Exam 1

CS 5460/6460, Fall 2010

10/26/2010

Name: ____

- You have until the end of class to complete this open-book, open-note exam.
- No computers, calculators, phones, or other electronic devices may be used.
- Please write all answers in the provided space; use the back of the exam if necessary.
- Make sure you have all 4 pages.
- Don't forget to write your name on this exam.
- It is intended that you can answer the questions on this exam without making additional assumptions. However, if you must make additional assumptions in order to provide a reasonable answer, go ahead and do so, *but write down the assumptions as part of your answer.*

1) Explain what is gained by mapping the kernel's memory into each process's address space. Also explain why doing so does not constitute a security hazard.

2) To the right of each UNIX program below, list all of its possible outputs (some programs may have only one possible output). You may assume that no system call or library call fails, that printf is atomic, and that the proper header files are included by each program.

```
int main (void) {
  int x = 0;
  x++;
  if (fork()) {
  }
  printf ("%d\n", x);
  return 0;
}
```

```c
int main (void) {
  int x = 0;
  if (fork()) {
    x++;
  }
  printf ("%d\n", x);
  return 0;
}
```

```c
int main (void) {
  int x = 0;
  if (fork()) {
  }
  x++;
  printf ("%d\n", x);
  return 0;
}
```

```c
int x=0;
pthread_mutex_t l;

void *thread_func (void *p) {
  pthread_mutex_lock (&l);
  x++;
  pthread_mutex_unlock (&l);
  return 0;
}

void printx (void) {
  pthread_mutex_lock (&l);
  printf ("%d\n", x);
  pthread_mutex_unlock (&l);
}

int main (void) {
  pthread_t t1, t2;
  pthread_mutex_init (&l, NULL);
  printx();
  pthread_create (&t1, NULL, thread_func, NULL);
  pthread_create (&t2, NULL, thread_func, NULL);
  printx();
  pthread_join (t1, NULL);
  pthread_join (t2, NULL);
  printx();
  return 0;
}
```

3) Recall from lecture that the Bakery synchronization algorithm for N processes uses the following pseudo-code to acquire the lock, where i is the id of the running process:

```
1  choosing[i] = 1;
2  number[i] = max (number[0], ..., number[N-1]) + 1;
3  choosing[i] = 0;
4  for (j = 0; j < N-1; j++) {
5    while (choosing[j]) { }
6    while ((number[j] != 0) && ((number[j],j) < (number[i],i))) { }
7  }
```

Remember that the less-than operator in line 6 gives a total ordering by comparing the contents of the number array, using process id to break ties. This is the unlock code:

```
1  number[i] = 0;
```

Assuming an in-order memory model, Bakery is a correct mutual exclusion solution. Explain which requirement for a correct mutual exclusion algorithm (mutual exclusion, freedom from deadlock, freedom from livelock) is violated if we remove lines 1, 3, and 5 from the code to acquire the lock (in other words, if we remove the code relating to the "choosing" variable). Give a counterexample (an execution trace) showing how this property is violated.

4) Assume you are designing a virtual memory system for a processor with 30-bit virtual addresses. You want the page size to be 8 KB, and you will use a single-level page table.

- How large is the virtual address space (assuming byte addressing)?
- How many bits of a virtual address are used to represent the page offset?
- How many bits of a virtual address are used to represent the page number?
- How big is the page table for a process, assuming that a page table entry is 8 bytes?

5) Consider a virtual memory system with 3 physical pages. A program whose virtual address space contains 6 pages called A–F accesses those pages in the following order:

C B F E F D E A F B C D E B C

- How many page faults are incurred if page replacement is optimal (the page used farthest in the future is replaced)? _____
- How many page faults are incurred if page replacement is LRU (least recently used)? _____