CSc 360
Operating Systems
Course Overview

Jianping Pan

Summer 2015

Assignment 0

- Due on Friday, May 8th, 2015
- Send an email to pan@uvic.ca
  - From *you*@uvic.ca (or *you*@csc.uvic.ca)
  - Subject: [csc360] A0
    - name, student number, academic program
    - things you already known in OS
    - things you want to know in OS
    - issues with logistics, course rep, USRA/JCURA?
    - a URL to your mug shot
      - let me know you! e.g., reference letters

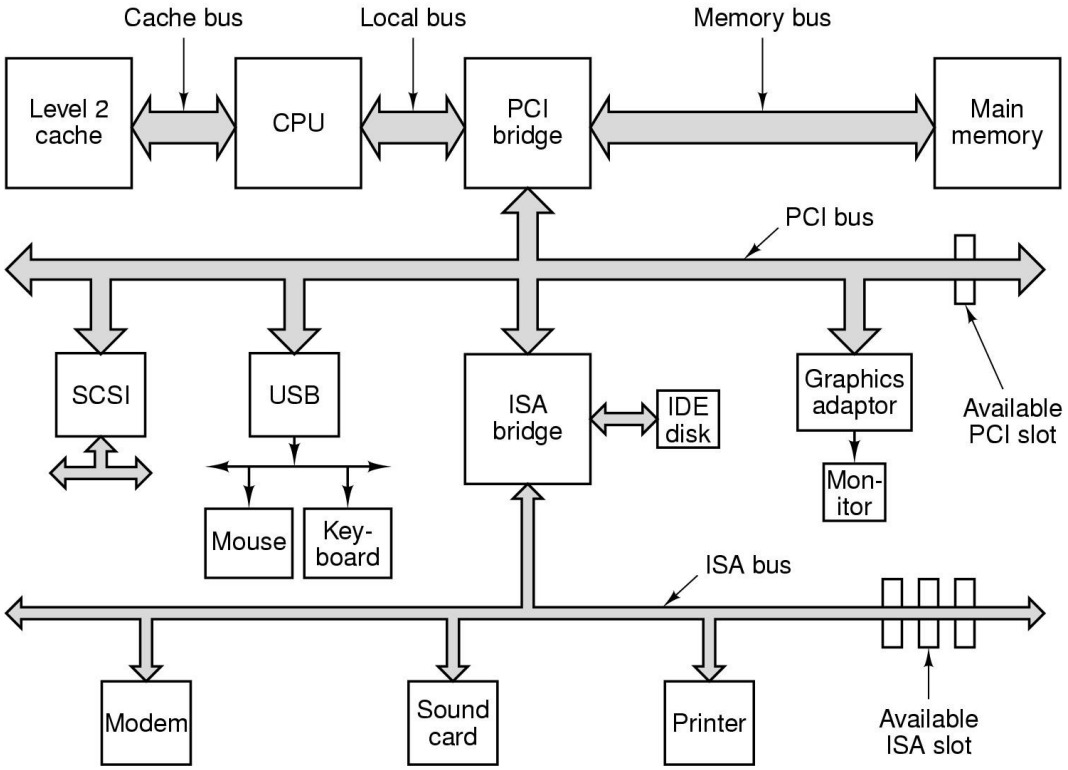
* connex->csc360->lectures or http://www.cs.uvic.ca/~pan/csc360

A quick review and a quick overview

- Computer organization and architecture
  - CPU
  - Memory
  - I/O: polling, interrupt, DMA
- Operating system
  - The software that manages CPU, memory and I/O, as many more
  - so let's review CPU, memory and I/O first

