System Calls

COMS W4118

**References:** Operating Systems Concepts (9e), Linux Kernel Development, previous W4118s

**Copyright notice:** care has been taken to use only those web images deemed by the instructor to be in the public domain. If you see a copyrighted image on any slide and are the copyright owner, please contact the instructor. It will be removed.
Address Space Overview

Kernel Address Space

Process A

Process B

Process C
System calls

• User process normally runs in unprivileged **user mode**
  – Cannot perform privileged operations
• User process issues **system call** to enter **kernel mode**
  – Privilege elevated, but only for predefined functions
Three kinds of interrupts

- Hardware interrupts
  - Ex) network packet, timer, key press, mouse click
- Exceptions
  - Ex) dividing by zero
- Software interrupts
  - Ex) int 0x80

```c
while (1) {
    if (interrupt or exception) {
        n = interrupt/exception type
        call interrupt handler n
    }
    fetch next instruction
    if (instruction == int n)
        call interrupt handler n
    else
        run instruction
}
```
Linux System Call Dispatch

To see code for a Linux syscall: http://syscalls.kernelgrok.com
• Syscalls with fewer than 6 parameters passed in registers
  – %eax (syscall number), %ebx, %ecx, %esi, %edi, %ebp
• If 6 or more arguments
  – Pass pointer to block structure containing argument list
• Maximum size of argument is register size
  – Larger arguments passed as pointers
• Use special routines to fetch pointer arguments
  – get_user(), put_user(), copy_to_user(), copy_from_user
  – Include/asm/uaccess.S
  – These functions can block. Why?
  – Why use these functions?
• OS must validate system call parameters
Tracing system calls in Linux

- Use the “strace” command (man strace for info)
- Linux has a powerful mechanism for tracing system call execution for a compiled application
  - Output is printed for each system call as it is executed, including parameters and return codes
- `ptrace()` system call is used to implement `strace`
  - Also used by debuggers (breakpoint, singlestep, etc)
- Use the “ltrace” command to trace dynamically loaded library calls
System Call Tracing Demo

- **pwd**

- **ltrace pwd**
  - Library calls
  - `setlocale`, `getcwd`, `puts`: makes sense

- **strace pwd**
  - System calls
  - `execve`, `open`, `fstat`, `mmap`, `brk`: what are these?
  - `getcwd`, `write`
Interesting System Calls

• brk, sbrk: increase size of program data
  – void* sbrk(int bytes)
  – Accessed through malloc
• mmap
  – Another way to allocate memory
  – Maps a file into a process’s address space
  – Or just grab memory with MAP_ANONYMOUS
  – MAP_PRIVATE or MAP_SHARED