CPS 110 Final Exam

Fall 1997

You have three hours, but the exam is written to be doable in two hours. My goals with this exam are to (1) give you credit for what you know, (2) give the most credit to students who know the most, and (3) deliver my grades to the registrar on time without blowing my weekend. Please keep your answers terse but complete: no wasted words, emphasize content over style, etc. Precise and/or quantitative answers (where appropriate) are more interesting than vague generalizations. Some questions are harder than others, but if you can't answer the whole question then make the best start you can. Grading blank anwers is easy but not much fun.

The exam has three parts worth 80, 100, and 90 points respectively. You get 5 points for signing your name on every page you turn in. You get 25 points as a holiday gift. 300 points total.

Part 1: Shorts

Answer all but one of the questions in this section. Each answer should consist of a single paragraph and possibly a diagram. You should spend about five minutes on each question. Each answer is worth 10 points (80 points total).

1. How can an advanced Unix virtual memory system efficiently handle loading and paging of the initialized static data segment? Be sure to consider the handling of shared executables.
2. Explain the difference between a hard link and a soft (symbolic) link in a Unix file system. Consider both the representation on disk and the semantics of the *unlink* system call (used by the *rm* command).
3. How does increasing the virtual memory page size affect paging performance?
4. How does increasing the rotation speed of a magnetic disk affect the average time to access a random block?
5. What is a "capability"? Why would I want one? Why might I want to give one to you?
6. What is programmed I/O (PIO)? What is Direct Memory Access (DMA)? Compare and contrast PIO and DMA with respect to performance.
7. What is "double buffering" and how does it improve I/O bandwidth?
8. Under what circumstances can the file system avoid reading a file block from disk in order to satisfy a *write* request? Hint: there is more than one case.
9. A computer with a 32-bit address uses a two-level page table. Virtual addresses are split into a 9-bit top-level page table field, an 11-bit second level page table field, and an offset. How large are the pages and how many are there in the address space?

Part 2: Longer Questions

Answer all but one of the questions in this section. Where code is needed, any kind of pseudocode is OK as long as its meaning is clear. Each answer is worth 25 points (100 total).

10. In class we discussed synchronization using semaphores and mutex/condition variable pairs. Are these facilities equivalent in power, i.e., is there any synchronization problem that can be solved with either facility that cannot be solved with the other? Prove your answer.

11. Tweedledum and Tweedledee are separate threads executing their respective procedures. The code below is intended to cause them to forever take turns exchanging insults through the shared variable X in strict alternation. The ***Sleep()*** and ***Wakeup()*** routines operate as discussed in class: ***Sleep*** blocks the calling thread, and ***Wakeup*** unblocks a specific thread if that thread is blocked, otherwise its behavior is unpredictable (like Nachos ***Scheduler::ReadyToRun***).

```
void
Tweedledum()
{
    while(1) {
        Sleep();
        x = Quarrel(x);
        Wakeup(Tweedledee thread);
    }
}
```

```
void
Tweedledee()
{
    while(1) {
        x = Quarrel(x);
        Wakeup(Tweedledum thread);
        Sleep();
    }
}
```

a) The code shown above exhibits a well-known synchronization flaw. Outline a scenario in which this code would fail, and the outcome of that scenario. Regurgitate the buzzword for this synchronization flaw.

b) Show how to fix the problem by replacing the ***Sleep*** and ***Wakeup*** calls with semaphore ***P*** (down) and ***V*** (up) operations. No, you may not disable interrupts.

c) Implement Tweedledum and Tweedledee using a mutex and condition variable.

12. The Global Memory Service (GMS) remote paging system runs using the Trapeze network interface as discussed in class. Suppose that GMS/Trapeze reduces the average stall time for a page fault by a factor of 50, from 10 milliseconds to 200 microseconds (not including system CPU overhead to handle the page fault). Suppose further (incorrectly for now) that the system CPU overhead to handle a remote page fault is equal to the system CPU overhead to handle a page fault from local disk. What effect will remote paging have on application performance? Be specific and quantitative, and explain your answer.

13. Advanced file systems can generally deliver full disk bandwidth for streams of sequential writes, as well as reads. Explain three techniques that systems use to deliver full disk bandwidth for writes. What is the role of the block buffer cache in implementing these optimizations?

14. Explain the difference between internal fragmentation and external fragmentation, and discuss how they occur in each of the following system functions: memory allocation using paging, memory allocation using variable partitioning or pure segmentation, Unix (UFS) file allocation, and heap managers (e.g., library routines for *malloc*/*free* or *new*/*delete*).

Part 3: The Return of Nachos

Answer all questions in this section. Each answer is worth 30 points (90 points total).

15. Discuss how your group's Nachos kernel uses bitmaps. For each use of bitmaps, explain why you chose to use bitmaps instead of some other structure.

16. Outline the steps taken by your group's Nachos kernel to handle an *exit* system call.

17. Describe the most interesting bug your group encountered during the semester. (Note: for purposes of this question, having to hit *ctrl-c* to exit Nachos is a feature, not a bug.) Which of your group members do you think will give the best answer to this question?

Thank you and have a safe and enjoyable holiday.