CS 537
Lecture 3
OS Structure

Michael Swift

Review from last time

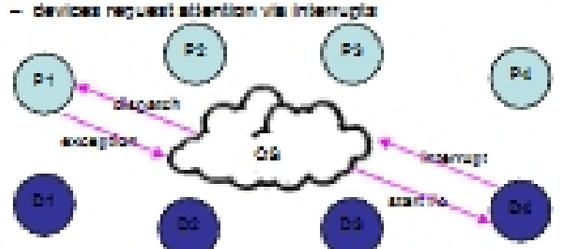
- What HW structures are used by the OS?
- What is a system call?

What you should learn from this lecture

- What are the major components of an operating system?
- How are operating systems structured and why?

OS structure

- The OS sits between application programs and the hardware
  - it mediates access and abstracts away ugliness
  - programs request services via exceptions (traps or faults)
  - devices request attention via interrupts



Major OS components

- processes
- memory
- I/O
- secondary storage
- file systems
- protection
- accounting
- shells (command interpreter, or OS UI)
- GUI
- networking

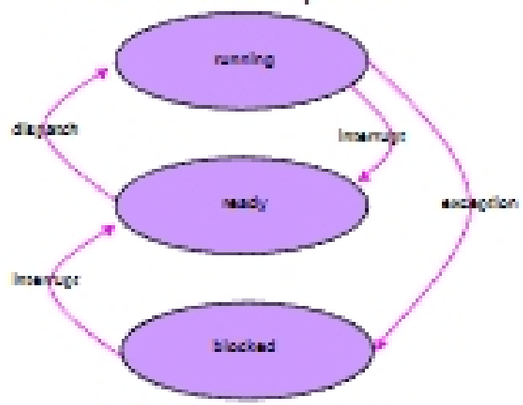
Process management

- An OS executes many kinds of activities:
  - users' programs
  - batch jobs or scripts
  - system programs
    - print spoolers, name servers, file servers, network daemons, ...
- Each of these activities is encapsulated in a process
  - a process includes the execution context
    - PC, registers, VM, OS resources (e.g., open files), etc...
    - plus the program itself (code and data)
  - the OS's process module manages these processes
    - creation, destruction, scheduling, ...

Process / processor / procedure

- Note that a program is totally passive
  - just bytes on a disk that contain instructions to be run
- A process is an instance of a program being executed by a (real or virtual) processor
  - at any instant, there may be many processes running copies of the same program (e.g., an editor); each process is separate and (usually) independent
  - Linux: `ps -ax` to list all processes

States of a user process



Process operations

- The OS provides the following kinds of operations on processes (i.e. the process abstraction interface):
  - create a process
  - delete a process
  - suspend a process
  - resume a process
  - clone a process
  - inter-process communication
  - inter-process synchronization
  - create/delete a child process (subprocess)

Memory management

- The primary memory (or RAM) is the directly accessed storage for the CPU
  - programs must be stored in memory to execute
  - memory access is fast (e.g., 60 ns to load/store)
    - but memory doesn't survive power failures
- OS must:
  - allocate memory space for programs (explicitly and implicitly)
  - deallocate space when needed by rest of system
  - maintain mappings from physical to virtual memory
    - through page tables
  - decide how much memory to allocate to each process
    - a policy decision
  - decide when to remove a process from memory
    - also policy

I/O

- A big chunk of the OS kernel deals with I/O
  - Millions of lines in Windows XP (including drivers)
  - 70% of Linux code
- The OS provides a standard interface between programs (user or system) and devices
  - file system (disk), sockets (network), frame buffer (video)
- Device drivers are the routines that interact with specific device types
  - encapsulates device-specific knowledge
  - e.g., how to initialize a device, how to request I/O, how to handle interrupts or errors
  - examples: SCSI device drivers, Ethernet card drivers, video card drivers, sound card drivers, ...
- Note: Windows has ~35,000 device drivers!

Secondary storage

- Secondary storage (disk, tape) is persistent memory
  - often magnetic media, survives power failures (hopefully)
- Routines that interact with disks are typically at a very low level in the OS
  - used by many components (file system, VM, ...)
  - handle scheduling of disk operations, head movement, error handling, and often management of space on disks
- Usually independent of file system
  - although there may be cooperation
  - file system knowledge of device details can help optimize performance
    - e.g., place related files close together on disk