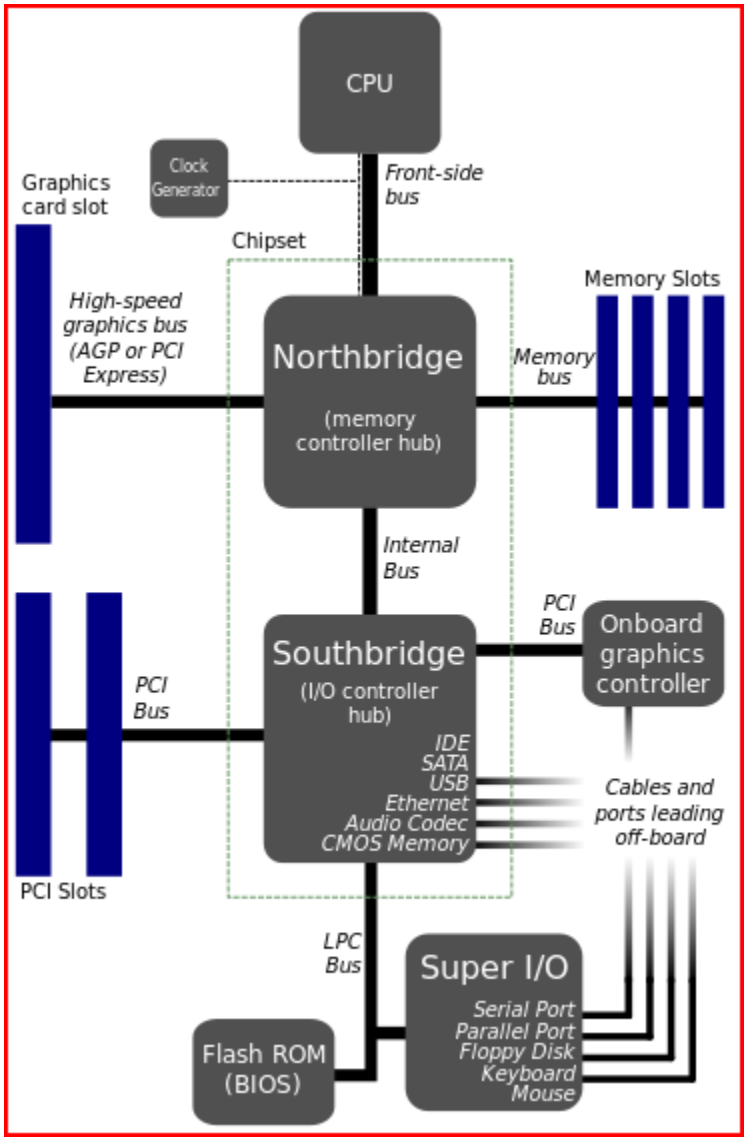
Computer Organization

- CPU
- Memory
- I/O

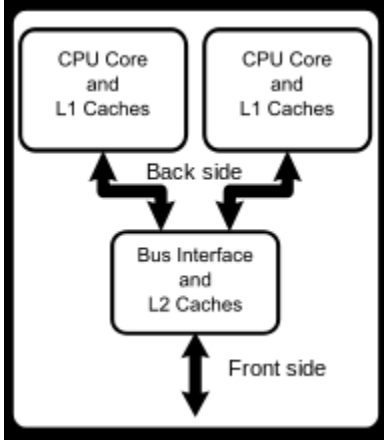


then

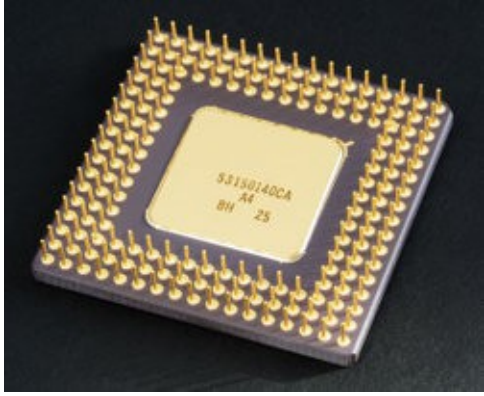
now

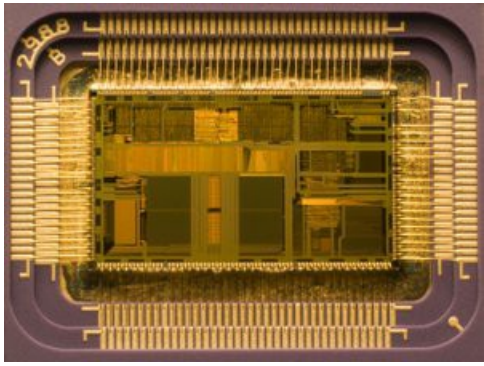


* north vs south bridge?

