A little historical perspective on OS.

Why are Operating Systems interesting and important?

Introduction and History of Operating Systems

Prerequisite & Course Sign Up (handout)

Course Organization & Outline (handout)

Today: Introduction to Operating Systems
**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.

**Coordination:** The OS coordinates multiple applications and users to achieve fairness and efficiency (throughput).

**Time-sharing:** Examples: the system, virtual memory, networking, CPU scheduling, and hardware implementations.

**Services:** The OS provides standard services (the interface) which the

---

**Operating System: Salient Features**

- Implements a virtual machine that is (typically) easier to program than raw hardware.
- Interface between the user and the architecture.

---

**What is an Operating System?**

```
+-------------------+          +-------------------+
|                   |          |                   |
|                   |          |                   |
| Hardware          |          | UserApplications  |
|                   |          |                   |
| Virtual Machine   |          |                   |
| Interface         |          |                   |
| Operating System  |          |                   |
|                   |          |                   |
| Physical Machine  |          |                   |
| Interface         |          |                   |
```
Chapter 37: Operating Systems

Why Study Operating Systems?

Background: To understand this course you must have a solid basic knowledge of computer science concepts, such as algorithms, data structures, and programming languages. Operating systems are the foundation of modern computing, and understanding them is crucial for anyone interested in computer science.

apply elsewhere.

excelent example of system design issues whose results and ideas you will enable you to use your computer more effectively. They also serve as an excellent introduction to the field of computer science.

Not many operating systems are used in industry, so you are unlikely to get a job building an OS. However, understanding operating systems will provide you with knowledge that will be useful in other areas of computer science.

Why Study Operating Systems?

Software meets the system interface. The OS is the point where hardware and application programs meet. The OS provides the services that allow application programs to work at all.

- Basic Understanding: The OS provides these services that allow application programs to work at all.
- System Design: How to make tradeoffs between memory, CPU resources, network, etc.
- Abstraction: How to get the OS to give users an illusion of infinite processing power.
- As systems change, the OS must adapt (e.g., new hardware, software).

System Interface: The OS is the point where hardware and application programs meet.

System Interface: The OS is the point where hardware and application programs meet.
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
   - Buffering and interrupt handling in OS

3. Data Channels, interrupts, overlap of I/O and computation
   - Spool jobs on drum
   - No protection — One job at a time
   - Performance improves because I/O and processing happen concurrently

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

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
   - Removes control from one program when one gives up the CPU
   - OS design and development is a science

Fault Tolerance

OS failures
- First OS failures
- OS/360 released with 1000 known bugs
- Multics announced in 1963, released in 1969
now we want to share across machines

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

Remember is putting all this functionality back into its OS

multiplatforming concurrent, and protection

idea was to make the OS simple (generic) 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

- number of simultaneous users
- New problems - response time is threshold
- simultaneously
- virtual memory hides costs of programs and data - many processes can run
- rapid context switching to provide users with ability to interact with programs
- file system to hold programs and data on disk
- shell to accept interactive commands
- UNIX shell
- Multiuser
- memory is cheap - programs and data go on-disk
- process switching occurs much more frequently
- 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 1953 to now (the 40 year history of computing), 9 orders of

<table>
<thead>
<tr>
<th></th>
<th>1999</th>
<th>1983</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I TByte</td>
<td>I GByte</td>
<td></td>
</tr>
<tr>
<td>10 MBit/s</td>
<td>1 MByte</td>
<td></td>
</tr>
<tr>
<td>$500</td>
<td>$100,000</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td></td>
<td></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

8. Resources: increased performance, increased reliability, sharing of specialized

7. Parallel and distributed computing: allow multiple processes to share

4. Hardware is very cheap, processing demands are increasing

History of Operating Systems

- Soft real-time OS allows deadlines to be missed.
- Timing: user shell, virtual memory, disks.
- Hard real-time OS must meet timing requirements. Must tolerate with unpredictable.
- Timing requirements provide deadlines by when tasks must be accomplished.
- Provide high-quality interaction in virtual reality.

- Advantages: increased performance, increased reliability, sharing of specialized

- In distributed systems, multiple processor communicate via a network

- In parallel systems, multiple processors are in the same machine, sharing memory, I/O

- Decrease, clock, ...
transporation to nearly the speed of light - 7 orders of magnitude.

and fax (text & pictures), communication went from the speed of light to the invention of the telephone (voice). T \& (video)

Magnitude.

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

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

Examples:

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

**History Lesson**