a running process and later resume execution as if the interruption had not occurred.

   j. _________ is a section of code within a process that requires access to shared resources and that must not be executed while another process is in a corresponding section of code.

2. (15 points) **Short attention span:** Answer the following questions (3 points each) on general concepts in operating systems.

   a. Compare the overhead of context switching between two threads in different processes and two processes. Explain.

   b. Can multiprogramming be useful in a single-user system? Explain your answer.

   c. Which of the following instructions should be privileged?
      1. Set value of timer
      2. Read the clock
      3. Clear memory
      4. Turn off interrupts
      5. Switch from user to kernel mode
      6. Access I/O device

   d. What is the purpose of system calls?

   e. In a system with threads, is there normally one stack per thread or one stack per process? Explain.

3. (10 points) **Process states:** Below is the process state transition diagram in Unix. How do the states in this diagram (and the transition between states) map onto the simplified, general 5-state model we discussed?

   [To write your answer, list the 5 states in the simple model. For each of them, list the corresponding states from the Unix model. Explain how they are related and the transition between states. You have more room on the next page.]

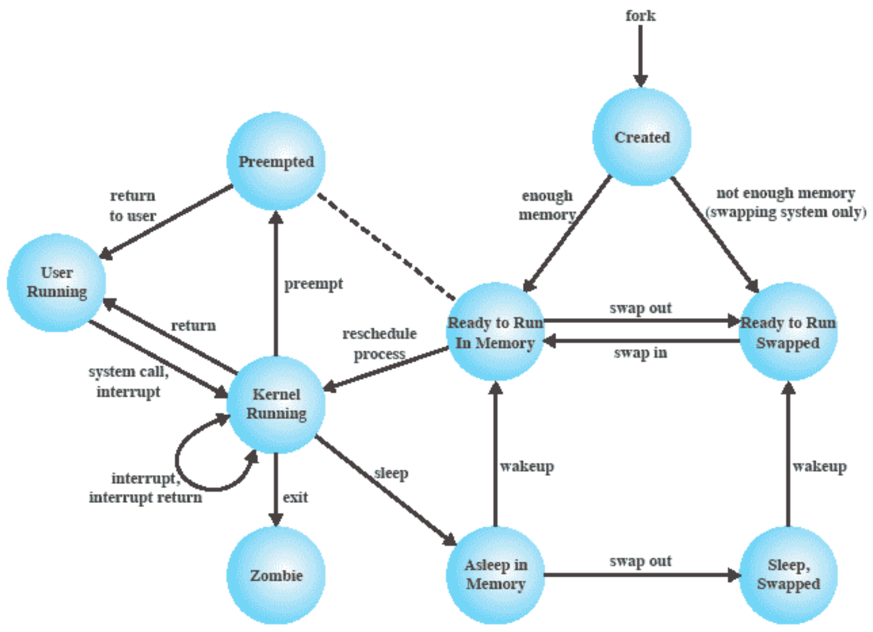

Figure 3.17 UNIX Process State Transition Diagram

4. (10 points) During a process switch, the operating system executes instructions that choose the next process to execute. These instructions are typically at a fixed location in memory. Why?

5. (15 points) The execution of the following program will create one or more processes. As a result, one or more statements will be printed to the screen.
   a. Show the process hierarchy resulted from the execution of this program.
   b. What will this program print on screen when it executes?

```
#include <stdio.h>
main()
{
   int x, y, count;
   x = 20; y = 30;
   count = 1;
   while (count < 3){
         if (x != 0){
               x = fork();
               y = y + 20;
         }
         else {
               x = fork();
               y = y + 35;
         }
   printf("y = %d\n", y);
   count = count + 1;
   }
}
```

6. (10 points) The following solution for the Barbershop problem was presented in class. Your classmate suggests that it would be simple and more elegant to implement the functionality in the code marked below with operations on a counting semaphore initialized to the number of chairs in the barbershop. Is this a good suggestion? If yes, write your suggested code to the right of the provided solution. If no, support your answer.

   Barbershop problem statement: There is one barber and n chairs for waiting customers. If there are no customers, then the barber sits in his chair and sleeps. When a new customer arrives and the barber is sleeping, the customer will wake up the barber. When a new customer arrives, and the barber is busy, he will take a sit on a chair if there is any available, otherwise (when all the chairs are full) he will leave.

```
int customers = 0;
mutex = Semaphore(1);
customer = Semaphore(0);
barber = Semaphore(0);

void customer (void){
     semWait(mutex);
     if (customers==n+1) {
          semSignal(mutex);
          balk();
     }
     customers +=1;
     semSignal(mutex);
     semSignal(customer);
     semWait(barber);
     getHairCut();
     semWait(mutex);
     customers -=1;
     semSignal(mutex);
}

void barber (void){
     semWait(customer);
     semSignal(barber);
     cutHair();
}
```

7. (15 points) Consider the following program:

```
void undecided()
{
      x = 10;
      while(1){
            x = x + 1;
            x = x - 1;
            if (x != 10)
                  printf("x is %d", x);
      }
}

void main(){
      int x; /* shared variable */
      parbegin{undecided(), undecided()};

}
```

The scheduler in a uniprocessor 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 a sequence (i.e., trace the sequence of interleaving of statements) such that the statement "x is 10" is printed. (5 points)
b. Show a sequence such that the statement "x is 8" is printed. (10 points)

Hint: 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).