Sample Exam Questions

These are examples of a few different types of questions that have been asked on other mid-terms, along with examples of full-credit, 2/3 credit, and 1/3 credit answers to those questions. The purpose of these examples is to show you:

- What kinds of questions I am likely to ask
- What kinds of answers I am expecting to see

"Principles" question

Why is it important to keep interface definitions distinct from implementations?
Give an example of a bad thing that could happen if this principle were violated.

Full Credit Answer

If the interface is an implementation-agnostic specification of expected behavior, it will be much easier to change the implementation, while maintaining interface compatability. Otherwise, it may be difficult to know what the actual interface is.

If the interface definition was "the current behavior of module X":

a. individual clients are left to examine behavior and try to guess the interface specifications. This greatly increases the likelihood that future changes will break old clients.
b. implementors are likely to find it impossible to change the implementation without breaking some clients. This will greatly constrain their ability to evolve their software.

2/3 Credit Answer

Without a clear interface specification, nobody will be sure what the right interface is, and a much greater chance that future implementations will break some clients.

1/3 Credit Answer

Better interace definitions, and fewer fewer upwards compatability problems.

"Description" question

List the primary types of segments found in a process' virtual address space and two characteristics (not contents) of each.

Full Credit Answer

The key segments in a process' address space are:

- text ... executable, read-only
- data ... read/write size, can change at run-time
- stack ... read/write, grown/shrunk by os as needed
- DLLs/shared libs ... executable, read-only other code segments loaded at load- or run-time
- thread stacks ... like program stacks, but are probably allocated in the data segment and may not be managed by OS

2/3 Credit Answer

- text ... executable code
- data ... read/write program data
- stack ... read/write, grows down

1/3 Credit Answer

- code
- data
- stack

"Process" question

Describe or illustrate (in detail) the sequence of operatins involved in the processing of an illegal-address trap from a user-mode program with a registered segmentation fault handler.

Full Credit Answer

1. illegal address raises fault in CPU
2. CPU loads new PC/PS from associated trap vector

3. CPU pushes PC/PS at time of trap onto supervisor mode stack
4. first level handler saves registers and status, forwards to 2nd level handler
5. 2nd level handler determines it was a user-mode segmentation fault and that there is a registered signal handler
6. push PC/PS at time of trap onto user mode stack, and changes the return PC (on the supervisor mode stack) to be the signal handler.
7. 2nd level handler returns to 1st level handler
8. 1st level handler restores registers and returns to user mode
9. user mode execution resumes in the registered signal handler

2/3 Credit Answer

1. CPU loads new PC/PS from associated trap vector
2. CPU pushes PC/PS at time of trap onto supervisor mode stack
3. OS trap handler saves registers and identifies cause of trap
4. Trap handler finds signal handler and changes stack to return to it
5. Restore registers, and return (to user-mode signal handler)

1/3 Credit Answer

- trap into operating system
- OS identifies cause
- OS identifies handler
- OS invokes signal handler

"Comparison" question

(a) List two key similarities between a (trap) system call and a procedure call.
(b) List two key differences between a (trap) system call and a procedure call.

Full Credit Answer

a. System calls are like procedure calls in that they pass parameters, perform a service, and return a result when the operation is complete.
b. System calls run with different privileges, in a different address space. Procedure calls are made with direct call instructions, while system calls are a requested with

an illegal instruction that is recognized by the trap handler. Consquently, system calls are much more expensive than procedure calls.

2/3 Credit Answer

a. parameter passing and return values
b. subroutines are fast, system calls go into OS, which is slow

1/3 Credit Answer

a. both are made from subroutine calls in C
b. subroutines are in libraries, system calls are in OS