Final Examination
Operating Systems

Jan. 4, 2010

1. (9%) Explain the following terms.
   (a) TLB reach (b) priority paging
   (c) unified buffer cache
2. (7%) Which of the following programming techniques and structures are "good" for a demand-paged environment? Which are "not good"?
   (a) Stack (b) Hashed symbol table
   (c) Sequential search (d) Binary search
   (e) Pure code (f) Vector operations
   (g) Indirection
3. (4%) On a system with paging, a process cannot access memory that it does not own. Why? How could the OS allow access to other memory? Why should it or should it not?
4. (4%) Consider a system that provides support for user-level and kernel-level threads. The mapping in this system is one to one. Does a multithreaded process consists of (a) a working set for the entire process or (b) a working set for each thread? Explain.
5. (4%) Is it possible for a process to have two working sets, one representing data and another representing code? Explain.
6. (6%) Show that the FIFO page replacement algorithm is not a stack algorithm.
7. (6%) Plot the diagram of the layered file system (six layers).
8. (6%) The disadvantage of the linked list allocation can be eliminated by taking the pointer from each disk block and putting in a FAT table in memory. Give one advantage and two disadvantages of using FAT.
9. (6%) A group of OS designers for the Frugal Computer Company are thinking about ways to reduce the amount of backing store needed in their new OS. The head guru has just suggested not bothering to save the program code in the swap area at all, but just page it in directly from the binary file whenever it is needed. Under what conditions, if any, does this idea work for the program code? Under what conditions, if any, does it work for the data?

10. (9%) Given memory partitions of 100K, 500K, 200K, 300K, and 600K (in order), how would each of the first-fit, best-fit, and worst-fit algorithms place processes of 212K, 417K, 112K, and 426K (in order)? Which algorithm makes the most efficient use of memory?

11. (12%) A page-replacement algorithm should minimize the number of page faults. We can do this minimization by distributing heavily used pages evenly over all of memory, rather than having them compete for a small number of page frames. We can associate with each page frame a counter of the number of pages that are associated with that frame. Then, to replace a page, we search for the page frame with the smallest counter.

    (a) (4%) Define a page-replacement algorithm using this basic idea. Specifically address the problems: (1) what is the initial value of the counters? (2) When are counters increased? (3) When are counters decreased? (4) How is the page to be replaced selected.

    (b) (4%) How many page faults occur for your algorithm for the following reference string, for four page frames?

    1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2, 9.

    (c) (4%) What is the minimum number of page faults for an optimal page replacement strategy for the reference string in part (b) with four page frames?

12. (8%) Suppose that the page reference string contains repetitions of long sequences of page references followed occasionally by random page reference. For example, the sequence: 0, 1, …, 511,*431*, 0, 1, …, 511, *332*, 0, 1, … consists of repetitions of the sequence 0, 1, …, 511 followed by a random reference to pages *431* and *332*.

    (a) Why won't the standard replacement algorithms (LRU, FIFO, and Second Chance) be effective in handling this workload for a page allocation that is less than the sequence length?

    (b) If this program were allocated 500 page frames, describe a page replacement approach that would perform much better than the LRU, FIFO, or Second Chance algorithms.

13. (6%) Assume we have a demand-paged memory. The page table is held in registers. It takes 8 milliseconds to service a page fault if an empty page is

available or the replaced page is not modified and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds. Assume that the page to be replaced is modified 70 percent of the time. What is the maximum acceptable page-fault rate for an effective access time of no more than 200 nanoseconds?

14. (6%) Consider the parameter Δ used to define the working-set window in the working-set model. What is the effect of setting Δ to a small value on the page fault frequency and the number of active (non-suspended) processes currently executing in the system? What is the effect when Δ is set to a very high value?

15. (6%) Discuss situations in which the least frequently used page-replacement algorithm generates fewer page faults then the least recently used page-replacement algorithm. Also discuss under what circumstances the opposite holds.

16. (6%) Consider a file system that uses inodes to represent files. Disk blocks are 8 KB in size, and a pointer to a disk block requires 4 bytes. This file system has 12 direct disk blocks, as well as single, double, and triple indirect disk blocks. What is the maximum size of a file that can be stored in this file system?

17. (6%) Assume that you have a page-reference string for a process with *m* frames (initially all empty). The page-reference string has length *p*; *n* distinct page numbers occur in it. Answer these questions for any page replacement:
    (a) What is a lower bound on the number of page faults?
    (b) What is an upper bound on the number of page faults?

18. (8%) In a virtual memory system, if we have no fault, we access a location in time *t*, and if we have a fault, in time *T*. Assume that the probability of fault is *p*.
    (a) Determine the mean time between faults.
    (b) Determine the average access time to the location.