Today: Introduction to Operating Systems

- Course Organization & Outline (handout)
- Prerequisite & Course Sign-Up (handout)
- Introduction and History of Operating Systems
  - What is an Operating System (OS)?
  - Why are Operating Systems interesting and important?
  - A little historical perspective on OS's.

Course Goals

What should you get out of this class? An understanding of:

- what are operating systems,
- what are some of their components,
- how are these components implemented, and
- how do these component interact?

Course Organization

- Course is already at capacity
- Enrollment policy: priority to graduating computer science seniors

Class Meetings

- Main lectures: Monday 1:25-2:15 and Wednesday 1:25-3:05
- Discussion section: Friday 1:25-2:15

Prerequisites and Syllabus

- CMPSCI 187 (Data structures) and CMPSCI 201 (Architecture)
- **Textbook: Applied Operating System Concepts (1st edition)** (Silberschatz,Galvin,Gagne)
- Course requirements:
  - 6 homework assignments plus a few pop quizzes (20%)
  - 4 lab projects (39%)
  - 3 exams (39%)
  - Class participation (2%)
  - Grading on a curve (more on this later)

Prerequisites and Syllabus II

- The programming assignments will use Java
- Assume that everyone is comfortable with unix (specifically, linux)
- Strict late policies and policies on cheating
  - Everything is due at 23:30 on the due date
  - Homework assignments: no credit if late
  - Lab projects: 10% penalty per day for up to 4 days (no credit otherwise)
- Everything will be handed in electronically (except tests/quizzes)
  - All submissions through the edlab machines (so you need to hone your unix skills)
  - More details to follow...

Programming Projects

- Lab 0: Get your EdLab skills down
- Lab 1: Brush up on your Java abilities
- Lab 2: Threads and Synchronization. Cooperating robots
- Lab 3: Interprocess Communication. Simple multi-player, multi-machine game of your own design
- Lab 4: File Systems. Implement your own file system that can handle directories and files of various sizes

What is Meant by Cheating?

It is Ok to:

- Discuss homework and lab assignments.
- On labs: work with one lab partner (labs 2,3,4).
- **Look** at discussions and code on the net.

It is **NOT** Ok to:

- Copy the answer of another student.
- Copy or modify an answer that you download from the net.

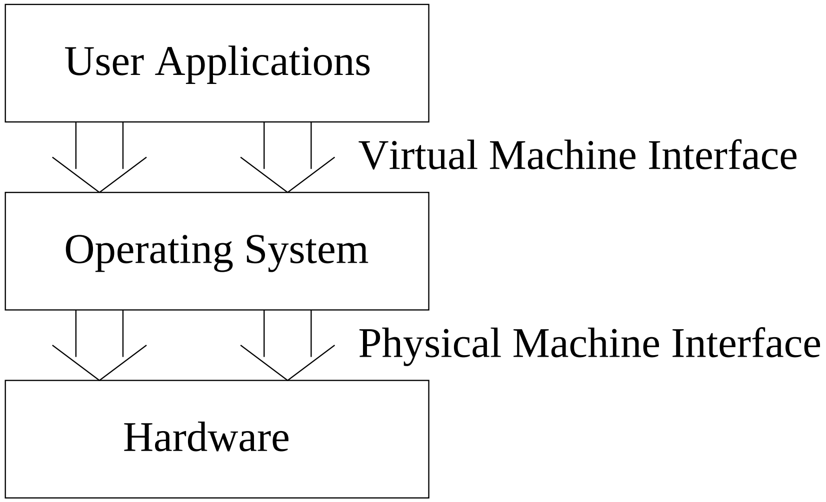
Cheating is taken very seriously!

Course Organization: Misc

- Accounts in the Ed-Lab: 35 Linux-based PCs (elnux0.cs.umass.edu - elnux34.cs.umass.edu)
- **Office hours:**
  - Andrew Fagg (Instructor): T/Th 10:00-12:00, CS 248 or by appt
  - Huan Li (TA): Wed 3:30-5:30, LGRT 220 (right across from the EdLab)
  - Vijay Sundaram (TA): Day/Time TBD, LGRT 220
- **Annoucements:**
  - All announcements will be posted on the "announcements" part of the web page.
  - Time critical announcements will be sent via email to your EdLab account. Please make sure that you check this regularly or forward your messages appropriately.

