Name:

Operating Systems (COP 6611)
Spring 2010
Final Exam

Closed books, notes, cell phones, PDAs, iPods, laptops, etc. No headphones, please. You do not need a calculator.

This exam contains 30 numbered statements. Indicate whether each statement is true or false. If you wish, you may elaborate on each answer with one sentence. Each correct answer is worth 3 points. You receive an end-of-semester gift of 10 points just by signing your name on your exam paper. Total points: 100.

Synchronization. Define a race as follows: a race exists if and only if two conflicting accesses to some shared variable occur concurrently (i.e., neither happened-before the other). The following statements pertain to multithreaded programs using mutexes to eliminate races.

1. The implementation of mutexes guarantees that happened-before defines a total ordering on the synchronization (acquire and release) operations for each mutex.

2. If two executions of a race-free program on the same input perform the same sequence of acquire and release events in the same order, then the executions perform the same sequence of accesses to shared variables, and both executions produce the same final result.

3. If a program uses down() and up() primitives correctly, then mutex synchronization induces a total ordering on the down() and up() events.

Memory, etc.

4. A page fault handler executing in kernel mode must not block.

5. For systems with preemptive scheduling, increasing the scheduling quantum is a good way to improve system throughput.

6. A process requires as many page tables as the number of threads it contains.

7. Virtual memory page table formats impose a limit on the amount of physical memory configured in a system.

8. The space overhead for multi-level virtual memory page tables depends on the page size and the virtual address space size, but it is independent of the configured size of physical memory.

9. The overhead of the Unix fork primitive grows linearly with the size of the active address space, and virtual memory increases this cost significantly.

10. A uniprocessor OS may need to flush entries from its TLB when evicting a page, but not when mapping a new page to satisfy a page fault.

11. An OS may need to flush entries from its TLB on a context switch, but not when a process exits.

Storage and file systems

12. The peak read bandwidth of a RAID-4 array grows linearly with the number of disks, but the single parity disk limits the peak write bandwidth.

13. Doubling the rotation speed of a 10K RPM (rotations per minute) disk doubles its peak transfer bandwidth, but improves its throughput for random 8KB reads only marginally.

14. Doubling the bit density of a 10K RPM disk doubles its peak transfer bandwidth, but improves its throughput for random 8KB reads only marginally.

15. You have a system with two primary workloads. One performs sequential access to a set of large files, the other performs small, independent, random reads to a separate set of large files. To satisfy these two workloads, you have a set of 4 disks. Both applications can saturate the peak bandwidth of your disk subsystem (i.e., they are I/O bound). Each disk can provide 100MB/s and has an average seek time of 10ms. The best way to organize data on these disks in order to provide the best performance for the random workload is to mirror the disks (that is, store 4 identical replicas on the 4 disks).

16. A log-based file system can deliver data at close to the peak transfer bandwidth of its disk system, independent of which files or directories are being accessed.

17. A log-based file system can perform writes and updates at close to the peak transfer bandwidth of its disk system, independent of which files or directories are being updated.

18. To preserve data integrity, file system software must control the order in which disk writes complete.

19. Google file system is a general purpose file system likely to be adopted on personal computers in the future.

20. The master node in Google file system provides a stateful service.

21. Swap space allocated as a special region (partition) on the disk can be faster than if allocated as a special file because it avoids file system costs due to disk organization.

22. A stateful file service recovers faster after a crash because all state information is stored locally, while a stateless file service needs to bring state information from a distant node.

Distributed coordination

23. The centralized approach to deadlock detection in a distributed system requires a larger number of messages than a decentralized approach (which is the reason why decentralized solutions are always preferred).

24. An election algorithm for a bidirectional ring can be more efficient than the one presented in the textbook for the unidirectional ring.

25. A deadlock detection scheme is always more efficient than a deadlock prevention scheme in distributed systems.

26. Events in a distributed system can be globally ordered in time using the happened-before relationship presented in Lamport's paper.

27. In a distributed system deployed over the Internet, agreement among n processes can be reached always in constant x (n+1) rounds of communication.

Protection and Security

28. The difference between a worm and a virus is in the type of attack they perform.

29. Revoking access rights is easier in protection with access lists than with capabilities.

30. Protection domains are implemented in Unix via users and domain switching is implemented via the file system.