OS Take-Home Exam #2, Fall 2017

This is an exam. You should not discuss it in any way with anyone but me. You should not consult the internet. You may use your book, notes, past projects/HW, a calculator, and any other materials from this class on Moodle. You may ask me any clarification questions, if in doubt, ask!

Due: Monday by 5pm (by 2:30pm if on paper)

Deliver: You may type your answers to these questions, print the exam and write on paper, or use a combination of both. However you must turn the completed exam in all in one format. You can either turn it in on paper (stapled together please) which you can deliver to my mailbox or put under my office door before 2:30pm on Monday. Alternatively you can submit it on Moodle up to 5pm on Monday, all in one document please (note you can take a picture of each handwritten page and insert the picture into a word document); just make sure it's legible!!!

Scheduling

1. It can be proven that SRTF (shortest remaining time first) scheduling yields an optimal average job-completion time given a preemptive CPU, so why don't all real OS's use it?
2. Two methods we discussed to approximate SRTF scheduling are multi-level feedback queues and a lottery scheduler. Describe 2 scenarios, one where you would expect better performance from MLFQ of the two schedulers, and the other where you expect lottery to perform better. Make sure it's clear which is which and why you expect that scheduler to perform well in that scenario.
3. Briefly describe a situation in which the following statements would be true, or if a statement can never be true explain why. FIFO refers to first-in first-out or first-come first-served, and RR refers to round-robin. Assume that **there are at least 3 jobs waiting** to be scheduled
    a. FIFO scheduling would lead to a lower *average completion time* than RR
    b. FIFO scheduling would lead to a lower *average start time* than RR
    c. RR scheduling would lead to a lower *average completion time* than FIFO
    d. RR scheduling would lead to a lower *average start time* than FIFO
    e. RR scheduling would lead to a higher *throughput* than FIFO after all waiting jobs have completed

Deadlock

1. Consider a system with four processes P1, P2, P3, and P4, and two resources, R1, and R2, respectively.
   Each resource has two instances. Furthermore:
   - P1 has already acquired an instance of R2, and requests an instance of R1;
   - P2 has already acquired an instance of R1, and doesn't need any other resource;
   - P3 has already acquired an instance of R1 and requires an instance of R2;
   - P4 has already acquired an instance of R2, and doesn't need any other resource.
    a. Draw the resource allocation graph
    b. Is there a cycle in the graph? If yes list the processes involved.
    c. Is the system in deadlock? If yes, explain why. If not, give a possible sequence of executions after which every process completes.
2. If the banker's algorithm finds that it's safe to allocate a resource to an existing thread, does it follow that all threads will eventually complete? Why or why not?
3. Explain one method to prevent deadlock in a system without using any detection algorithm.

Memory

1. For each of the following memory management mechanisms, describe all of the steps (related to memory) that should be completed when a process terminates.
   a. Simple translation with dynamic relocation – processes are stored in contiguous memory, translation occurs on execution of each instruction that accesses memory
   b. page-table for each process – maps process VPN to PFN (i.e. VPN is index into table and PFN is physical frame number of where this page's data is stored in RAM.
   c. single reverse look-up table for all of RAM – maps PFN to VPN for one specific process (i.e. each entry contains both process id and virtual page number)

2. Consider an initially empty **3-slot** (holds 3 entries) TLB for caching pages, and the below sequence of page references:

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

   No matter what cache-size and eviction-algorithm you use, the above page reference sequence will cause at least 8 compulsory TLB misses (first load of each page 0-7).
   a. For each eviction algorithm below, give the number of **non-compulsory** TLB misses. Show your work for partial credit.
      i. True LRU (least-recently-used)
      ii. Clock algorithm – initial pointer to index 0 in TLB
   b. How many slots (entries) would be required in the TLB to avoid **all non-compulsory** TLB misses given an optimal eviction algorithm (page that won't be needed for the longest time in the future is replaced)?
   c. How many non-compulsory TLB misses would true LRU cause given the number of slots you stated in part (b)?

File Systems

1. Answer the following questions for a 1GB hard drive with a 2KB block size using 16 bit logical addresses.
   a. What is the most blocks that a single file could use? Explain how you got this answer.
   b. If we were to use a basic iNode block allocation method, with only direct entries in the iNode table, and each entry used 32 bits, would the largest possible iNode table fit in one disk block? Show your work or explain your answer.
   c. If we were to use a FAT table as our file block allocation method, how big would the FAT table itself need to be? Specifically state how many entries it would have and how big each entry would be with a brief explanation of why.

2. For each of the following file allocation methods, state the steps that would be required to increase the length of a file. Assume that you are writing and go off the end of the last block, so you already know the location of the current last block, and now need to add one more block to the end of the file. If the answer is "it depends", explain that too!
   a. contiguous allocation (file is stored in consecutive blocks on disk)
   b. linked allocation (each block stores number of next block)
   c. FAT (single table of block numbers for all files on system, already loaded into RAM)
   d. iNode (table in first block of file holds some number of direct entries for file block numbers, then 1 or more multi-level iNode tables for the rest of the file blocks)

Distributed Lock Acquisition

The following questions refer to the 2 methods we discussed in class to achieve mutual exclusion in a distributed network of computers, which are

1) token passing
2) fully distributed message passing using timestamps and acknowledgements

Briefly explain your answers to the following questions:

1. Given a network with **N nodes**, if **exactly one** computer in the network attempts to acquire a lock, how many total **lock-related** messages will be passed **among all nodes** before the requesting computer can safely proceed with its critical section using each of the 2 methods? If the answer is "it depends", state **the least and most** messages that may be sent before the lock can be safely assumed to be acquired.
    a. token passing
    b. fully distributed messaging

2. Given a network with **N nodes**, if **every** computer in the network attempts to acquire a lock at the same time, how many total **lock-related** messages will be passed **among all nodes** before any one computer can safely proceed with its critical section using each of the 2 methods? If the answer is "it depends", state the least and most messages that may be sent before the lock can be safely assumed to be acquired.
    a. token passing
    b. fully distributed messaging

Distributed Chat Program

See separate assignment on Moodle for this!