COMPSCI 110
Operating Systems

• Who - Introductions
• How - Policies and Administrative Details
• Why - Objectives and Expectations
• What - Our Topic: Operating Systems

How COMPSCI 110 will work

• It’s all explained on the web
  http://www.cs.duke.edu/courses/hands-on/index.htm
• Don’t expect handouts regularly
• Discussion sections
  – Goals: provide opportunity for interaction, questions answered, exploration of details that can’t be covered in lecture, problem-solving experiences.
  – Based on problems assigned from textbook
  – Bring your Nachos questions there

How COMPSCI 110 will work

• Immediate ToDo’s:
  – Form project groups - email me
    • raw@cs.duke.edu - subject: 110 groups
  • Info needed:
    – name for group
    – desired password
  • Names and emails for each member of group
  – Begin reading textbook:
    • Chapter 1
    • Next lecture - Review of CPS 104
    • First big topic: Process Mgt and Concurrency - Chapter 2
  – Read introductory material on NACHOS (see “Assignments”)
Objectives/Expectations

• What we want to accomplish today.
• What I want you to learn in this class ...
• What you can expect from me.
• What I expect from you.

What IS an OS?

• A set of “conventions” and programs which make executing “application” programs convenient and efficient.
  – Conventions: Where do programs put their output? Where does input usually come from? What form does the submitted program have?
  – Programs: I/O programs that control devices efficiently. Programs to assist in sharing memory and disk space.
“efficiency”

- Early computers VERY expensive -- $90K/month
- Keep ALL PARTS running at all times!
  - But I/O slow (100 char/sec printers) compared to CPU (1M instructions/sec) -- How keep CPU busy?
  - Interactive computing even worse -- person at keyboard takes 10 sec/input line

“Convenient” execution

- People want to
  - Compile some parts of program
  - Link it with previously compiled pieces
  - Execute result (reading input)
  - Get answers from run
  - Modify entire submission (job) and do it again
- Need “job control language”, ways to direct program input and output from/to data programmer has access to.

Techniques used

- Run multiple programs at once, so I/O delays of one occur while another computes
  - Memory must be shared
  - Disk must be shared
  - Protect one program from all others
- I/O buffering and scheduling
- “Batch” jobs are executed by interpreting the job control language of each, quickly.
What you will learn

• What an OS *does*. What services are provided, what functions are performed, what resources are managed, and what interfaces and abstractions are supported.
• How the OS is *implemented*. How the code is structured. What algorithms are used.
• Techniques, skills, and "systems intuition" (e.g., concurrent programming).
• Peaks at current research topics.

What is an OS? (generalization)

• *Resource Manager* of physical (HW) devices ...
• *Abstract machine* environment. The OS defines a set of logical resources (objects) and operations on those objects (an interface on the use of those objects).
• Allows *sharing* of resources. Controls interactions among different users.
• Privileged, protected software - the *kernel*. Different kind of relationship between OS and user code (entry via system calls, interrupts).

What is an OS? (history)

• Birthplace of system design principles; e.g., Separation of Policy and Mechanism.
• Supporting role - to provide services for the target workload, not an end product itself.
• Not the command interpreter and not a library of utility functions that can be linked into user programs.
  - These are possible implementation techniques
HW Resources to be Managed

- CPU (computation cycles)
- Primary memory
- Secondary memory devices (disk, tapes)
- Networks
- Input devices (keyboard, mouse, camera)
- Output devices (printers, display, speakers)

Working simultaneously. Shared among tasks. Concurrent demands from all directions.

Examples of Abstractions

- Threads or Processes (Fork)
- Address spaces (Allocate)
- Files (Open, Close, Read, Write)
- Messages (Send, Receive)
Main Issues in OS

- Structure
- Concurrency and Synchronization
- Extensibility, Compatibility
- Communication
- Sharing
- Naming
- Performance

- Protection, Access control, Security
- Reliability, Fault Tolerance
- Persistence, Longevity
- Scalability, Distribution
- Accounting - $$