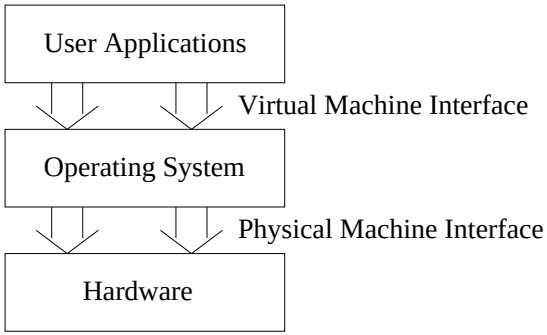
Operating Systems — Today

- Course Organization & Outline (handout)
- 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.

What is an Operating System?

An operating system is the interface between the user and the architecture.



An operating system (OS) implements a virtual machine that is (hopefully) easier to program than raw hardware.

Services: The OS provides standard services (the interface) which the hardware implements. These services include the file system, virtual memory, networking, CPU scheduling, and sharing.

Coordination: The OS coordinates multiple applications and users to achieve fairness and efficiency (throughput). Examples: concurrency, protection, networking, and security.

$\Longrightarrow$ The OS design goal is to make the machine **convenient** to use (a software engineering problem) and **efficient** (a system and engineering problem).

Summary of Operating System Principles

OS as juggler: providing the illusion of a dedicated machine with infinite memory and CPU.

OS as government: protecting users from each other, allocating resources efficiently and fairly, and providing secure and safe communication.

OS as complex system: keeping OS design and implementation as simple as possible is the key to getting the OS to work.

OS as history teacher: learning from past to predict the future, i.e., OS design tradeoffs change with technology.

Operating Systems Functionality

Key unit of management: a process - a running program

- CPU (computation) management
- Memory management
- Files, I/O, and other device management
- Distributed system and network management

Management: The OS shares, coordinates, protects, and exploits the concurrency within a single process and between processes on all the resources at once.

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, and
- putting functionality in hardware or software.

As systems change the OS must adapt (e.g., new hardware, software).

Basic Understanding: The OS provides the services that allow application programs to work at all.

System Intersection Point: The OS is the point where hardware and software meet.

Why Study Operating Systems?

The operating system makes the computer work - it is a key abstraction layer for applications, and it is the jockey that drives the horse.

Understanding operating systems will enable you to use your computer more effectively. They are an excellent example of system design issues whose results and ideas you will apply in all aspects of experimental computer science.

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.)

$\Longrightarrow$ Obviously, you cannot understand the implications of how components intersect without understanding the components.

History of Operating Systems

Phase 1: Hardware is very expensive, humans are cheap

1. One user, one process, at a time on the console
   - One process at a time
   - User must be on the console to debug
   - No sharing, no concurrency, only protect the OS from users
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 (e.g., Fortran and Pascal programs)
   - OS loads, runs, and dumps user jobs
   - Batch processing makes better use of the hardware, but debugging is much more difficult
   - Still no sharing or concurrency
3. Data Channels, interrupts, overlap of I/O and computation
   - No sharing, only protect OS from users
   - Add concurrency within the same process
   - Buffering and interrupt handling in OS
   - Spool jobs on drum
   - Performance improves because I/O and processing happen concurrently

History of Operating Systems

Phase 1: Hardware is very expensive, humans are cheap

4. Memory Protection and Relocation

- Requires: sharing, protection, and additional concurrency

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 processes to execute on the CPU
- First OS failures
  - Multics announced in 1963, released in 1969
  - OS/360 released with 1000 known bugs

$\Longrightarrow$ OS design and development is a science, and it is hard.

History of Operating Systems

Phase 2: Hardware is cheap, humans are expensive

5. Interactive timesharing

- More sharing, more protection, more concurrency
- Terminals are cheap -
  - Let all the users interact with the system at once
  - Debugging becomes a lot easier
  - 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
- UNIX simplifies Multics so it can be built
- New OS services:
  - 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 to hold lots of programs and data so that many processes can run simultaneously
- New problems - response time & thrashing
  - No control over how many users are in the system at once, at some point adding another user slows down the CPU by a factor of 1000.

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.
- Did not really work... 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

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

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

- Timing requirements provide deadlines by when tasks must be accomplished.
- 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.

History Lesson

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

$\Longrightarrow$ **Change is one of the defining forces in computer science.**

Example

	1983	**1997**	**2003**
MIPS	0.5	500	2000
price/MIP	\$100,000	\$500	\$2
memory	1 MByte	512 MByte	1 GByte
network BW	9600 bit/s	32 Mbit/s	100 Mbit/s
disk capacity	10 MByte	1 GByte	10 GByte
address bits	16-32	64	64
users/machine	100	1	1

- From 1953 to now (the 50 year history of computing), 10 orders of magnitude change in almost every computer system component.

History Lesson

This degree of change has no counterpart in any other area of business.

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.

Typical OS:

- Small OS today - 100K lines
- Big OS today - 50M lines
- 100 - 1000 people-years

Next Time:

- Processes - the unit of execution