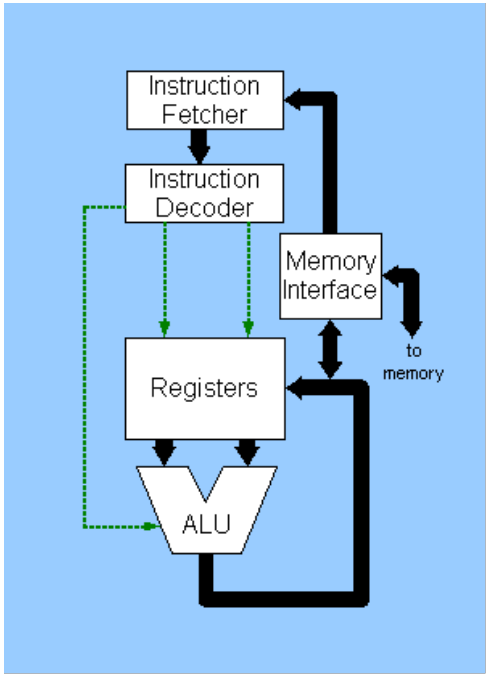


CPU





- Access
  - pins: address, data, control, status
- Internals
  - program counter (PC)
  - registers: address, data, control, flags
  - arithmetic logic unit (ALU), FPU, etc
- Benchmarks
  - clock (GHz), instructions/cycle, MIPS



CPU operations

- Fetch
  - retrieve instructions from memory (cache)
- Decode
  - instruction: operator, operands; microcode
- Execute
  - arithmetic/logic operation
  - move data between register, memory, I/O
  - change execution flow

RAM

Memory

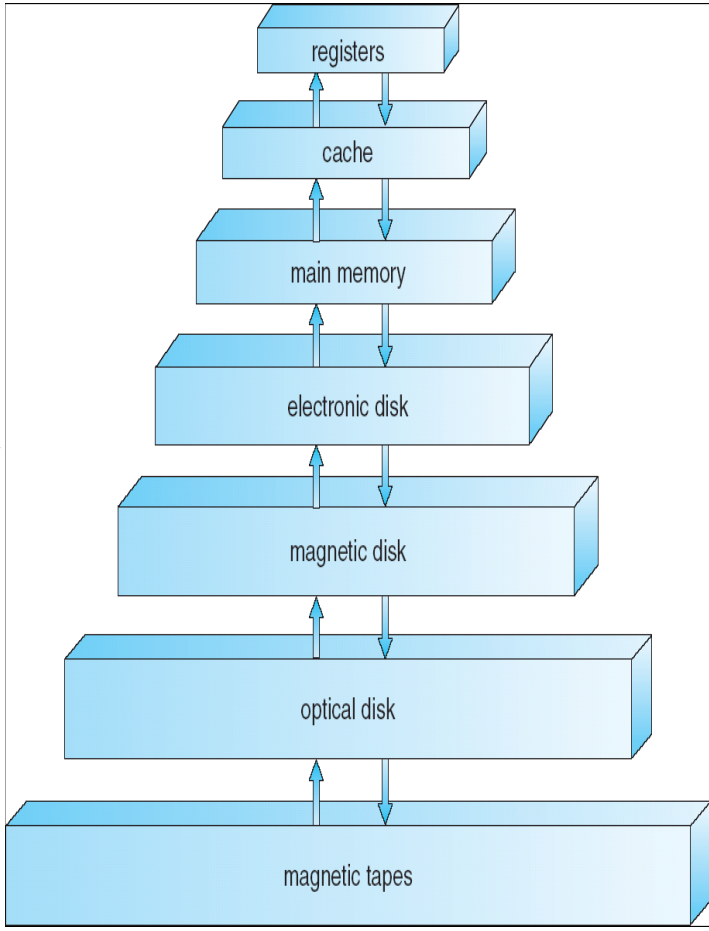
SSD

- Access
  - linear address
  - segmented address: segment, index
  - physical address: cylinder, header, sector (disk)
- Benchmarks
  - clock (MHz); read/write cycles
  - width (bits)
  - throughput (Mbps)

* there are many different kinds of memory (devices)

