https://commons.wikimedia.org/wiki/File:Bletchley_Park_Bombe4.jpg
https://en.wikipedia.org/wiki/File:UnivacII.jpg
https://photos5.appleinsider.com/gallery/29370-47319-000-3x2-Apple-History-Mac-launch-xl.jpg
https://commons.wikimedia.org/wiki/File:Raspberry_Pi_4_Model_B_-_Side.jpg

Operating Systems

What is an OS?

- No accepted definition
- Two common pieces:
  - **Kernel** (protected software)
  - **Libraries** (shared, unprotected software)
- Together, these create the “world” for your program
  - What it can do, how it thinks

Operating Systems

What does an OS do?

- Coordinator
  - Assigns resources to programs fairly
  - Manages sharing, enforces protection
- Standard abstractions & routines
  - Builds complex things (ex: files, directories)
  - From simple things (ex: blocks on disk)

Operating Systems

The Problem:

How to build a system that is

- Safe
- Fast
- Flexible

but makes it easy to write complex programs?

Operating Systems

History

OS - History

- No OS in the first computers
- Ran a single program
  - Originally, programmed by plugging wires!
- Reboot to run another

- Was still common in personal computers until late 80s, early 90s !

OS - History

Problem

- Hardware changes over time (upgrades, peripherals, etc.)
- Don’t want to rewrite all of our programs, every time HW changes

Insight

- Many peripherals are similar, at a high level

OS - History

- Solution: **Device Drivers**
  - Also TSRs

- Zero protection
- All code has the same permissions
- Still only using one program at a time

What is a Device Driver?

- Additional software added to OS (or program)
- Software abstraction
  - Common interface: “printer”
  - Specific device: HP Printer 123 on parallel port

  - Common interface: “disk”
  - Specific device: WD 789 on IDE port

OS - History

Problem

- Want to share the computer
- But programs assume that they have *complete* control

OS - History

- Solution: Batch Computing
  - Submit your program to a queue
  - Runs when it’s your turn
  - Complete control of the computer while you run
  - Reboot the computer to start the next job

- Allows users to share hardware
  - Painful to use, long latency on runs

OS - History

Problem

- Users want quick response to their programs
- Users want interactive **shells**

Insight

- Most users don’t use the full power of the machine, most of the time

OS - History

- Solution: Time Sharing
  - Multiple programs running at the same time
  - Shared resources: CPU, memory, disk, I/O
- First example of **multiprocessing**
  - Each process pretends that it controls the entire computer
  - OS manages sharing of resources

OS - History

Classic Time Sharing

- Many users logged in at once
- Shared storage on disk, shared RAM
- Need protection of programs from each other
- Need permissions & control systems

- Common in business/academia in 70s

OS - History

PCs: Time Sharing, Simplified

- Assume only one user per computer
- Very few protections, no security
- Software generally trusted to do the right thing
    - Simple but dangerous!

- Common in personal computers in 90s
    - Added security later

OS - History

Question

- Why allow multiple programs to run at once if there is only one user?

Think, Pair, Share:

Why do you need multiprocessing on your computer today?

Back in the 90s – with no network and only simple media devices – why would multiprocessing still be useful?

OS - History

Question

- Why allow multiple programs to run at once if there is only one user?

Answer

- Users often do multiple things at once!
  - Print a document
  - Play music
  - Check email / social media

OS - History

Problem

- Many, many devices in a single computer
- Many need very high-speed interaction
  - Network, USB, etc.
- But not have anything *interesting* to do for ages

Insight

- What if the OS only talked to the device when something interesting has happened?

OS - History

- Devices Gradually Get Smarter
  - Devices start doing non-trivial work on their own
    - Don’t need CPU except occasionally
    - Often have a small on-board CPU (later: GPU)
  - Printers
  - Sound cards
  - Graphics
  - Disks
  - Network

OS - History

- Modern OSes
  - 100s of programs at once
  - Sometimes, many users at once, through networks
  - Programs can die in microseconds, or live for years
  - Pre-emptive multitasking

- Now the norm in all computers except microcontrollers. Even the Raspberry Pi has pre-emptive multitasking and strict user control!

OS - History

- Multi-core, Multi-processor
  - Multiple CPUs running at the same time
  - Share the same memory
  - Hordes of race conditions

- Have existed almost since the beginning
  - Mostly only in servers until late 90s
  - Now, in all computers other than microcontrollers

OS - History

- Distributed Computing
  - Multiple physical computers
  - Each computer has its own OS
  - Libraries make it easy to communicate
  - Some programs think of the entire system as one unit

OS - History

- Virtual Machines
  - A program simulates a computer
    - Including attached hardware (disk, keyboard, etc.)
  - Install & run real OSes inside it
    - It controls its own programs
- Containers
  - Run programs inside an ordinary OS
  - Lock down what it can see about other programs, files
  - Almost as good as a VM, and **much** cheaper

OS - History

- Our simulator (USLOSS)
  - Not even a VM
  - Just a toy for playing around with OS concepts
  - Lets you simulate a few things about an OS
  - But lots of hacks to make it simple