UNIVERSITY OF WISCONSIN-MADISON
Computer Sciences Department

Operating Systems **Spring 2009**

Instructions: There are *six* questions on this exam; you must answer *all* of the following questions.

Question 1: Think about Synch-ronization

This question is about synchronization in multi-threaded programs.

1. Composability is always a problem for monitor-based synchronization systems (e.g. Mesa). Suppose procedure P1 of monitor M1 calls procedure P2 of monitor M2 and P2 executes a "wait"; what would the monitor system do at this point? What are the potential problems?
2. Deadlock is a potentially nasty problem one encounters in multi-threaded programs. Please list the necessary conditions for a deadlock to occur.
3. Can the following approaches guarantee that deadlock will be avoided or recovered from? Please explain, using examples to support your answers.
    A. Never do context switch inside a critical section
    B. Whenever a deadlock occurs, roll back the program to an earlier checkpoint outside any critical section and re-execute the program

Question 2: NFS - Not-another File System question!

This question is about NFS version 2, the classic stateless distributed file systems protocol from Sun.

1. An application on a client issues a write to a file. That write is buffered in client memory for some time. Why do NFS clients do this? What are the advantages and disadvantages?
2. The client at some point decides to issue the write to the server. When does the client usually do this? Why?
3. The write reaches the server, and the server writes it to disk immediately, before acknowledging the request. Why does the server do this? What if the server ack'd the request and then wrote the data to disk?
4. The ack from the server gets lost, and the client resends the write protocol request. What happens in this case? (and why does it work?) What could the server do to improve performance in this situation?

Question 3: Kernel.org(anization)

Many important operating system kernels were designed in the context of a uniprocessor or a system with a small number of processors.

1. List and describe **three** impacts that the number of processors have on operating system kernel design.

Now that you have done part 1, let's get more specific. For parts 2 and 3 of this question, answer the questions about **any two** of these important systems:

- THE
- Mach
- exokernel

2. Does the existence of a large number of processor cores improve the design, providing additional benefits beyond those on a single processor? If so, under what circumstances?
3. Do additional processors expose flaws in the design? If so, under what circumstances.

Question 4: Game, Working Set, Match

The working set model of memory determines how much memory a process requires to make steady progress. The original working set mechanism was described for fast memory and a slow hard disk.

1. Explain what the working set of a program is.
2. When using the working set concept to manage memory, the paging system and the scheduler must cooperate. Explain how the two subsystems cooperate to prevent thrashing.
3. Explain how the working set of a program changes when the hardware you are using changes.
    1. What if you replace the hard disk with flash memory that is 10 times faster?
    2. What if your processor gets 10 times faster?

Question 5: You Heard it through the ... er ... Grapevine

This question is about Grapevine, a classic distributed system from Xerox PARC.

1. Grapevine strives to be a highly available and reliable message delivery service. Describe the guarantees Grapevine provides to clients regarding availability and reliability.
2. Grapevine uses replication to achieve high availability for the message delivery service. What does Grapevine replicate? Why? What does Grapevine not replicate? Why not? What are the advantages and disadvantages of this approach?
3. Grapevine specializes its definition of consistency for this particular service. In what way(s) does Grapevine relax its consistency requirements? How does this definition impact the user? Why (or why not) are these consistency semantics appropriate given their goals?

Question 6: Feeling Insecure

Consider a virus that infects a program through its input. The virus copies some of the user's files up to a central server and propagates itself by generating output that is later read by another instance of the program on another system.

1. Explain how the user's security is violated: what illegal actions is the virus executing?
2. Are the security facilities of Unix sufficient to address this problem? If so, how? If not, what is missing?
3. Do the broad facilities of Hydra help solve this problem? If so, how should they be used? If not, why not?