Homework Assignment #3

J. H. Wang

Apr. 25, 2024

Homework #3

- Chap.7: 7.8
- Chap.8: 8.20, 8.27(a)(b), 8.30
- Chap.9: 9.15, 9.24
- Programming exercises
  - Programming problems: 7.15*, (7.17**,) 8.32*, 9.28*
    - Note: Each student must complete all programming problems on your own
  - Programming projects for Chap. 7*, Chap. 8*, and Chap. 9*
    - Team-based: select at least one programming project for Chap.7, and at least one from Chap.8 & 9.
- Due: two weeks (May 9, 2024)

- Chap 7:
  - 7.8: The Linux kernel has a policy that a process cannot hold a spinlock while attempting to acquire a semaphore. Explain why this policy is in place.

- Chap 8:
  - 8.20: 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.

- 8.27: Consider the following snapshot of a system :

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

(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=(2,2,2,3)

(b) Available=(4,4,1,1)

- 8.30: 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. 9:
  - 9.15: Compare the memory organization schemes of contiguous memory allocation and paging with respect to the following issues:
    - (a) external fragmentation
    - (b) internal fragmentation
    - (c) ability to share code across processes
    - .

- 9.24: Consider a computer system with a 32-bit logical address and 8-KB page size. The system supports up to 1 GB 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

Programming Problems

- 7.15*: In Exercise 4.27, you wrote a program to generate the Fibonacci sequence.
  - The program required the parent thread to wait for the child thread to finish its execution before printing out the computed values.
- If we let the parent thread access the Fibonacci numbers as soon as they were computed by the child thread—rather than waiting for the child thread to terminate—what changes would be necessary to the solution for this exercise? Implement your modified solution.

- (7.17**): Exercise 4.24 asked you to design a multithreaded program that estimated π using the Monte Carlo technique.
  - In that exercise, you were asked to create a single thread that generated random points, storing the result in a global variable.
  - Once that thread exited, the parent thread performed the calculation that estimated the value of π.
- Modify that program so that you create several threads, each of which generates random points and determines if the points fall within the circle.
- Each thread will have to update the global count of all points that fall within the circle.
- Protect against race conditions on updates to the shared global variable by using mutex locks.

- 8.32*: Implement your solution to Exercise 8.30 using POSIX synchronization.
  - 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.

- 9.28*: Assume that a system has a 32-bit virtual address with a 4-KB page size. Write a C 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:

  ./addresses 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.

End-of-Chapter Programming Projects

- Programming Projects for Chap. 7: (Choose one)
  - Project 1: Designing a Thread Pool
  - Project 2: The Sleeping Teaching Assistant
    - Room: 1 desk with a chair and computer
    - Hallway: 3 chairs
    - POSIX threads, mutex locks, and semaphores
  - Project 3: The Dining Philosophers Problem
    - Pthread mutex locks and condition variables
  - Project 4: The Producer-Consumer Problem
    - Use standard counting semaphores for empty and full and a mutex lock, rather than a binary semaphore, to represent mutex.
    - Producer and consumer threads
    - Pthreads mutex locks/semaphores
    - Windows mutex locks/semaphores

Programming projects

- Programming Project for Ch.8*: Banker's Algorithm
  - Write a multithreaded program that implements the *banker's algorithm* discussed in Section 8.6.3. 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.

Programming projects

- Programming Project for Ch.9*: Contiguous Memory Allocation
  - In Section 9.2, we presented different algorithms for contiguous memory allocation.
  - This project will involve managing a contiguous region of memory of size *MAX* where addresses may range from 0 ... *MAX*− 1.
  - Your program must respond to four different requests:
    - **1.** Request for a contiguous block of memory
    - **2.** Release of a contiguous block of memory
    - **3.** Compact unused holes of memory into one single block
    - **4.** Report the regions of free and allocated memory
  - Your program will be passed the initial amount of memory at startup

Any Questions or Comments?