Name: ______________________________

CS 4/53201

Exam #1

Operating Systems

Monday 9 October 2000

1. **(10 points) Explain how each of the following hardware features is used to protect the system from malicious or erroneous processes.**

   a. **Privileged instructions**

      Potentially harmful or damaging assembly language instructions, such as those that perform I/O, halt the machine, etc., can be designated as privileged instructions. Since privileged instructions can only be executed when the CPU is in kernel mode, and the only way for the CPU to get into kernel mode is through a trap instruction, their use can be restricted the OS.

   b. **Timer**

      A hardware timer can be used to interrupt and run the CPU scheduler after a given process has been running on the CPU for a specified amount of time. This "timeout" mechanism will prevent a process from dominating the CPU for a prolonged (possibly indefinite) period.

2. **(20 points) Suppose that two user-level processes are running on a machine. The wordprocessor (WP) process is currently waiting for the user's next keystroke. The spreadsheet (SS) process is currently recalculating (i.e. using the CPU).**

   a. **Using the 5-state process model, which state is each of the two processes in?**

      WP is in state: blocked **SS is in state:** running

   b. **Now suppose that the user presses a key (for WP), while SS is still calculating. Describe briefly what happens from the time that the key is pressed until the time that the WP process changes state. Hint: hardware is involved.**

      The keypress causes an interrupt to occur, which runs the appropriate interrupt handler (part of the OS, not part of WP) to process the input data. The WP process is then put on the end of the ready list, and execution of the SS process resumes. The WP process does <u>not</u> immediately start running or doing anything with the input data!

   c. **Immediately after the state change at the end of the events described in part b) above, which state is each of the two processes in?**

      WP is in state: ready **SS is in state:** running

3. **(10 points) Describe how the producers / consumers problem with 3 producers and 2 consumers can be solved using message passing and indirect communication. Assume that any consumer can consume an item created by any producer.**

Name: ______________________________

Indirect communication refers to message passing using mailboxes owned by the receiver. In this case, each consumer can set up a mailbox, and any one of the three producers can send to that mailbox. Each producer could alternately send an item to each of the three consumers in turn, or perhaps could only send to a particular consumer in response to a request.

4. (10 points) Suppose an operating system implement its window manager using threads, so that a separate thread is responsible for each window on the screen. Why is it more appropriate to use threads for this purpose than processes?

The various threads can be part of a single window manager process, which will allow them to share data (such as the screen buffer), yet will allow one to block without affecting the others, etc. Furthermore, by being threads instead of processes, switching between windows will be faster than if each window were implemented using a separate process. (Note that just describing why threads are "better" than processes is not sufficient; you have to justify the use of threads in a window manager as well.)

5. (10 points) One way that semaphores can be implemented is by enabling and disabling interrupts (as is done in Nachos).

a. This technique can affect the system's I/O in an adverse way. Explain.

An interrupt system may attempt to "queue" up interrupts during the time that interrupts are disabled, but if too many interrupts are received too quickly, the queue may be overrun and some I/O data may be lost.

b. This technique does not work on multiprocessor systems. Explain.

If one processor disables interrupts, that does not disable interrupts on the other processors, so although other processes / threads on the first processor will be kept out of the critical section, processes / threads on the other processors will not.

6. (15 points) For each of the following code segments, where the variables are initialized as shown, and threads A and B can run concurrently, what are the possible ending values of x?

a.

```
int x = 10;
int y = 35;
semaphore m = 1;
semaphore s = 0;

thread A {
    x = x + 1;
}

thread B {
    x = y + 1;
```

b.

```
(same
 variables)

thread A {
    P(m);
    x = x + 1;
    V(m);
}

thread B {
    P(m);
```

c.

```
(same
variables)

thread A {
    P(s);
    x = x + 1;
    V(m);
}

thread B {
    P(m);
}                                x = y + 1;             x = y + 1;
                                 V(m);                  V(s);
                             }                      }
```

Ending values are:	**Ending values are:**	**Ending values are:**
36 if A runs, then B 37 if B runs, then A 11 if both read old value of X but A writes last	36 if A runs, then B 37 if B runs, then A	37 (A waits until B does a V(s))

7. **(25 points) Consider the code below (taken directly from Lecture 15) for solving the Readers / Writers problem with writers priority:**

```
Reader:                          Writer:
acquire(mutex);                  acquire(mutex);
while (AW+WW > 0) {              while (AW+AR > 0) {
  WR++;                            WW++;
  wait(OKToRead);                  wait(OKToWrite);
  WR– –;                           WW– –;
}                                }
AR++;                            AW++;
release(mutex);                  release(mutex);

read database                    write database

acquire(mutex);                  acquire(mutex);
AR– –;                           AW– –;
if (AR == 0 &&                     if (WW > 0)
  WW > 0)                            signal(OKToWrite);
  signal(OKToWrite);               else
release(mutex);                      br'cast(OKToRead);
                                 release(mutex);
```

a. **Since writers have priority, if there are active or waiting writers, a reader will have to wait. Where in the code does it wait?**

   The first reader will wait at the wait(OKToRead) at the top of the reader's code.

b. **When a reader is waiting to read as in part a) above, what happens if another reader starts to execute?**

   If this were semaphores, other readers would have to wait at the acquire(mutex) at the top of the reader's code. However, a condition variable wait releases the lock, so the other readers will wait at the same wait(OKToRead) as in part (a).

c. **What is the purpose of the code after "read database" in the reader?**

   When the last reader finishes reading the database, if there is one or more waiting writers, it must signal a waiting writer to tell it that it's OK to write to the database.

d. **How are multiple writers prevented from writing the database at the same time?**

If one writer is writing, other writers will have to wait at the wait(OKToWrite) at the top of the writer's code.

e. (Extra Credit) Why is there a broadcast at the end of the writers code instead of a simple single?

Because it is OK for multiple readers to read the database at the same time, all readers are awakened instead of only waking one.