Homework Assignment #3

J. H. Wang

Apr. 24, 2023

Homework #3

- Chap.7: 7.6, 7.12, 7.15
- Chap.8: 8.5, 8.9, 8.13
- Chap.9: 9.6, 9.8, 9.17, 9.19
- Programming exercises
  - Programming problems: 7.17*, 8.25*, 9.26*
    - Note: Each student must complete all programming problems on your own
  - Programming projects for Chap. 7* and Chap. 9*
    - Team-based: At least one selected programming project
- Due: three weeks (May 15, 2023)

- Chap 7:
  - 7.6: In a real computer system, neither the resources available nor the demands of processes for resources are consistent over long periods (months). Resources break or are replaced, new processes come and go, and new resources are bought and added to the system.
  - If deadlock is controlled by the banker's algorithm, which of the following changes can be made safely (without introducing the possibility of deadlock), and under what circumstances?

- (a) Increase Available (new resources added).
- (b) Decrease Available (resource permanently removed from system).
- (c) Increase Max for one process (the process needs or wants more resources than allowed).
- (d) Decrease Max for one process (the process decides that it does not need that many resources).
- (e) Increase the number of processes.
- (f) Decrease the number of processes.

- 7.12: Consider the following snapshot of a system :

	Allocation A B C D	Max A B C D
P0	3 0 1 4	5 1 1 7
P1	2 2 1 0	3 2 1 1
P2	3 1 2 1	3 3 2 1
P3	0 5 1 0	4 6 1 2
P4	4 2 1 2	6 3 2 5

(to be continued…)

(… continued from the previous slide)

Use the *banker's algorithm*, determine whether or not each of the following states is unsafe. If the state is safe, illustrate the order in which the processes may complete. Otherwise, illustrate why the state is unsafe.

(a) Available=(0,3,0,1)

(b) Available=(1,0,0,2)

- 7.15: A single-lane bridge connects the two Vermont villages of North Tunbridge and South Tunbridge. Farmers in the two villages use this bridge to deliver their produce to the neighbor town.
- The bridge can become deadlocked if a northbound and a southbound farmer get on the bridge at the same time. (Vermont farmers are stubborn and are unable to back up.)

- Using semaphores and/or mutex locks, design an algorithm in pseudocode that prevents deadlock.
- Initially, do not be concerned about starvation (the situation in which northbound farmers prevent southbound farmers from using the bridge, or vice versa).

- Chap. 8:
  - 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
  - 8.9: Compare paging with segmentation with respect to how much memory the address translation structures require to convert virtual addresses to physical addresses.

- 8.13: The BTV operating system has a 21-bit virtual address, yet on certain embedded devices, it has only a 16-bit physical address. It also has a 2KB page size. How many entries are there in each of the following?
- (a) A conventional, single-level page table
- (b) An inverted page table

- Chap. 9:
  - 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.
  Assume demand paging with three frames, how many page faults would occur for the following replacement algorithms?
  (a) LRU replacement
  (b) FIFO replacement
  (c) 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.
- (… to be continued)

- (continued from the previous slide…)
- (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?
- (b) How many page faults occur for your algorithm for the following reference string with 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.
- (c) 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?

- 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?

Programming Problems

- 7.17*: Implement your solution to Exercise 7.15 using POSIX synchronization (or Java if you want).
- In particular, represent northbound and southbound farmers as separate threads.
- Once a farmer is on the bridge, the associated thread will sleep for a random period of time, representing traveling across the bridge.
- Design your program so that you can create several threads representing the northbound and southbound farmers.

(Exercise 7.15 for your reference)

- A single-lane bridge connects the two Vermont villages of North Tunbridge and South Tunbridge.
- Farmers in the two villages use this bridge to deliver their produce to the neighboring town.
- The bridge can become deadlocked if a northbound and a southbound farmer get on the bridge at the same time. (Vermont farmers are stubborn and are unable to back up.)
- Using semaphores and/or mutex locks, design an algorithm in pseudocode that prevents deadlock.
- Initially, do not be concerned about starvation (the situation in which northbound farmers prevent southbound farmers from using the bridge, or vice versa).

- 8.25*: Assume that a system has a 32-bit virtual address with a 4-KB page size. Write a program that is passed a virtual address (in decimal) on the command line and have it output the page number and offset for the given address.

  (to be continued…)

- (… continued from the previous slide)

  As an example, your program would run as follows:

  ./a.out 19986

  Your program would output:

  The address 19986 contains:
  page number=4
  offset=3602

  Writing this program will require using the appropriate data type to store 32 bits. We encourage you to use unsigned data types as well.

- 9.26*: Write a program that implements the FIFO, LRU, and optimal page-replacement algorithms presented in this chapter. First, generate a random page-reference string where page numbers range from 0 to 9. Apply the random page-reference string to each algorithm, and record the number of page faults incurred by each algorithm. Implement the replacement algorithm so that the number of page frames can vary from 1 to 7. Assume that demand paging is used.

Programming projects – Choose one from Ch.7 or Ch.9

- Ch.7*: Banker's Algorithm
  - Write a multithreaded program that implements the *banker's algorithm* discussed in Section 7.5.3. This assignment combines three topics: (1) multithreading (2) preventing race conditions (3) deadlock avoidance.
  - Create n customer threads that request and release resources from the bank. The customers will continually loop, requesting and then releasing random numbers of resources. The banker will grant a request if it satisfies the safety algorithm.
  - Since multiple threads will concurrently access shared data, access must be controlled through mutex locks to prevent race conditions.
  - You should invoke your program by passing the number of resources of each type on the command line.

- Ch.9*: Designing a Virtual Memory Manager
  - This project consists of writing a program that translates logical to physical addresses for a virtual address space of size 2^{16}=65,536 bytes.
  - Your program will read from a file containing logical addresses and, using a TLB as well as a page table, will translate each logical address to its corresponding physical address and output the value of the byte stored at the translated physical address.

- The test file *addresses.txt* contains several 32-bit integer numbers that represent logical addresses. However, you need only be concerned with 16-bit addresses, so you must mask the rightmost 16 bits of each logical addresses. These 16 bits are divided into (1) an 8-bit page number and (2) 8-bit page offset.
- Your program will implement demand paging. The backing store is represented by the file BACKING_STORE.bin, a binary file of size 65,536 bytes

Any Questions or Comments?