CSE 120 Principles of Computer Operating Systems
Fall Quarter, 2000
Final Exam

Instructor: Geoffrey M. Voelker

Name _____

Student ID _____

Attention: This exam has nine questions worth a total of 110 points, and the last one is a freebie. You have three hours to complete the questions. 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.

Best of luck, and have a great holiday!

1	/12
2	/14
3	/12
4	/12
5	/8
6	/14
7	/10
8	/18
9	10/10
Total	/110

1. (12 pts) Potpourri: Answer yes or no, or with a single term or phrase, as appropriate. You should go through these quickly, answering with the first answer that comes to mind — it probably is the correct one. Only dwell on ones you are unsure of after finishing the other questions.

   (a) Which bit in the PTE does the operating system use for approximating LRU replacement?

   (b) The layout of disk blocks for a file using Unix inodes is always/sometimes/never contiguous on disk?

   (c) Does a TLB miss always/sometimes/never result in a pagein from disk?

   (d) In RPC, what does the server-stub call?

   (e) What are the two components of a virtual address used in segmented virtual memory?

   (f) In general, are there more/same/fewer inodes than directories in a file system such as Unix?

   (g) In what Nachos file was StartProcess implemented?

   (h) What kind of pages in a process' virtual address space are usually protected as "Read Only"?

   (i) The layout of virtual pages for a virtual address space is always/sometimes/never contiguous in physical memory?

   (j) In what metric do RAID arrays improve performance?

   (k) Is it possible to implement a file system inside of a file stored in another file system?

   (l) Was the exam was too easy/just right/too hard (this is a free point)?

2. (14 pts) Identify the following in one phrase or sentence (do more than just expand the acronym, though). Then state whether each item is related to an operating system *mechanism* or *policy*.

    (a) PTE

    (b) SCAN

    (c) Working set

    (d) LRU

    (e) Thrashing

    (f) Inode

    (g) Access control list

3. (12 pts) Operating systems frequently exploit locality to improve performance. *Briefly* describe two examples where operating systems do so, and state how locality is exploited.

4. (12 pts) One day famed student Joe Surfer had an inspiration while hanging ten at Mission Beach. He observes that most programs have most of their data at the beginning of the address space. For his homegrown SaltWater OS, he decides that he is going to implement his page tables similar to the way Unix implements inodes. He calls this page table design *Inode Page Tables*. Inode Page Tables are essentially two-level page tables with the following twist: The first half of the page table entries in the master page table directly map physical pages, and the second half of the entries map to secondary page tables as normal. Call the first half the entries *fast*, and the second half *normal*.

For the following questions, assume that addresses are 32 bits, the page size is 4 KB, and that the master and secondary page tables fit into a single page.

   (a) How many virtual pages are *fast* pages?

   (b) How many virtual pages are *normal* pages?

   (c) What is the maximum size of an address space in bytes (use exponential notation for convenience, e.g., 2^3)?

   (d) Inode Page Tables reduce the lookup time for fast pages by one memory read operation. Do you think that this is an effective optimization? Briefly explain.

5. (8 pts) In lecture we said that, if the semaphore operations *Wait* and *Signal* are not executed atomically, then mutual exclusion may be violated. Assume that *Wait* and *Signal* are implemented as below:

```
void Wait (Semaphore S) {
  while (S.count <= 0) {}
  S.count = S.count - 1;
}

void Signal (Semaphore S) {
  S.count = S.count + 1;
}
```

Describe a scenario of context switches where two threads, T1 and T2, can both enter a critical section guarded by a single mutex semaphore as a result of a lack of atomicity.

6. (14 pts) Unix provides two mechanisms to link one file to another, *hard links* and *soft links*. With hard links, assume that the directory entry for the link maps the link file name to the *inode* for the file to which it is linked. With soft links, assume that the directory entry for the link maps the link file name to the *file name* of the file to which it is linked.

Consider the situation where we want to create a link "/bin/ls" to the existing file "/sbin/ls". In this case, "/bin/ls" is the link file name and "/sbin/ls" is the file to which it is linked. For the questions below, assume that, each time Unix has to retrieve information from the disk, it takes only one disk read operation.

(a) First, succinctly describe what steps Unix will take, in terms of the disk data structures it must read, in order to resolve the path name "/sbin/ls" so that it can read the first byte of the file.

(b) How many disk reads will be required to resolve the path name and read the first byte of the file "/sbin/ls"?

(c) Consider when we use a hard link to link "/bin/ls" to "/sbin/ls". Succinctly describe the steps required to read the first byte of "/sbin/ls" starting with the link "/bin/ls". How many disk reads will it require?

(d) Consider when we use a soft link to link "/bin/ls" to "/sbin/ls". Succinctly describe the steps required to read the first byte of "/sbin/ls" starting with the link "/bin/ls". How many disk reads will it require?

(e) Unix maintains a reference count of the number of hard links to a file, and it only removes a file's blocks on disk when this count reaches zero. If we remove the file "/bin/ls", is the hard link still valid? Is the soft link still valid?

7. (10 pts) Briefly describe the most challenging bug that your group encountered in project 3, how you found it, and how long it took you to find it. Which of your group members do you think will give the best answer to this question?

8. (18 pts) In this problem you will outline an implementation of shared memory in Nachos. Shared memory will be implemented using two new system calls:

```
RegionID SharedRegionCreate (int beginAddress, int endAddress);
     void SharedRegionAttach (RegionID region);
```

SharedRegionCreate creates a shared memory region in the caller's address space defined by the begin and end addresses. If successful, it returns a RegionID identifying the shared region. SharedRegion-Attach maps the existing shared region identified by the RegionID into the caller's address space. A region can only be created by one process, but it can be attached by an arbitrary number of processes. You can assume that each PTE has an additional flag *Shared*, which should be TRUE if that virtual page is being shared. You can also assume that a shared memory region can only be created within a defined region of a process' virtual address space.

These operations can be used as follows. A parent process uses SharedRegionCreate to create a shared memory region, and then Execs a child process and passes the returned RegionID for the shared region to the child as an argument to Exec. The child then uses SharedRegionAttach to map that region into its address space so that it can share memory with its parent. As a result, whatever data the parent places and modifies in the shared region, the child can access it (and vice versa).

For each of the following questions, answer them descriptively at a high level. The answers should be brief, and capture the essence of what needs to be implemented at each stage. You do not need to use psuedo-code.

(a) What should SharedRegionCreate do to the caller's page table?

(b) What should SharedRegionAttach do to the caller's page table?

(c) What *additional* page table operations must be performed when a shared page is paged out (evicted) from physical memory?

(d) What *additional* page table operations must be performed when a shared page is paged in from backing store?

(e) How should the destructor for AddrSpace be changed to account for shared pages?

(f) Name three error conditions that an implementation of SharedRegionCreate and SharedRegion-Attach will have to check for.

9. (10 pts) You get full credit for the problems below whether you answer them or not, or what your answer is. In other words, these questions will have no impact on your grade. **Do not** bother to write answers to these questions until you are finished with the rest of the exam, if you choose to do so at all.

    (a) What topic in this course struck you as the least interesting, least relevant, and made the least impact on your education?

    (b) What topic in this course was the most interesting to you?

    (c) What operating system topic were you hoping to learn about, but we didn't cover?

    (d) What message would you give to incoming students of the next CSE 120 class I teach? I will use some of these answers in the intro lecture next time.