4190.307 Operating Systems, Fall 2015

Problem Set 2

Part 3 Memory management
- Chapter 8 Memory-management strategies
- Chapter 9 Virtual-memory management

Part 4 Storage management
- Chapter 10 File system
- Chapter 11 Implementing file-systems
- Chapter 12 Mass-storage structure
- Chapter 13 I/O systems

Part 5 Protection and security
- Chapters 14, 15 (System security)

The problems provided in this file are excerpted from the exercises at the end of each chapter in the textbook. It is suggested that each student review these problems, especially to prepare himself/herself for the scheduled exams

8. Memory-management strategies

8.5 Compare the memory organization schemes of contiguous memory allocation, pure segmentation, and pure paging with respect to the following issues:

a. External fragmentation

b. Internal fragmentation

c. Ability to share code across processes

8.8 Although Android does not support swapping on its boot disk, it is possible to set up a swap space using a separate SD nonvolatile memory card. Why would Android disallow swapping on its boot disk yet allow it on a secondary disk?

8.12 Assuming a 1-KB page size, what are the page numbers and offsets for the following address references (provided as decimal numbers):

a. 3085 b. 42095 c. 215201 d. 650000 e. 2000001

8.16 Consider a computer system with a 32-bit logical address and 4-KB page size. The system supports up to 512 MB of physical memory. How many entries are there in each of the following?

a. A conventional single-level page table

b. An inverted page table

8.19 Explain why sharing a reentrant module is easier when segmentation is used than when pure paging is used.

8.24 Consider the Intel address-translation scheme shown in Figure 8.22.

a. Describe all the steps taken by the Intel Pentium in translating a logical address into a physical address.

b. What are the advantages to the operating system of hardware that provides such complicated memory translation?

c. Are there any disadvantages to this address-translation system? If so, what are they? If not, why is this scheme not used by every manufacturer?

9. Virtual-memory management

9.2 A simplified view of thread states is Ready, Running, and Blocked, where a thread is either ready and waiting to be scheduled, is running on the processor, or is blocked (for example, waiting for I/O). This is illustrated in Figure 9.30. Assuming a thread is in the Running state, answer the following questions, and explain your answer:

a. Will the thread change state if it incurs a page fault? If so, to what state will it change?

b. Will the thread change state if it generates a TLB miss that is resolved in the page table? If so, to what state will it change?

c. Will the thread change state if an address reference is resolved in the page table? If so, to what state will it change?

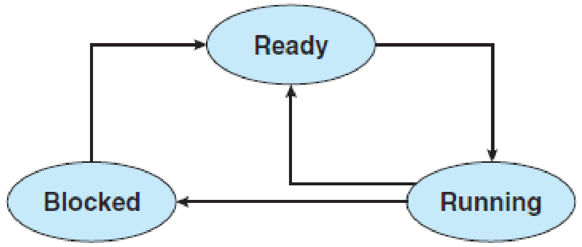

Figure 9.30

9.4 What is the copy-on-write feature, and under what circumstances is its use beneficial? What hardware support is required to implement this feature?

9.5 A certain computer provides its users with a virtual memory space of 232 bytes. The computer has 222 bytes of physical memory. The virtual memory is implemented by paging, and the page size is 4,096 bytes. A user process generates the virtual address 11123456. Explain how the system establishes the corresponding physical location. Distinguish between software and hardware operations.

9.6 Assume that 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 frame is available or if 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?

9.8 Consider the following page reference string:

7, 2, 3, 1, 2, 5, 3, 4, 6, 7, 7, 1, 0, 5, 4, 6, 2, 3, 0 , 1.

Assuming demand paging with three frames, how many page faults would occur for the following replacement algorithms?

- LRU replacement • FIFO replacement • Optimal replacement

9.17 A page-replacement algorithm should minimize the number of page faults. We can achieve 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 associated with that frame. Then, to replace a page, we can search for the page frame with the smallest counter.

a. Define a page-replacement algorithm using this basic idea. Specifically address these problems:

   i. What is the initial value of the counters?

   ii. When are counters increased?

   iii. When are counters decreased?

   iv. How is the page to be replaced selected?

9.19 What is the cause of thrashing? How does the system detect thrashing? Once it detects thrashing, what can the system do to eliminate this problem?

9.21 Consider the parameter used to define the working-set window in the working-set model. When is set to a small value, what is the effect on the page-fault frequency and the number of active (nonsuspended) processes currently executing in the system? What is the effect when is set to a very high value?

