

Slide 1

Administrivia

- (None.)

Slide 2

System Calls

- Recall that some things can/should only be done by o/s (e.g., I/O), and hardware can help enforce that.
- But application programs need to be able to request these services. How can we make this work? System calls . . .

System Calls — Mechanism

Slide 3

- Library routine (running in user mode) sets up parameters and issues TRAP instruction or similar. A key parameter says which system call is being made (to create a process, open a file, etc.).
- TRAP instruction switches to kernel mode and transfers control to a fixed address.
- At that address is code for “handler” that uses parameters set up by library routine to figure out which system call is being invoked and call appropriate code.
- When processing of system call is finished, control returns to calling program — *if* appropriate. (What are other possibilities? Consider situations involving waiting, errors.) Return to calling program also switches back to user mode.

System Calls — Services Provided

Slide 4

- Typical services provided include creating processes, creating files and directories, etc., etc. — details depend on (and in some ways define, from application programmer's perspective) operating system.
 - Examples discussed in textbook:
 - POSIX (Portable Operating System Interface (for UNIX)) — about 100 calls.
 - Win32 API (Windows 32-bit Application Program Interface) — thousands of calls.
- Worth noting that the actual number of system calls is likely smaller — interface may contain function calls that are implemented completely in user space (no TRAP to kernel space).

Interrupts

Slide 5

- Processing of TRAP instructions is similar to interrupts, so worth mentioning here:
- Very useful to have a way to interrupt current processing when an unexpected or don't-know-when event happens — error occurs (e.g., invalid operation), I/O operation completes.
- On interrupt, goal is to save enough of current state to allow us to restart current activity later:
 - Save old value of program counter.
 - Disable interrupts.
 - Transfer control to fixed location (“interrupt handler” or “interrupt vector”) — normally o/s code that saves other registers, re-enables interrupts, decides what to do next, etc.

Minute Essay

Slide 6

- None — sign in.