A little historical perspective on OS.

- Why are Operating System Interesting and Important?
- What is an Operating System (OS)?

Introduction and History of Operating Systems

Prerequisite Lecture

Outline (handout)

Today: Introduction to Operating Systems
Strict late policies and policies on cheating

Programming assignments will use Java

- Grading on the curve
- 3 exams (40%)
- 3 programming assignments (40%)
- 6 homeworks plus few in-class assignments (20%)

Course Requirements:

(Select course, CSCI 187, 201, 217, 227)


CSCI 187 (Data Structures) and CSCI 201 (Architecture)

Pre-requisites and Syllabus

Enrollment Policy

Course is over capacity.
Introduction to Operating Systems

Course Organization: Misc

Office Hours:

Discussion section to help you with lab assignments

Accounts in the EdLab: 20 Linux-based PCs, 20 Dec Alphas

Class Times:

- MF 3:00-4:00, Fri 2:00-3:00 Location: TBA
- TuTh 4:00-5:00, CS 336 or by appointment

TA: Ven V. Nguyen: vnguyen@cs.wellesley.edu

Instructor: TuTh 4:00-5:00, CS 336 or by appointment
Goal: Design an OS so that the machine is convenient to use (a software engineering problem) and efficient (a system and engineering problem).

- Examples: concurrency, memory protection, networking, and security

  - Achieve fairness and efficiency (throughput).

  - Example: the system, virtual memory, networking, CPU scheduling, and hardware implementations.

  - Example: the system provides standard services (the interface) which the user sees.

  - Operating System Interfaces (OSI)

  - Operating System (OS)
Why Study Operating Systems?

Software meets hardware.

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

**Application Programs to work at all.**

**Basic Understanding:** The OS provides the services that allow

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

- putting functionality in hardware or software,
- performance and the simplicity of OS design, and
- performance and the convenience of OS abstractions.

**System Design:** How to make tradeoffs between memory, CPU's resources, world wide computing, etc.

**Abstraction:** How to get the OS to give users an illusion of infinite

Obviously, you cannot understand the implications of how components

classes of an encapsulation mechanism, etc.)

I/O, systems, etc.) and solid programming skills (complex data structures, underlying of hardware (CPU, instruction sets, memory hierarchy,

Background: To understand this course you must have a solid basic

apply elsewhere.

Exceptional example of system design issues whose results and ideas you will

get a job building an OS. However, understanding operating systems will

Not many operating systems are under development, so you are unlikely to

Why Study Operating Systems?
OS design and development is a science

- OS/360 released with 1000 known bugs
- Multics announced in 1963, released in 1969
- First OS failures
  - Decides which process to resume when one gives up the CPU
  - Protects one program’s memory from other programs
  - Decides which program is most important to start
  - OS manages interactions between concurrent programs
  - One job runs until it performs I/O, then another job gets the CPU

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

**Phase I**: Hardware is very expensive, humans are cheap

**History of Operating Systems**

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

3. Data Channels, Interrupts, overlay of I/O and computation
- More efficient use of the hardware, but debugging is more difficult
- OS loads, runs, and dumps user jobs
- Users give their program (on cards or tape) to a human who then schedules the jobs

2. Batch processing: Load program, run, print results, dump, repeat

  - User must be on the console to debug
  - One function at a time (no overlap of computation and I/O)

1. One user at a time on the console

**Phase I**: Hardware is very expensive, humans are cheap

**History of Operating Systems**
now we want to share across machines

Why? Distributed computing over networking - we still want to share resources, but

- Did not really work - lack of activity on the network
- Multi-programming, concurrency, and protection
- Idea was to make the OS simple (e.g., 8-bit) by getting rid of support for

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

Phase 3: Hardware is very cheap, humans are expensive

| History of Operating Systems |

- No control over number of simultaneous users
- New problems - response time & thrashing
- Simultaneously
- Virtual memory hides lots of programs and data = many processes can run
- Rapid process switching to provide users with ability to interact with programs
- shell to accept interactive commands
- UNIX simplifies multics so it can be built
- Memory is cheap - programs and data go on-line
- CPU/360 was a stack of cards several feet high
- I punch card = 100 bytes, 1MB = 10K cards
- Many users can interact with the system at once, debugging is easy
- Terminals are cheap
- Interactive time-sharing

Phase 2: Hardware is cheap, humans are expensive

| History of Operating Systems |
Magnitude change in almost every computer system component.

From 1973 to now (the 40 year history of computing), 9 orders of

growth:

<table>
<thead>
<tr>
<th>1999</th>
<th>1983</th>
</tr>
</thead>
<tbody>
<tr>
<td>$500</td>
<td>$100,000</td>
</tr>
<tr>
<td>0.5</td>
<td>50</td>
</tr>
<tr>
<td>1 MIP</td>
<td>5 T V</td>
</tr>
<tr>
<td>1 GB/s</td>
<td>10 MIP/s</td>
</tr>
<tr>
<td>1 GB/s</td>
<td>1 GB/s</td>
</tr>
<tr>
<td>64</td>
<td>32</td>
</tr>
<tr>
<td>I T</td>
<td>1 T P B</td>
</tr>
</tbody>
</table>

Example

Change is one of the defining forces in computer science.

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

History Lesson

---

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

- Soft real-time OS allows deadlines to be missed.
- Hard real-time OS must meet timing requirements. "Grtfeels" with unpredictable.
- Tight requirements provide deadlines by when tasks must be accomplished.

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

- Resources: increased performance, increased reliability, sharing of specialized.
- In distributed systems, multiple processes communicate via a network
- Global, clock...
- In parallel systems, multiple processors are in the same machine, sharing memory, I/O

6. Operating system design is determined by what is currently available.

- Advantages: increased performance, increased reliability, sharing of specialized.
- In distributed systems, multiple processes communicate via a network
- Global, clock...
- In parallel systems, multiple processes are in the same machine, sharing memory, I/O

5. Users are not required to program directly on the computer.

- Operating system design is determined by what is currently available.
- Advantages: increased performance, increased reliability, sharing of specialized.
- In distributed systems, multiple processes communicate via a network
- Global, clock...
- In parallel systems, multiple processes are in the same machine, sharing memory, I/O

4. Hardware is very cheap, processing demands are increasing.

History of Operating Systems
transportation to nearly the speed of light - 7 orders of magnitude.

and fax (text & pictures), communication went from the speed of

Communication - at the invention of the telephone (voice), TV (video)

Magnitude.

(10 miles/hour) to the Concorde (1000 miles/hour) - 2 orders of

Transportation - over the last 200 years, we have gone from horseback

Example:

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