What is an Operating System?

What is an Operating System?



- Operating system (OS):
  - Interface between the user and the physical architecture
  - Implements a virtual machine that is (hopefully) easier to program than raw hardware
  - The virtual machine represents a set of abstractions that are independent of the many possible implementations of the hardware

Operating System: Salient Features

What are the salient features of an operating system?

Operating System: Salient Features

- **Services:** The OS provides a standard set of services to the hardware.
  - Examples: the file system, virtual memory, networking, CPU scheduling, time-sharing, and I/O
- **Coordination:** The OS coordinates multiple applications and users to achieve fairness and efficiency (throughput).
  - Must be able to deal with conflicting requests for services
  - Examples: concurrency (of many competing processes), memory protection, networking, and security.
- **Goal:** Design an OS so that the machine is **convenient** to use (a software engineering problem) and **efficient** (a system and engineering problem).

Why Study Operating Systems?

- **Abstraction:** How to get the OS to give users an illusion of infinite memory, CPUs, resources, world wide computing, etc.?
- **System Design:** How to make tradeoffs between:
  - performance and the convenience of OS abstractions,
  - performance and the simplicity of OS design,
  - putting functionality in hardware or software, and
  - hidden (controlled) and open OS implementations.

  As systems change, the OS must adapt (e.g., new hardware, software).
- *System Intersection Point:* The OS is the point in a computer system where hardware and software meet.

Why Study Operating Systems?

We continue to see an evolution of operating systems (within the commercial, research, and free software sectors). Also – understanding operating systems will enable you to use your computer more effectively. OS's also serve as an excellent example of system design issues whose results and ideas you will apply elsewhere.

Background: To understand this course you must have a solid basic understanding of hardware (CPU instruction sets, memory hierarchies, I/O systems, etc.) and solid programming skills (complex data structures, classes as an encapsulation mechanism, etc.)

You cannot understand the implications of how components intersect without understanding the components.

$\Rightarrow$ We will be studying a number of OS components in depth.

History of Operating Systems

Phase 1: Hardware is very expensive, humans are cheap

1. One user at a time on the console
   - One function at a time (no overlap of computation and I/O)
   - User must be on the console to debug
2. Batch processing: load program, run, print results, dump, repeat
   - Users give their program (on cards or tape) to a human who then schedules the jobs
   - OS loads, runs, and dumps user jobs
   - More efficient use of the hardware, but debugging is more difficult

History of Operating Systems

Phase 1: Hardware is very expensive, humans are cheap

3. Data Channels, interrupts, overlap of I/O and computation

- Buffering and interrupt handling in OS
- Spool jobs on drum
- No protection → One job at a time
- Performance improves because I/O and processing happen concurrently

History of Operating Systems

Phase 1: Hardware is very expensive, humans are cheap

4. Multiprogramming: several programs run at the same time, sharing the machine, i.e., I/O and CPU processing overlap.

- One job runs until it performs I/O, then another job gets the CPU
- OS manages interactions between concurrent programs
  - Decides which spooled jobs to start
  - Protects one program's memory from other programs
  - Decides which process to resume when one gives up the CPU

    A **process** *here is an executing program + supporting data*
- First OS failures
  - Multics announced in 1963, released in 1969
  - OS/360 released with 1000 known bugs

⇒ OS design and development is a science

History of Operating Systems

Phase 2: Hardware is cheap, humans are expensive

5. Interactive timesharing

- Terminals are cheap
  - Many users can interact with the system at once, debugging is easy
  - Process switching occurs much more frequently
- Memory is cheap - programs and data go on-line
  - 1 punch card = 100 bytes, 1MB = 10K cards
  - OS/360 was a stack of cards several feet high