Memory hierarchies

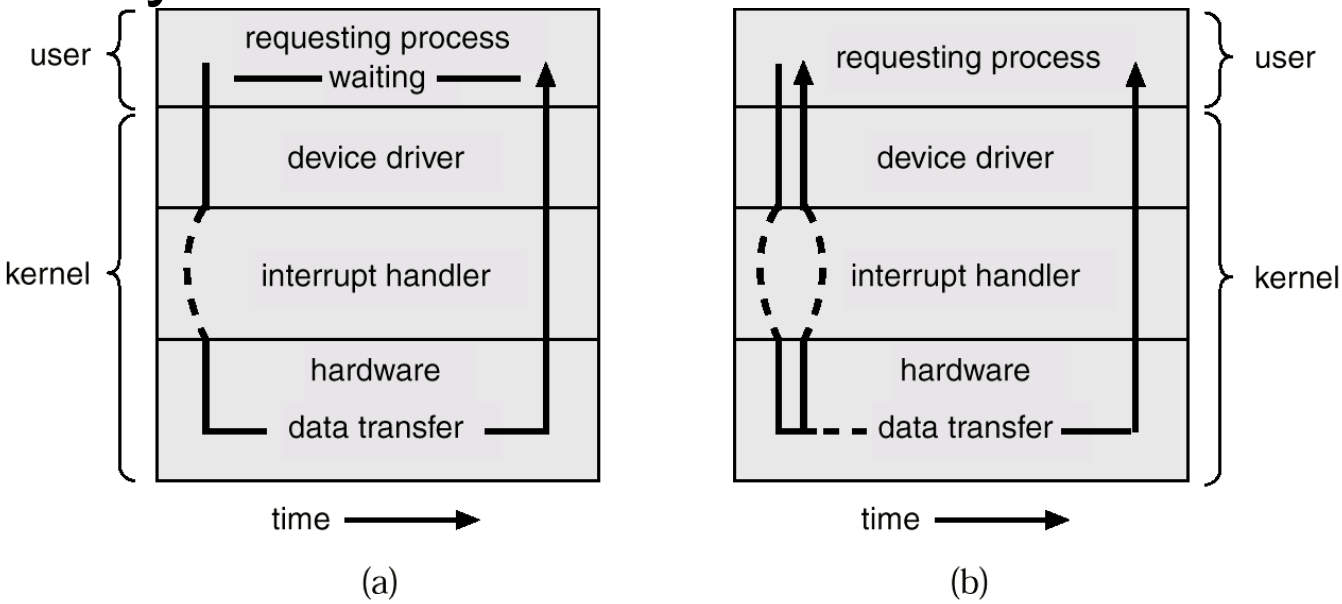
- Speed vs. size (vs. cost)
  - registers: inside CPU
  - cache: transparent to programs
  - memory: main storage
    - DRAM, DDR, SDRAM, SRAM, etc
  - disks: secondary storage
    - electronic, magnetic, optical, etc
  - tapes: backup storage
  - networked storage: NAS/SAN
- Caching



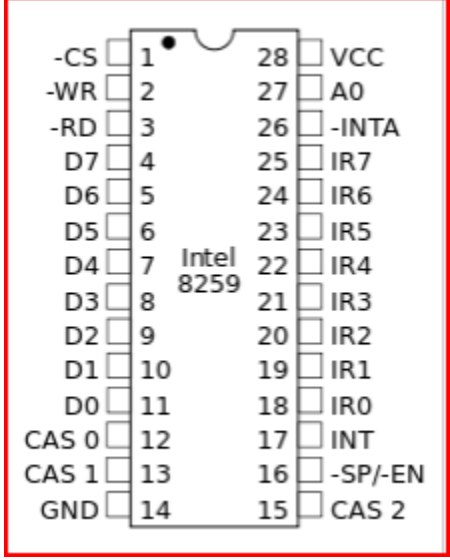
* who's the largest consumer of HDDs and tapes nowadays?

I/O

- A large variety of input/output devices
  - keyboard/mouse, video, audio, network, etc
- Access
  - Address
    - port numbers
    - I/O vs. memory space
  - Interrupt
  - Direct memory access (DMA)
- Synchronous vs asynchronous



Interrupts



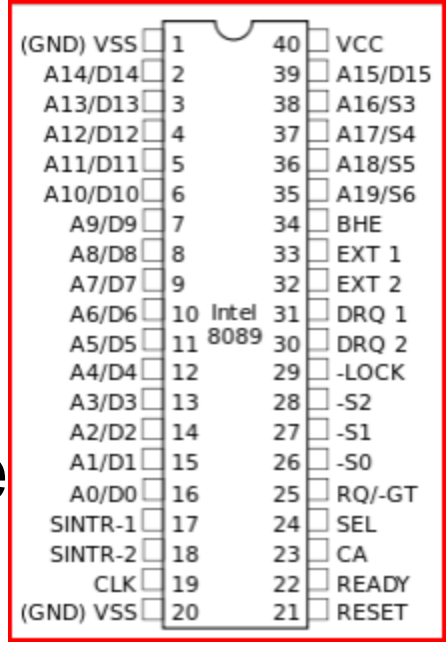
- Asynchronous operation
- Nonmaskable interrupts
  - e.g., hardware fault
- Hardware interrupts
  - hardware events: e.g., I/O completion
  - interrupt controller: priority & arbitration
- Software interrupts
  - trap, system call

