Final Examination
CS 111
Fall 2015

Name: ______________________________

This is an open book, open note test. You may use electronic devices to take the test, but are not permitted to access the Internet during the test, except to get to the sections of the textbook found on line and to get copies of the lecture slides from the class web site. You have three hours to complete this. Please remember to put your name on all sheets of your answers.

There are 7 questions on the test, each on a separate page. You must answer 4 of them. You can choose any subset to answer. If you choose a problem, you must answer all of the questions on that page. Each problem you answer is worth 25% of the total points on the test.

You cannot answer more than four problems. If you start to answer questions on one page, and then decide that you would rather answer a different problem, clearly cross out the entire page. If you answer more than four problems I will only grade the first four.

You must answer every part of every chosen problem. Read each question CAREFULLY, make sure you understand EXACTLY what question is being asked and what type of answer is expected, and make sure that your answer clearly and directly responds to the asked question.

I am looking for depth of understanding and the ability to solve real problems. I want to see specific answers. Vague generalities will receive little or no credit (e.g., zero credit for an answer like "no, due to the relocation problem.").

None of these questions requires a long answer, but many of the questions may require you to do a lot of thinking and sketching before you come up with a reasonable answer. Feel free to use scratch paper to organize your thoughts. If the correct part of your answer is buried under a mountain of words, I may have trouble finding it.

1. One form of distributed system is a symmetric multiprocessor (SMP). Assume we have a 16 node SMP with shared memory and local caches for each node, with a bus manager and cache coherence hardware. We wish to run two large multi-process jobs on this machine simultaneously. Each job consists of 20-30 cooperating processes that share data structures between themselves and require some degree of synchronization.

a. Should we reserve a pool of the 16 nodes for each of the two jobs or should we dynamically share the entire set of 16 nodes? What information about the behavior of the processes comprising the jobs would allow you to better answer that question?
b. For whichever option you selected above, describe how scheduling should be performed. Will you maintain one scheduling queue for all processes on all nodes, one scheduling queue for each job, but covering all nodes for that job? One scheduling queue per node? One scheduling queue per job per node? Why is your choice best?
c. If you have already run job A's process 5 on node 13 for several time slices, how much effort should you take to run this process' next time slice on node 13? Why?

2. The following code is part of a file system that uses inodes, directories, and hard links in a manner similar to the Unix System V file system. Part of the code creates a new link to a file and part of it deletes an existing link to a file.

   a. Study the code and circle EXACTLY AND ONLY the instructions that are part of a critical section. Show the actual code changes you would make to reasonably and adequately protect the identified critical sections.
   b. Describe another common file system operation likely to be included in this file system that will probably contain a critical section that must be synchronized with those shown here. Why is this likely? How would you go about finding that critical section in the code for this other operation?

```
struct inode         /* The in-memory data structure for an inode.  NOTE:
not all fields listed. */
{
          short i_mode;    /* ACL info*/
          short i_uid; /* owner's user ID */
. . .
          short i_links_count;    /* number of hard links */
. . .
}
create_name(dir,file_name,inode_num, inode_ptr)   /* Add a file name to
a directory */
          directory_pointer *dir;
          char *file_name;
          int inode_num;
          inode *inode_ptr;
{
          if (check_name_validity(file_name) != 0)   /*Make sure name is
valid.   Zero returned if valid. */
          {
                    error("illegal name\n");
                    return (-1);
          }
          add_to_directory(dir,file_name, inode_num);
          inode_ptr->i_links_count++;
          return (0);
}
remove_name(dir,file_name,inode_num,inode_ptr) /*Remove a file name
from a directory */
          directory_pointer *dir;
          char *file_name;
          int inode_num;
          inode *inode_ptr;
{
          if (remove_from_directory(dir,file_name) == 0) /*Name removed
from directory if return code is 0 */
          {
                    if ((inode_ptr->i_links_count-- == 0)){
                         free_data_blocks(inode_ptr);
                         update_free_inode_bitmap(inode_num);
                    }
                    return (0);
          }  else
                    return (-1);
}
```

3. Most security problems seen in modern systems arise out of flaws in applications, not flaws in the operating system. Attackers exploit these application flaws and are then able to use the privileges given to the application to perform malicious actions. While not the operating system's fault, perhaps the operating system can help solve the problem. One possible operating system-supported approach would be to allow fine grained access control, such that each process is given access rights to exactly what it needs to access and nothing more. Another possible OS-supported approach would be to run intrusion detection in the operating system, observing aspects of the behavior of processes with the intention of detecting when attackers have compromised and exploited them.

a. Discuss the advantages and disadvantages of these two approaches, including discussion of their relative security, the overhead involved, and the range of security problems they can address.
b. What changes would be required in a typical commercial operating system to support each of these two alternatives?
c. Argue in favor of one of the two alternatives.

4. Each of the following resources has the potential to create or contribute to a deadlock, livelock or hang situation. For each resource, describe the best avoidance or prevention technique to use to eliminate this possibility. Make sure you tell me why you chose each technique, and exactly how you would apply it to solving the problem.

a. Eight critical sections, each controlled by a distinct mutual-exclusion lock. Some operations require the simultaneous holding of multiple locks, but the operations are such that it is difficult to predict (at the beginning of an operation) which of the eight locks a process will need to hold in order to complete that operation.
b. Swap space. If there is not room on the swap device to swap out one of the in-memory processes, we will be unable to make room to swap in and run any of the processes that are currently swapped out.
c. Records in a database, where most operations involve complex update transactions for multiple records (merchandise, inventory, shipping, invoices, order status, etc).
d. A distributed system offers resources to remote systems, and allows them to be locked. A central lock manager on one machine grants and manages all locks on remote resources. However, each computer can also issues locks on its own local resources for local processes.

5. A system has been in use and performing well for many months. The response time has recently become terrible (5-10x what it was a few weeks ago) for a large number of applications. The application mix and load have indeed evolved over the last few months, but the strange thing is that the CPU, disk, and network interface still seem to be lightly loaded (50% idle).

a. What general problem have we discussed that might explain this situation?
b. Suggest a LIKELY SPECIFIC thing that could be causing the problem in this system.
c. What measurements or experiments could you conduct to confirm or refute this diagnosis?
d. Assuming the measurements and experiments confirm your hypothesis, what SPECIFICALLY could you do, in this instance, to eliminate the problem?

6. Consider a diskless machine that is only expected to run when connected to a network. Since it is diskless, all file access is via a remote file system, but we also want to run demand paging on this machine to provide virtual memory. Lacking a disk, we will demand page across the network, with some other machine storing pages not currently in the RAM of the diskless machine. The diskless machine does have a suitable MMU.

a. Describe the operations that must be performed when there is a page fault on the diskless machine.

b. What special challenges will this memory management system face that a demand paging system backed by a local disk would not face? How would you handle those challenges?

c. Assume you are choosing between this system and a more traditional disk-based system supporting demand paging. Write an equation for each of the two systems that will describe the expected paging latency you would see from the systems. How would you obtain the necessary measurements to plug into these equations to allow for the desired comparison?

7. IPC can run into problems with flow control when the writer gets too far ahead of the reader. There are two general approaches for dealing with this situation: blocking the writer, or flushing written data out of the buffer.

(a) If we blocked a writer, and the reader never resumes, what problems might occur? How could we reasonably recover from this?

(b) If we flushed some data, and then the reader catches up again, why is this a problem? How could we reasonably recover from this?

(c) If the reader was a service that failed, but was restarted, how might it be possible for the client to be redirected to the new server and continue his session?