History of Operating Systems

5. Interactive timesharing (cont)

- UNIX simplifies Multics so it can be implemented
  - Shell to accept interactive commands
  - File system to hold programs and data on disk
  - Rapid process switching to provide users with ability to interact with programs
  - Virtual memory holds lots of programs and data $\Rightarrow$ many processes can run simultaneously
- New problems - response time & thrashing
  - No control over number of simultaneous users

History of Operating Systems

Phase 3: Hardware is very cheap, humans are expensive

6. Personal computing: computers are cheap, so put one in each terminal

- Idea was to make the OS simple (again) by getting rid of support for multiprogramming, concurrency, and protection (of both the processor and of the file system).
  - Did not really work (e.g., for DOS and early versions of Windows)... Microsoft is putting all this functionality back into its OS
  - Why? Distributed computing & networking - we still want to share resources, but now we want to share across machines
- Focus moves to include user interaction (e.g., the window abstraction, mice, graphics, etc.)

History of Operating Systems

Phase 4: Hardware is very cheap, processing demands are increasing

7. Parallel and distributed computing: allow multiple processors to share resources

- In parallel systems, multiple processors are in the same machine, sharing memory, I/O devices, clock, ...
- In distributed systems, multiple processor communicate via a network
- Advantages: increased performance, increased reliability, sharing of specialized resources
- But: more processors add more overhead in coordination
- 2-4 node (Pentium) machines are now commonplace; 1000 node machines have been built
  - Symmetric: all processors are identical
  - Asymmetric: processors specialize

History of Operating Systems

Phase 4 cont: But why parallel computing?

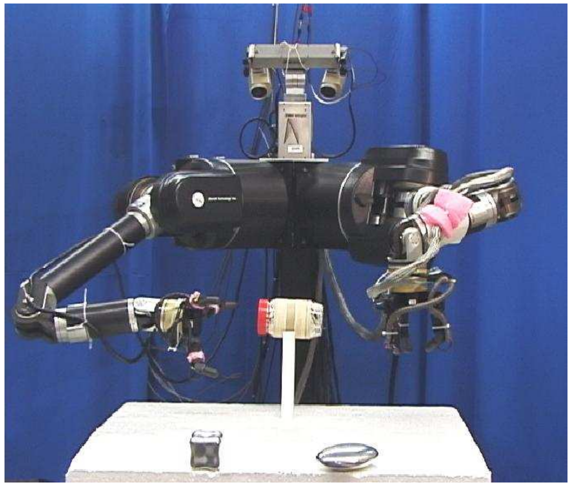
- What are the biggest machines computing today?

History of Operating Systems

Phase 4 cont: Hardware is very cheap, processing demands are increasing

8. Real-time systems allow computers to control physical machines or provide high-quality interaction as in virtual reality

- Timing requirements: tasks must be completed on time.
- Hard real-time OS must meet timing requirements. Omit features with unpredictable timing: user shell, virtual memory, disks.
- Soft real-time OS allow deadlines to be missed.



Robotics: UMass Torso

Virtual and Augmented Reality

History Lesson

Batch processing was right for its time, but it is not common anymore.

⇒**Change is one of the defining forces in computer science.**

Example

	1983	**1999**
MIPS	0.5	500
price/MIP	$100,000	$500
memory	1 MByte	1 GByte
network	10 Mbit/s	1 Gb/s
store	1 GByte	1 TByte
addressable bits	32	64

- From 1953 to now (the 40 year history of computing), 9 orders of magnitude change in almost every computer system component.

History Lesson

This degree of change has no counterpart in any other domain.

Examples:

- Transportation – over the last 200 years, we have gone from horseback (10 miles/hour) to the Concorde (1000 miles/hour) - 2 orders of magnitude.
- Communication – at the invention of the telephone (voice), TV (video) and fax (text & pictures), communication went from the speed of transportation to nearly the speed of light - 7 orders of magnitude.

Questions?

Before you go:

- Hopefuls, please see me now

Next time:

- Friday: EdLab Issues, Linux intro, and Lab 0
- Monday: Read chapters 1 & 2