Interrupt handling

- Save current state
  - CPU counters, registers, flags at system stack
- Update program counter
  - interrupt controller; interrupt vectors
- Execute interrupt routine
- Restore previous state
- Multiple interrupts
  - priority, masking, reentry

e.g., fire bell rings (for a dry run)

DMA



I/O controller

- High-speed I/O, bulk data transfe
- DMA controller
  - source/destination address
  - counter: the amount of data to be moved
- DMA handling
  - program DMA controller
  - execute DMA *concurrently*
  - issue an interrupt on DMA completion

Q: compare interrupt vs DMA

Computer architectures

- Single-processor systems
- Multi-processor systems
  - symmetric multiprocessing (SMP)
- Cluster systems
  - interconnected systems
- Distributed systems
  - networked systems
- Grid systems → Cloud systems

* the other end of the spectrum: embedded systems

OS: historical view

- Requirements evolve
- Was
  - computers were more expensive than users
  - goals: make computers more efficiently used
  - results: share computers
- Now
  - users become more “expensive” than computers
  - goals: make computers more effectively used
  - results: share users (among many computers)

* network OS; data-center OS

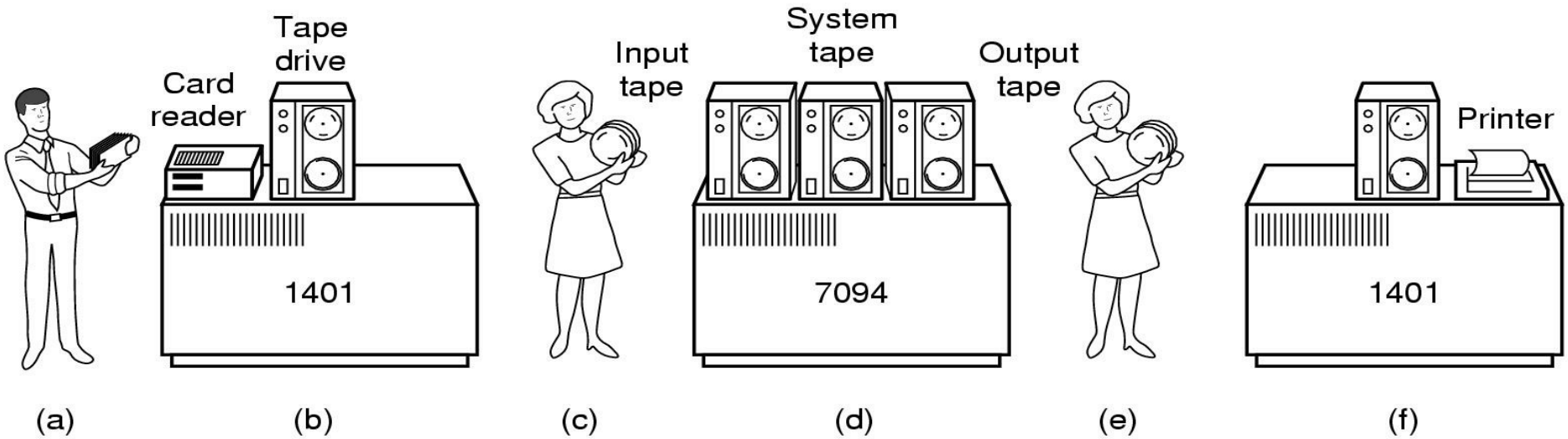
* “attention” war

OS: generations

- Uniprogramming
  - "One program at a time"
    - start, execute, {wait, execute}*, finish
    - wait for: input/output, other programs, etc
    - CPU may be idle most of the time
- Multiprogramming
  - "Many programs at a time"
    - try to keep CPU always busy
    - handle multiple programs at the same time
    - "share" (a) CPU

Batch processing

- Load a pool of jobs
- Execute one job *until* it is blocked
- Pick another one to execute



