Stanford University Computer Science Department

Fall 2001 Comprehensive Exam in Software Systems

SOLUTIONS

1. **CLOSED BOOK**: no notes, textbook, computer, Internet access, etc.
2. **WRITE ONLY IN BLUE BOOKS: No credit** for answers written on these exam pages.
3. **WRITE MAGIC NUMBER** on the cover of **EACH** blue book.
4. The exam is designed to take less than an hour.
5. If you need to make assumptions to answer a question, *state them clearly*.
6. **For questions that refer generically to "the OS": if you think the answer would differ substantially depending on the OS in question, assume any popular flavor of Unix (Linux, BSD, Solaris, etc.).**

Short Answers (3 pts each, 30 pts total)

1) Explain how dynamically linked libraries (DLL's or *.so* libraries) facilitate in-place OS or application software upgrades. (Hint: you can do this by contrasting them with statically-linked libraries.)
One can replace (upgrade) a DLL without replacing or modifying the applications that use it. With static libraries, in order to take advantage of a new version of the library, each app that used it would have to be recompiled.

2) Why are DLL's now back in favor, after being out of favor during the early days of the PC revolution?
Libraries have gotten large and machines can run multiple apps at once; having multiple copies of a static library rather than sharing one copy among multiple apps quickly eats up memory.

3) When a context switch occurs and a user process must give up the CPU, what *hardware* state information is saved by the OS in order to be able to later resume that user process?
Program counter, stack pointer, integer register file, FP register file (if present), processor flags (arithmetic and possibly privilege level).

4) This information is not sufficient to capture the state of the whole process from the OS's point of view. Name two other types of state that must be captured, and specify where they are kept.

Various choices of things kept in the process control block, including: page table base register

for the process; open file descriptors or other I/O endpoints (eg sockets), ie references to kernel data structures about open files. Page tables themselves are also acceptable.

5) Name one advantage of user-level threads over kernel-level threads.

   No kernel crossing to do a thread switch, so thread switches within a process are lower overhead.

6) Name one advantage of kernel-level threads over user-level threads.

   Scheduling can be done at finer grain (all user-level threads for a process compete for the process's scheduled time and resources).

7) What is the difference between authentication and authorization?

   Authentication proves you are who you say you are. Authorization says you have the right to do something.

8) Asynchronous I/O devices can be handled by either polling or interrupts. Describe one advantage of using polling over interrupts.

   Interrupts incur high processing overhead (because of the kernel crossing) and require hardware support.

9) Describe one advantage of using interrupts over polling.

   Interrupts occur exactly when an interesting event happens; polling can miss important events (if it's not frequent enough) or waste time checking too frequently (if important events are rare).

10) Name one difference between paging and segmentation.

    Page sizes are fixed to one (or a very few) particular sizes; segments can usually be of essentially arbitrary length.

Slightly Longer Answers (5 pts each, 30 pts total)

11) You look up the documentation for a particular Unix library function that you want to use. According to the documentation, the library function's implementation is reentrant. What does this mean, and under what circumstances would you care about this fact?

    It means multiple loci of control (typically, multiple threads) can be in the function at the same time. You would care if the function is being called in a multithreaded program, because you need to know whether to enforce mutual exclusion for threads that might call the function.

12) To successfully prevent user programs from causing damage to other programs or the OS, hardware support is required. Name two hardware support mechanisms in modern CPU's that enable this.

13) For each mechanism, briefly describe how it is used by the OS and in what specific way(s) it protects the OS or other user programs (i.e. what specific kind(s) of damage are prevented).

Virtual memory/page tables: allows OS to give each process its own address space and protects them from each other. OS also runs in its own address space, protecting it from user programs.

Privilege levels: some operations can only be done in "high privilege" or supervisor mode. This allows OS the exclusive ability to manage I/O and other resources, rather than exposing them to abuse by user programs.

14) A RAID system can fail if two or more of its drives crash within a short time interval. Suppose each drive fails once an hour with probability p, and that drive failures are independent. What is the probability of a *k*-drive RAID system failing in a given hour?

Probability that zero drives fail during a given hour: $(1-p)^k$. Probability that exactly one drive fails: $kp(1-p)^{k-1}$. Therefore, probability that 2 or more fail: $1-(1-p)^k-kp(1-p)^{k-1}$.

15) You want to transmit a large text file securely to a friend over a slow network. You decide to use both compression and encryption on the file. Which should you apply first and why?

Compress first, because the redundancy in text will be eliminated when the text is encrypted, so that the compression step would have nothing to do.

16) Ethernet and other CSMA networks use *exponential backoff* when a transmit collision is detected. Briefly explain what this is, and concretely explain the intuition behind using exponential (as opposed to some other function) backoff.

The more senders there are, the more likely are collisions. If everyone starts with a random backoff, even if they are staggered, they're more likely to collide again if conditions are crowded. With exponential backoff, the relative staggering of the re-sends quickly gets very far apart, so the probability of *repeated* collisions does not grow linearly with the number of senders.