What is an OS?

Some say:

Users → Apps → OS

"stuff between"

H/W

Two common: Standard Lib, Resource mgr

Common: Standard Library (OS as "virtual machine")

Converts H/W ⇒ something usable by apps

Why good?

⇒ makes different devices look same (uniformity)
  e.g. SCSI, IDE disk

⇒ Application can reuse common facilities (reusability)

⇒ Higher level abstractions (ease of use)

Why difficult?

⇒ what are the right abstractions? e.g.
  Too low level:
  Too high level:
Common

Resource Mgr (Coordinator)

Bottom-up approach:
- Resources make up system, OS must "manage" them

What is a resource?
- CPU, memory, disk ("anything valuable")

Why good?
- Allow _N_ users to use resource 
  \( \text{virtualization} \) 
  \( \text{multiplexing} \)
- Protect apps from each other 
  \( \text{protection} \)
- Protect OS from app
- Provide efficient (fair access) to resource 
  \( \text{performance} \)

Why difficult?
- What mechanisms?
- What policies?
- How to share effectively?
What functionality in OS?

⇒ No easy answer
  (lots of outside factors)

  \[ \text{Users/expectations} \Rightarrow \text{OS} \Rightarrow \text{Technology change} \]

⇒ OS must adapt (over time)

  ⇒ Abstractions to users
  ⇒ Algs to implement abstractions
  ⇒ Low-level implementation (HW changes)

⇒ Current OS's

  ⇒ Evolution of these things

⇒ Major trend in history

  ⇒ HW gets cheaper + cheaper

  ⇒ (Computers / Person increases)

\[ \text{HW:} \]

  mainframe \{ \text{many people (institution)} \}

  mini-computer \{ \text{group} \}

  PC \{ \text{person} \}
**First systems**

- B/G: Huge, $, slow
- I/O: punch cards, line printers

**Goal:**
- Get system to work
  - Single operator runs/debugs (reserve time, go in, etc.)

**OS**
- Simple library (no resource coordination)

**Problem**
- Machine is $$ \Rightarrow $$ need to use efficiently
  - Most of time machine is idle
- Need better performance, utilization (throughput)

**Batch Processing**

**Batch:** Group of jobs submitted to machine together
- Operator collects orders, runs 1/time
- Resource mgr

**OS:** Same as before

**Why good?**
- Lower setup costs
- Operator quite skilled at using machine
- Machine busy more (programmers off thinking)

**Why bad?**
- Must wait for results for long time

\( \Rightarrow \) Utilization $\nearrow$ Interactivity $\searrow$
Spooling

Problem: I/O takes time too!

- Old: cards => disk => printer
- New: Read, Compute, Write output

Idea: Overlap execution

Job A: R - C - W

Job B: R - W

OS: Buffering, DMA, interrupts

Good: Better throughput-utilization

Bad: Still wasting CPU during Compute (disk I/O)

Multiprogramming

=> Keep multiple "jobs" in memory
   OS chooses which to run when job waits for I/O, switch

=> New OS functions
   => Job scheduling mech/policies
   => mem mgmt/protection

=> Good: Better throughput
   Bad: Still not interactive
Even Cheaper

(Mainframes $\rightarrow$ Minicomputer $\rightarrow$ Personal Computer

50's/60's $\rightarrow$ 70's $\rightarrow$ 80's $\rightarrow$

Goal of

Response time!

Concept: Time-Sharing

Switch between jobs to give "appearance" of dedicated machine

Good: submit jobs, immediate feedback

OS functionality

- more complex scheduling, mem mgmt
- concurrency control, synchronization

PC

Whole computer cheap ($1/user!$)

Don't need so much functionality!

- time-sharing, protection, VM,

$\Rightarrow$ OS is subtractive again (DOS)

$\Rightarrow$ H/W + users $\Rightarrow$ OS functionality
Current Trends

\{ 
\Rightarrow \text{ Huge systems?} \\
\Rightarrow \text{ Tiny systems} 
\}

Complexity
\Rightarrow \text{ millions of lines of code?} \quad \text{at odds}

Robustness
\Rightarrow \text{ crash-proof} (?)

Secure
\Rightarrow \text{ not easy to compromise}

Why study OS?

Tangible
\Rightarrow \text{ You may work for Solaris group at Sun} \\
\Rightarrow \text{ Administer / use / program system well}

Intangible
\Rightarrow \text{ Curiosity} \Rightarrow \text{ How stuff works} \\
\Rightarrow \text{ Broad understanding of CS} \\
\Rightarrow \text{ H/W, P/L (synch), Algorithms, Performance} \\
\Rightarrow \text{ Challenging:} \\
\Rightarrow \text{ How to design/implement large/complex systems?}