

# Announcements

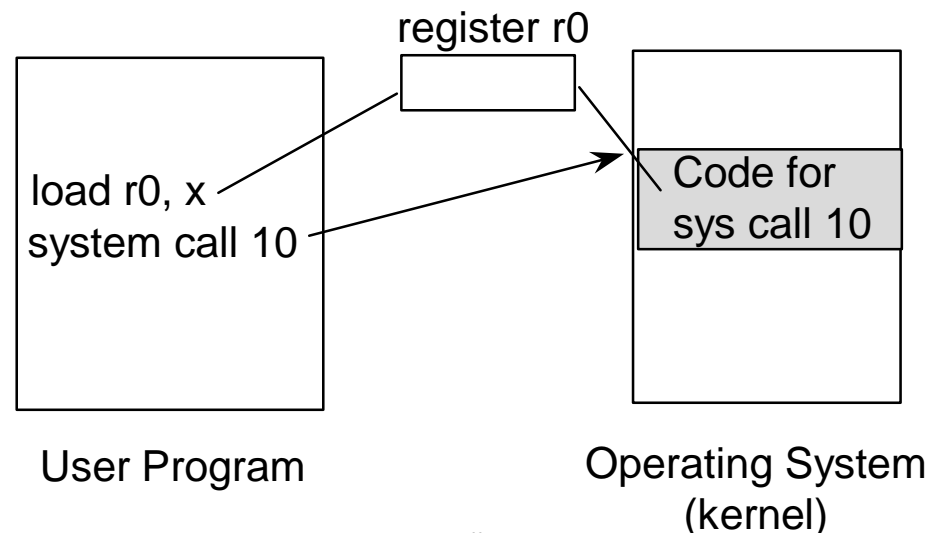
- Program #0 is due on Wed in section
- Program #1 is due next Wed
- Reading chapter 4 (4.1-4.4)

# Hardware Protection

- Need to protect programs from each other
- Processor has modes
  - user mode and supervisor (monitor, privileged)
  - operations permitted in user mode are a subset of supervisor mode
- Memory Protection
  - control access to memory
  - only part of the memory is available
    - can be done with base/bound registers
- I/O Protection
  - I/O devices can only be accessed in supervisor mode
- Processor Protection
  - Periodic timer returns processor to supervisor mode

# System Calls

- Provide the interface between application programs and the kernel
- Are like procedure calls
  - take parameters
  - calling routine waits for response
- Permit application programs to access protected resources



# System Call Mechanism

- Use numbers to indicate what call is made
- Parameters are passed in registers or on the stack
- Why do we use indirection of system call numbers rather than directly calling a kernel subroutine?
  - provides protection since the only routines available are those that are export
  - permits changing the size and location of system call implementations without having to re-link application programs

# Types of System Calls

- File Related

- open, create
- read, write
- close, delete
- get or set file attributes

- Information

- get time
- set system data (OS parameters)
- get process information (id, time used)

- Communication

- establish a connection
- send, receive messages
- terminate a connection

- Process control

- create/terminate a process (including self)

# System Structure

- **Simple Structure (or no structure)**
  - any part of the system may use the functionality of the rest of the system
  - MS-DOS (user programs can call low level I/O routines)
- **Layered Structure**
  - layer n can only see the functionality that layer n-1 exports
  - provides good abstraction from the lower level details
    - new hardware can be added if it provides the interface required of a particular layer
  - system call interface is an example of layering
  - can be slow if there are too many layers
- **Hybrid Approach**
  - most real systems fall somewhere in the middle

# Policy vs. Mechanism

- Policy - what to do
  - users should not be able to read other users files
- Mechanism- how to accomplish the goal
  - file protection properties are checked on open system call
- Want to be able to change policy without having to change mechanism
  - change default file protection
- Extreme examples of each:
  - micro-kernel OS - all mechanism, no policy
  - MACOS - policy and mechanism are bound together

# Processes

- What is a process?

- a program in execution
- “An execution stream in the context of a particular state”
- a piece of code along with all the things the code can affect or be affected by.
  - this is a bit too general. It includes all files and transitively all other processes
- only one thing happens at a time within a process

- What's not a process?

- program on a disk - a process is an active object, but a program is just a file



# Multi-programming

- Systems that permit more than one process at once
  - virtually all computers today
- Permits more efficient use of resources
  - while one process is waiting another can run
- Provides natural abstraction of different activities
  - windowing system
  - editor
  - mail daemon
- Preemptive vs. non-preemptive multi-programming
  - preemptive means that a process can be forced off the processor by the OS
  - provides processor protection