Introduction to Operating Systems
Midterm Examination

Name:____________________________ E-mail:____________________________

Winter 2014

Part I: Multiple Choice – Mark your answer (2 points each)

1. Multiprogramming is:
   (a) Multiple processes running at different times.
   (b) Multiple processes residing in memory and running more or less at the same time.
   (c) Multiple processes running at exactly the same time.

2. Which has the fastest access times:
   (a) Cache.
   (b) Registers.
   (c) Main memory.

3. Which are all system calls:
   (a) `chmod()`, `chdir()`, `read()`, and `main()`
   (b) `waitpid()`, `lsdir()`, `mount()`, and `open()`
   (c) `fork()`, `exit()`, `lseek()`, and `kill()`

4. Which is true about processes and threads:
   (a) Threads in a process share the same stack.
   (b) Threads in a process share the same file descriptors.
   (c) Threads in a process share the same register values.

5. A race condition is when:
   (a) Multiple processes are all trying to finish first.
   (b) A process runs too slow.
   (c) The correctness of the code depends upon the timing of the execution.

6. Mutual exclusion is when:
   (a) A set of processes are prevented from running because higher priority processes keep getting in ahead of them.
   (b) A set of processes is waiting for an event that only another process in the set can cause.
   (c) A set of processes are prevented from simultaneously accessing a shared data structure.

7. *Semaphores, locks* and *condition variables,* and *monitors* support two distinct operations:
   (a) Parallelization and event notification.
   (b) Synchronization and event notification.
   (c) Synchronization and parallelization.

8. Which is true:
   (a) Round-robin has the highest turn-around time.
   (b) Shortest remaining time next is the fairest scheduling algorithm.
   (c) Priority-based scheduling is the most general.

9. The difference between preemptive and non-preemptive scheduling is:

   (a) Whether or not a ready process can be involuntarily terminated from the ready state.

   (b) Whether or not a running process is involuntarily removed from the running state.

   (c) Whether or not a blocked process can have its resources involuntarily taken away from it.

10. The four conditions that must hold in order for deadlock to occur are:

    (a) Mutual exclusion, hold-and-wait, circular wait, and no preemption.

    (b) No preemption, no starvation, circular wait, and mutual exclusion.

    (c) Hold-and-wait, circular wait, no starvation and mutual exclusion.

11. Four strategies for dealing with deadlock are:

    (a) Conservative resource allocation through the Banker's algorithm, spooling, two-phase locking, and mutual exclusion.

    (b) Do nothing, kill one of the processes, preemptible resources, and spooling.

    (c) Two-phase locking, resource ordering, resource discovery, and do nothing.

12. Which is false:

    (a) Memory compaction is time consuming.

    (b) Degree of multiprogramming refers to the amount of time a process spends waiting for I/0.

    (c) Swapping can result in external fragmentation.

13. The main advantage of paging is:

    (a) That it doesn't require any extra hardware support.

    (b) The efficiency with which data can be brought into memory.

    (c) That only the current working set of the process need actually be in memory at any one time.

14. The main advantage of multilevel page tables is that they:

    (a) Use page table memory efficiently.

    (b) Reduce the complexity of the paging system.

    (c) Speed up page table references.

15. Which are both Stack algorithms:

    (a) Optimal and LRU.

    (b) LRU and FIFO.

    (c) FIFO and Optimal.

Part II: Short answer – Write a paragraph or two (no more) (5 points each)

1. Explain the difference between processes and threads. What are the advantages of using threads (rather than processes) in implementing a complex application?

2. List two different ways a higher-priority process could be given a larger fraction of the CPU time. Assume processes get to run for a fixed-length quantum: the only freedom you have is in choosing which process to run for the next quantum.

3. The five (basic) states that a process can be in are: *new, ready, blocked, running* and *terminated.* Describe the transitions among these states, and state when they occur.

   Hint: The best way to do this is to draw a diagram.

4. Recall that a *monitor* is a method for controlling access to a critical section (like a semaphore, but a higher level abstraction). Briefly describe the monitor abstraction, and do not forget to consider the case when a process may have to wait *inside* the monitor (condition variables).

   Hint: You may find it helpful to draw a picture.

5. Explain the differences between a *safe* state, an *unsafe* state, and a *deadlocked* state. For each of the following transitions, briefly explain what causes a system to make that transition, or if such a transition is impossible, why is impossible: safe → unsafe; unsafe → deadlocked; safe → deadlocked.

6. Using the reference string ⟨2 1 0 3 2 1 4 2 1 0 3 4⟩, fill in the two tables below (representing systems with *three* and *four* page frames, respectively) using the FIFO page replacement policy. Indicate when a page fault occurs, and note how many page faults occur. Do you notice any anomalies?

7. Using the same reference string as in the previous question, fill in the following two tables as before using the LRU page replacement policy. Indicate when a page fault occurs, and note how many page faults occur.

8. The LRU page replacement algorithm has the property that $\mathcal{M}(r, m) \subseteq \mathcal{M}(r, m+1)$. What is "Belady's anomaly" and why does an algorithm with this property preclude it from happening?

9. The *shortest job first* (SJF) scheduling CPU algorithm has a definite benefit, but also has a serious problem. The benefit is that it is provably optimal. In what way is it considered optimal? What is the serious problem?

10. Consider the *working set* model of virtual memory. In a multiprocess environment, what should be done if the working set of the resident processes cannot be kept in main memory?