University of Rochester **Computer Systems (CSC2/456)**

Final Exam

Amy Murphy

6 May 2003

Read before beginning: Please write clearly. Illegible answers cannot be graded. Be sure to identify all of your answers in your blue book (e.g., 1a, 2b, etc). All questions are worth 10 points, numbers in parenthesis indicate relative point values.

While I have tried to make the questions as unambiguous as possible, if you need to make any assumptions in order to continue, state these as clearly as possible as part of your answer. In the name of fairness, I would like to avoid answering questions during the exam.

CSC256: Students enrolled in 256 must choose **6 of the 8** questions to answer. For up to 10 extra credit points, you may answer **one** of the other 2 questions, indicating clearly on the front of the bluebook which question is to be counted as extra credit. If you do not specify, I will assume the first 6 in your bluebook are for *normal* credit and the next one (if any) is *extra.* If you answer all 8 questions, only 7 of them will be counted toward your grade. Grades will be assigned 0-60 (with 10 possible extra credit).

CSC456: Students enrolled in 456 must choose **7 of the 8** questions to answer. For extra credit, you may answer the 8th question, but you must clearly identify which question is to be counted as *extra.* If you do not indicate any, I will assume the 8th answer (if any) is *extra.* Grades will be assigned 0-70 (with 10 possible extra credit)

1. **Short answer.** Your responses for each part should be at most 3-4 lines of text:
   (a) (3) What is the difference between a user-level instruction and a privileged instruction?
   (b) (3) What is the difference between a network operating system and a distributed operating system?
   (c) (4) Name two positive motivations for building distributed systems and two problems they introduce.

2. **Scheduling.** I have just invented a new scheduling algorithm that I claim gives the highest priority to processes that have just entered the system, but is fair to all processes. The algorithm works like this: There are two queues, one for *new* processes and one for *old* processes. When a process enters the system, it is put at the end of the *new* queue. After 2 milliseconds on the *new* queue, whether a process has been scheduled or not, it is moved to the end of the *old* queue. When it is time to schedule a process, the system schedules the process at the head of one of the queues, alternating between the two queues. Each process runs to completion before the next process is scheduled. Assume that processes enter the system at random times and that most processes take much longer than 2 milliseconds to execute.
   (a) (4) Does this algorithm give the highest priority to new processes? Explain your answer.
   (b) (3) Is this algorithm starvation free? Explain your answer.
   (c) (3) Discuss whether this algorithm is fair to all processes. By "fair" we mean every process has a wait time approximately equal to the average wait time, assuming all processes have close to the same execution time.

3. **Synchronization.** Suppose a program has three threads Thread$_1$, Thread$_2$, and Thread$_3$, and a shared counter, `count`, as shown below:

```
int count = 10;
Semaphore Lock = 1; // initial value is 1
```

```
Thread1(...)
{
// do something
Lock.Wait();
count++;
Lock.Signal();
}
```

```
Thread2(...)
{
// do something
Lock.Wait();
count--;
Lock.Signal();
}
```

```
Thread3(...)
{
// do something
Lock.Wait();
printf(‘‘$d’’, count);
Lock.Signal();
}
```

   (a) (4) What are the possible outputs of this program? Hint: there is more than one answer, and you must provide them all.

   (b) (3) Define a race condition.

   (c) (3) Does this process suffer from a race condition? Justify your answer.

4. **Caching.**

   (a) (3) Define spatial locality and temporal locality.

   (b) (3) Disk manufacturers added a full track buffer to hard drives to improve program performance. Which of spatial or temporal locality most motivated this? Explain your answer.

   (c) (4) Hardware caches in the memory hierarchy are other performance enhancing techniques. Explain how/why spatial locality and temporal locality contribute to this performance enhancement.

5. **Paging.** Consider a reference string 1,2,3,4,2,5,6,2,3,2,1,6,7; and a system with only 4 frames, pure demand paging, and all frames initially empty.

   (a) (2) How many page faults would occur with a FIFO replacement scheme? What are the identities of pages in the frames when the reference string has completed?

   (b) (2) How many page faults would occur with a perfect LRU replacement scheme? What are the identities of pages in the frames when the reference string has completed?

   (c) (3) Describe one possible implementation scheme for LRU.

   (d) (3) Would increasing the number of frames always decrease the number of page faults for a particular reference string for FIFO? for LRU? Why or why not? You need not provide "proofs", just an explanation.

6. **Memory Management.** LRU can be thought of as an attempt to predict future memory access patterns based on previous access patterns (i.e. if a page has not been accessed for a while, it is not likely to be be referenced again soon). Another idea that some researchers have explored is to record the memory reference pattern from the last time the program was run and use it to predict what it will access next time.

   (a) (3) For what kinds of programs will this be most effective?

   (b) (3) In what kinds of programs might this lead to worse performance?

   (c) (4) Why would this be difficult to implement in an operating system? Give at least two reasons.

7. **File systems.**

   (a) (4) A file system must keep track of free blocks on disk. Name two schemes for doing this, and one advantage of each.

   (b) (3) Consider a very large file: the maximum size file supported by a particular system. Describe the process to read the last byte of that file if it is stored in a FAT file system. Assume you already have the directory entry cached.

   (c) (3) Describe the process to read the last byte of that file if it is stored in a UNIX inode-like system where the inode contains 10 direct block pointers, one single indirect pointer, one double indirect pointer, and one triple indirect pointer. Assume you already have the inode cached.

8. **Unix File Systems.**

   (a) (3) One of the changes FFS (Fast File System) made over the "classic" Unix file system was to place the inodes at multiple locations on the disk, as opposed to placing all inodes at the front. Why did this (generally) increase file system performance?

   (b) (3) FFS tries to achieve the advantages of both of large and small block sizes. Briefly describe the technique by which this is achieved, and state one of its disadvantages.

   (c) (4) What was the motivation for LFS (Log-based File System), and why does it achieve better write performance than FFS when there are many random writes of a large file?