Systems Qualifying Exam — Operating Systems

May 1999

The objective of the exam is to find out what you know about operating systems and to assess your ability to identify and develop solutions for problems that arise in operating systems. Note that the questions have many "right" answers, so be sure to justify your answers. In particular, state any assumptions you make. Point form answers are acceptable. Grammar optional.

Problem 1 — Protection and Performance (20 points)

The Capabilities-R-Us mega-corporation wants to make virtual memory obsolete. They ask you to implement a system based on fine-grained capabilities. On this machine, processes execute in physical memory and can issue loads and stores to any physical memory location. To support protection, each byte of physical memory has a corresponding read and write capability, and every process has a capability list (stored as an in-memory array) holding the capabilities it owns. On every memory operation, the hardware checks that the specified memory's capability matches some entry in the process's capability list. The instructions to switch a capability list and to set a byte's capabilities are privileged.

1. Sketch how to enforce protection on this system. In particular, say when the capability list must be switched and the different events that cause the OS to set a byte's capabilities.

2. Sketch how the OS could use the memory system to eliminate the requirement that the "set capability" instruction be privileged. You may need to modify the hardware (slightly) to do so. Be sure to handle the case that a malicious application generates code at runtime.

3. While one use of virtual memory is to provide protection, it has other applications. Give two reasons that you might still want virtual memory.

4. Since the money is good, you keep your doubts to yourself and build the machine. However, you notice it spends a lot of time doing capability lookups. Sketch how to solve this problem using a hardware mechanism based on a traditional TLB.

Problem 2 — Caching fun (10 points)

Give two advantages of client caching over server caching in a distributed file system. In a few sentences, state two problems client caching creates, and give a simple fix for each or state why such a fix is impossible.

Problem 3 — File system atomicity (20 points)

The Happy-go-lucky file system (HFS) creates a new file by:

1. allocating a disk block to hold the file's inode from its free list;
2. inserting the file-name-to-inode mapping in a free directory entry;
3. returning to the user;
4. at some later point HFS writes the cached copies of the modified directory block, the inode, and the free list to disk (in no particular order).

Recall that disks can only atomically write a sector at a time.

1. The system crashes during file creation. Give two possible file system inconsistencies that you might see.
2. Briefly describe how to use synchronous disk writes and file system reconstruction to eliminate these two inconsistencies. (Recall, "synchronous disk write" means when the OS issues a disk write, it waits until the write completes before performing another disk operation.)
3. You notice that your system clock has 30 bytes of non-volatile RAM (NVRAM). You decide time is for fools, disable the clock, and use its NVRAM to hold a miniature "write-ahead log." Sketch how to use this log to eliminate the problems you pointed out in Part 1. Be specific about what you store in the log, and how and when you use this information.

   Will this approach be faster than the use of use of synchronous writes? Why or why not?

Problem 4 — Predicting the future (10 points)

The past predicts the future (frequently). For example, yesterday's weather predicts today's with good accuracy and the department will probably give an OS qualifying exam next year.

1. Give two examples of how an operating system uses this heuristic to improve performance.
2. Assume the OS can predict the future perfectly. In a few sentences, state how this fact would change the traditional OS implementation of one of your examples. (For example, how the implementation could be made simpler, more aggressive, etc.)