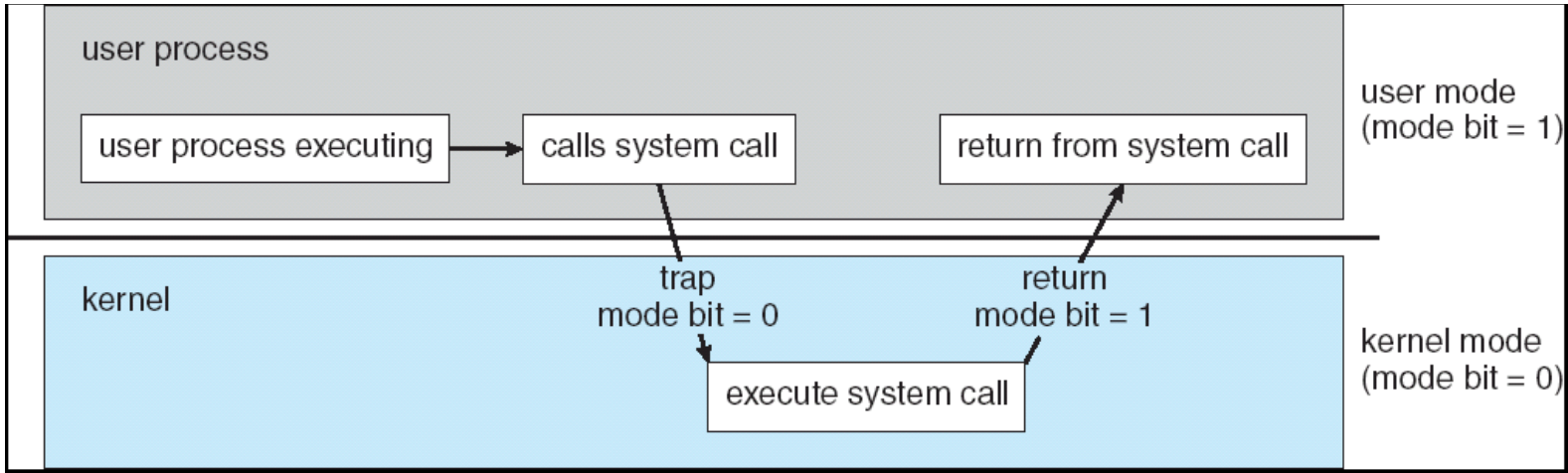
Time sharing

- Execute one job up to a certain time
  - e.g., hardware timer with counter
- Switch to another one to execute
  - job scheduling, memory swapping
- Seem to execute *many* jobs at the same time
- Batch processing vs time sharing
  - job responsiveness
  - switching overhead

Q: compare batch processing vs time sharing

OS: operations

- Interrupt the current job
  - *yield*: system call trap (e.g., I/O)
  - *yank*: hardware timer interrupt
  - how about an “abusive” job?
- Dual-mode operation
  - user mode for regular applications
  - kernel mode with privileged instructions
  - trap: user to kernel entry



Process management

- Process: a running program
  - vs thread
- Create, delete, suspend, resume process
  - resource allocation: CPU, memory, I/O, etc
- Schedule processes/threads
- Synchronize processes
- Communicate between processes
- Handle deadlocks

Memory management

- (Main) memory
  - store instructions for execution
  - store data for processing
- Keep track available memory
- Allocate and reclaim memory
  - provide protected access
  - trap invalid access
- Swap in/out (virtual) memory

* memory leak? garbage collection?

Storage and I/O management

- "In Unix, everything is a file"
  - a logical interface: open, read, write
- Create and delete files and directories
  - directory is a special file
  - file system hierarchy
- Manipulate files and directories
  - provide protected access
  - handle device-specific issues (disks, etc)

* main reason to see OS (installation disc) grow in size (drivers)

User management

- Authentication
  - who's who
  - user credentials (e.g., password, token)
- Authorization
  - what can do what
  - access control (e.g., read, write, execute)
- Accounting
  - what has been done (e.g., logging)

Specialized OS

- Different requirements and constraints
  - real-time systems
    - “hard” real-time OS in embedded systems
    - “soft” real-time OS in multimedia systems
  - handheld systems
    - almost a full-blown OS, with resource constraints
  - embedded systems
    - very severe resource constraints

This lecture

- An overview on operating systems
  - Multiprogramming
    - Batch processing vs time sharing
  - Dual-mode operations
  - Issues in
    - process management
    - memory management
    - file management

Next lecture

- Interfaces to OS
  - CLI, GUI, system calls, API
  - read OSC7 Chapter 2 (or OSC6 Chapter 3)