10. File system

10.2 The open-file table is used to maintain information about files that are currently open. Should the operating system maintain a separate table for each user or maintain just one table that contains references to files that are currently being accessed by all users? If the same file is being accessed by two different programs or users, should there be separate entries in the open-file table? Explain.

10.4 Provide examples of applications that typically access files according to the following methods:

- Sequential
- Random

10.6 If the operating system knew that a certain application was going to access file data in a sequential manner, how could it exploit this information to improve performance?

10.11 What are the implications of supporting UNIX consistency semantics for shared access to files stored on remote file systems?

11. Implementing file-systems

11.4 Consider a system where free space is kept in a free-space list.

a. Suppose that the pointer to the free-space list is lost. Can the system reconstruct the free-space list? Explain your answer.

b. Consider a file system similar to the one used by UNIX with indexed allocation. How many disk I/O operations might be required to read the contents of a small local file at /a/b/c? Assume that none of the disk blocks is currently being cached.

c. Suggest a scheme to ensure that the pointer is never lost as a result of memory failure.

11.5 Some file systems allow disk storage to be allocated at different levels of granularity. For instance, a file system could allocate 4 KB of disk space as a single 4-KB block or as eight 512-byte blocks. How could we take advantage of this flexibility to improve performance? What modifications would have to be made to the free-space management scheme in order to support this feature?

11.7 Consider a file system on a disk that has both logical and physical block sizes of 512 bytes. Assume that the information about each file is already in memory. For each of the three allocation strategies (contiguous, linked, and indexed), answer these questions:

a. How is the logical-to-physical address mapping accomplished in this system? (For the indexed allocation, assume that a file is always less than 512 blocks long.)

b. If we are currently at logical block 10 (the last block accessed was block 10) and want to access logical block 4, how many physical blocks must be read from the disk?

12. Mass-storage structure

12.2 Explain why SSDs often use an FCFS disk-scheduling algorithm.

12.3 Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4,999. The drive is currently serving a request at cylinder 2,150, and the previous request was at cylinder 1,805. The queue of pending requests, in FIFO order, is:

2,069, 1,212, 2,296, 2,800, 544, 1,618, 356, 1,523, 4,965, 3681

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms?

a. FCFS b. SSTF c. SCAN d. LOOK e. C-SCAN f. C-LOOK

12.10 Compare the throughput achieved by a RAID level 5 organization with that achieved by a RAID level 1 organization for the following:

a. Read operations on single blocks

b. Read operations on multiple contiguous blocks

12.15 Discuss the reasons why the operating system might require accurate information on how blocks are stored on a disk. How could the operating system improve file-system performance with this knowledge?

13. I/O systems

13.3 Consider the following I/O scenarios on a single-user PC:

a. A mouse used with a graphical user interface

b. A tape drive on a multitasking operating system (with no device preallocation available)

c. A disk drive containing user files

d. A graphics card with direct bus connection, accessible through memory-mapped I/O

For each of these scenarios, would you design the operating system to use buffering, spooling, caching, or a combination? Would you use polled I/O or interrupt-driven I/O? Give reasons for your choices.

13.4 In most multiprogrammed systems, user programs access memory through virtual addresses, while the operating system uses raw physical addresses to access memory. What are the implications of this design for the initiation of I/O operations by the user program and their execution by the operating system?

13.5 What are the various kinds of performance overhead associated with servicing an interrupt?

13.8 Some DMA controllers support direct virtual memory access, where the targets of I/O operations are specified as virtual addresses and a translation from virtual to physical address is performed during the DMA. How does this design complicate the design of the DMA controller? What are the advantages of providing such functionality?

14/15. System protection, security

14.5 Discuss the strengths and weaknesses of implementing an access matrix using access lists that are associated with objects.

14.6 Discuss the strengths and weaknesses of implementing an access matrix using capabilities that are associated with domains.

15.1 Buffer-overflow attacks can be avoided by adopting a better programming methodology or by using special hardware support. Discuss these solutions.

15.4 The list of all passwords is kept within the operating system. Thus, if a user manages to read this list, password protection is no longer provided. Suggest a scheme that will avoid this problem. (Hint: Use different internal and external representations.)

15.11 What commonly used computer programs are prone to man-in-the middle attacks? Discuss solutions for preventing this form of attack.

15.15 Consider a system that generates 10 million audit records per day. Assume that, on average, there are 10 attacks per day on this system and each attack is reflected in 20 records. If the intrusion-detection system has a true-alarm rate of 0.6 and a false-alarm rate of 0.0005, what percentage of alarms generated by the system correspond to real intrusions?