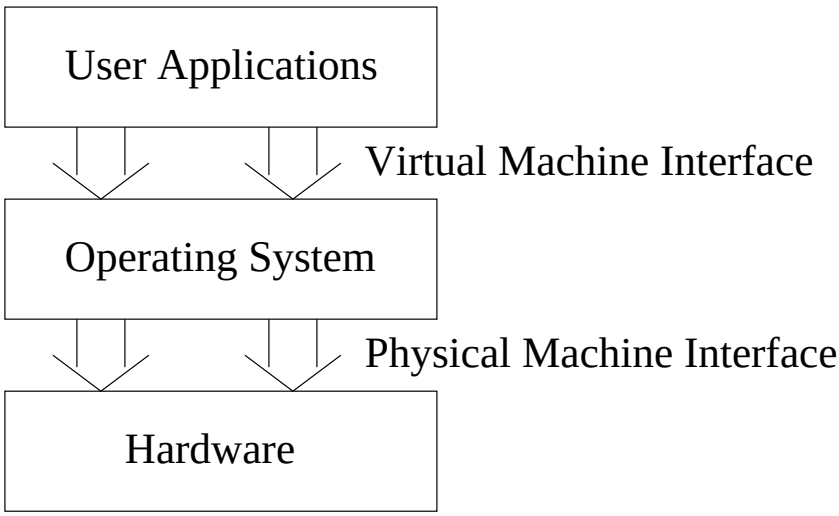
What is an Operating System?

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



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

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

and time-sharing.

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

$\Longrightarrow$ The goal in designing an OS is to make the machine **convenient** to use (a software engineering problem) and **efficient** (a system 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, 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?

Not many operating systems are under development, so you are unlikely to get a job building an OS. However, understanding operating systems will enable you to use your computer more effectively. They 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, 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 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 operator who then schedules the jobs (e.g., Fortran programs)
    - OS loads, runs, and dumps user jobs
    - Batch processing makes better use of the hardware (throughput), but debugging is much more difficult

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

   - Buffering and interrupt handling in OS
   - Spool jobs on drum
   - No protection $\rightarrow$ 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. Memory Protection and Relocation

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

- 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

History of Operating Systems

Phase 2: Hardware is cheap, humans are expensive

5. Interactive timesharing

- 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 any more.

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

Example

	1983	**2000**
MIPS	0.5	1000
price/MIP	$100,000	$5
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 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.