Last Name:_____, First Initial:____

COP 4600 Operating Systems

Exam 1 r2

21 February 2017

Instructions

1. Read all instructions. Failure to follow instructions will result in loss of points.
2. This is a closed-book examination.
3. You are permitted one 8.5 by 11 inch sheet of notes (both sides OK) that you have prepared.
4. You are permitted **50 minutes** to complete this examination.
5. **Do not start** the exam until the proctor has told you to start.
6. **Answer any two (2) questions, and no more**. All questions are of equal value.
7. **Leave sufficient room in the upper left-hand corner for the staple** and staple your answer sheets in the room you have left.
8. Put the **question number in the top center** of each answer page and label each part of the question answer.
9. Include your last name and page number in the upper right hand corner of each answer page.
10. Show your work.
11. Start the answer to each question on a **new page** (i.e., do **not** put the answer to more than one question on the same page).
12. Use exactly one page of paper (both sides is OK, or two pages front side only) to hold the answer to each question, and please write legibly.
13. Assemble your answers in **numerical order** of the questions when you submit them.
14. Print your family name and first initial in the upper right hand corner of this page, and complete the honor statement affirmation below.

Read and sign the following statement. This page **MUST** be attached to your examination answers and **MUST** be completed to obtain credit for this examination.

On my honor, I have neither given nor received unauthorized aid on this examination.

Signed _____

Printed Name: _____

UFID: _____-_____

1. Architectures
   a. Compare virtual machines to user-level thread packages. Specifically, compare how processes in VMs and threads in a ULTP are managed.
   b. Continuing part (a), compare how memory management is handled.
   c. Continuing part (a), compare system calls.

2. Processes and Scheduling
   a. Give a practical way for a system using demand paging to determine whether there are processes that have too little RAM allocated to them. Include how the particular processes with insufficient allocation are identified, and justify that the methods you give are practical.
   b. Assuming that the system from (a) has a memory scheduler, what factors should be considered when selecting a process to swap out, and how these factors should be used to select the process(es) to swap out.
   c. How can such a system determine when a swapped-out process can safely be swapped back in? What factors should be taken into account in selecting a process to swap in, and how should they be used?

3. Memory Management
   a. Explain how kernel level threads in the same process are able to have shared memory. Be specific in terms of process layout and memory map.
   b. Give a possible approach for a user-level threads package to support memory needed for multiple threads without OS support. Include what parts of memory are unique per thread within a process and what parts are shared by all threads within a process.

4. Synchronization
   Consider the following generalization of semaphores, PVblock. Like semaphores, after creation/initialization, PVblock semaphores have only two operations, Up and Down. However, both primitives take two parameters: the semaphore name (like regular semaphores) and a positive integer value (count). Down(S,count) is like Down(S) on a regular semaphore S, except that S must have a stored value no less than count or else the caller is blocked.
   a. Sketch the code necessary to implement Up(S, count) and Down(S, count). You may assume basic data structures such a queues.
   b. Under what circumstances can a process remain blocked even when the value associated with a semaphore is positive? Explain.
   c. Explain the policy issues that can arise from performing Up() on S when there are multiple processes blocked on S. Describe two distinct policies and explain how they can be implemented, and what the consequences of each may be.