CSCI 4320 (Principles of Operating Systems), Fall 2003

Homework 4

Assigned: November 4, 2003.

Due: November 11, 2003, at 5pm.

Credit: 40 points.

> *Note:* The HTML version of this document may contain hyperlinks. In this version, hyperlinks are represented by showing both the link text, formatted like this, and the full URL as a footnote.

Be sure you have read chapter 4.

1 Problems

Do the following problems. You may write out your answers by hand or using a word processor or other program, but please submit hard copy, either in class or in my mailbox in the department office.

1. (6 points) Consider a computer system that uses variable partitions and swapping to manage memory (as described in chapter 4 pages 197–198); suppose it has 10,000 bytes of main memory, currently allocated as shown by the following table. (In this problem, amounts and addresses are in decimal, and unrealistically small, to keep the arithmetic simple.)

Locations	Contents
7500 – 9999	free
4500 – 7499	process B
4000 – 4499	free
3000 – 3999	process A
2000 – 2999	free
0 – 1999	operating system

   Answer the following questions:

   (a) Suppose we now need to allocate space first for process C, which needs 400 bytes, and then for process D, which needs 200 bytes. At what addresses (locations) will memory be allocated for these two processes if we use

      i. a best-fit strategy for allocating memory?

      ii. a worst-fit strategy for allocating memory?

      iii. a first-fit strategy for allocating memory?

   (b) Suppose the MMU for this system uses the simple base register / limit register scheme described in chapter 1 of the textbook (pages 26–27).

      i. What value would need to be loaded into the base register if we performed a context switch to restart process B?

ii. What memory locations would correspond to the following virtual (program) addresses in process B?

- 100
- 4000

2. (10 points) Now consider a computer system using paging to manage memory; suppose it has 64K (2^{16}) bytes of memory and a page size of 4K bytes, and suppose the page table for some process (call it process A) looks like the following.

Page number	Present/absent bit	Page frame number
0	1	5
1	1	4
2	1	2
3	0	?
4	0	?
5	1	7

Answer the following questions about this system.

(a) If the only processes in main memory were process A and an operating system using page frame 0, how many free page frames would there be?

(b) How many additional page table entries would be required to give process A an address space of 64K bytes?

(c) How many bits are required to represent physical addresses (memory locations) on this system? If each process has a maximum address space of 64K bytes, how many bits are required to represent virtual (program) addresses?

(d) What memory locations would correspond to the following virtual (program) addresses for process A? (Here, the addresses will be given in hexadecimal, i.e., base 16, to make the needed calculations simpler. Your answers should also be in hexadecimal. Notice that if you find yourself converting between decimal and hexadecimal, *you are doing the problem the hard way.* Stop and think whether there is an easier way.)

- 0x1420
- 0x2ff0
- 0x4008
- 0x0010

(e) If we want to guarantee that this system could support 16 concurrent processes and give each an address space of 64K bytes, how much disk space would be required for storing out-of-memory pages? Justify your answer. (If your answer depends on making additional assumptions, state what they are — e.g., you might assume that the operating system will always use the first page frame of memory and will never be paged out.)

(f) Would it be possible on this system to give each process a maximum address space of 1M (2^{20}) bytes? Briefly explain your answer.

3. (6 points) Consider a small computer system with only four page frames and address spaces consisting of eight pages. Suppose we start out with all page frames empty (pure demand paging) and run a program that references pages in the following order:

$$0, 1, 7, 2, 3, 2, 7, 1, 0, 3$$

(That is, its first reference to memory is in page number 0, its second reference to memory is in page number 1, etc.) If FIFO page replacement is being used, how many page faults will occur during the running of this program?

4. (6 points) Consider another small computer system with only four page frames. Suppose you have implemented the aging algorithm for page replacement, using 4-bit counters and updating the counters after every clock tick, and suppose the R bits for the four pages are as follows after the first four clock ticks.

Time	R bit (page 0)	R bit (page 1)	R bit (page 2)	R bit (page 3)
after tick 1	0	1	1	1
after tick 2	1	0	1	0
after tick 3	1	0	1	1
after tick 4	1	1	0	1

What are the values of the counters (in binary) for all pages after these four clock ticks? If a page needed to be removed at that point, which page would be chosen for removal?

5. (8 points) Now consider a much bigger computer system, one in which addresses (both physical and virtual) are 32 bits and the system has 2^{32} bytes of memory. (You can express your answers in terms of powers of 2, if that is convenient.)

   (a) What is the maximum size in bytes of a process's address space on this system?

   (b) Is there a logical limit to how much main memory this system can make use of? That is, could we buy and install as much more memory as we like, assuming no hardware constraints?

   (c) If page size is 4K (2^{12}) and each page table entry consists of a page frame number and four additional bits (present/absent, referenced, modified, and read-only), how much space is required for each process's page table? If we allow a maximum of 64 (2^6) processes, how much space in total is required for all their page tables? (You should express the size of each page table entry in bytes, not bits, assuming 8 bits per byte and rounding up if necessary.)

   (d) Suppose instead we use a single inverted page table (as described in chapter 4, pages 213–214), in which each entry consists of a page number, a process ID, and four additional bits (free/in-use, referenced, modified, and read-only), and we allow at most 64 processes, how much space is needed for this inverted page table? (You should express the size of each page table entry in bytes, not bits, assuming 8 bits per byte and rounding up if necessary.)

6. (4 points) A particular system using demand paging and observed to be running slowly is monitored. It is discovered that the CPU is in use about 20% of the time, the paging disk is in use about 98% of the time, and other disks are in use about 5% of the time. For each of the following, say whether it would be likely to increase CPU utilization and why.

   (a) Installing a faster CPU.

   (b) Installing a larger paging disk.

   (c) Increasing the number of processes (degree of multiprogramming).

   (d) Decreasing the number of processes (degree of multiprogramming).

(e) Installing more main memory.

(f) Installing a faster paging disk.