A complete picture of how programs execute?

CPS 6, 100, 108: ideas into high-level languages (e.g. Java)

CPS 104: high-level program into machine-language program; how processor executes machine-language program

What is missing?
What is an Operating System?

A software layer between the hardware and application programs

- A program that manages the hardware and makes it easier to use by applications by presenting nicer abstractions (i.e. the virtual machine).

- For any area of OS (e.g. threads, virtual memory, file systems, networking) ask:
  - What interface does the hardware present to the software (physical reality)?
  - What interface does the OS present to its applications (the nicer abstraction)?

Relationship between user programs and operating system

First perspective: user programs are main programs and get services by calling into the kernel (via system calls).

Second perspective: OS is the main program, calls user programs as sub-routines.
Two main functions of an operating system

Illusionist
- Makes computer appear to be more and/or better than it actually is.
- Examples?

Government
- Parcels out the single shared hardware resource to multiple competing applications fairly and efficiently.
- What hardware resources does the OS manage?

Why study operating systems?

The functions of an operating system (illusionist, government) appear in many other domains (e.g. web server farms, Google, concurrent applications)

Many of the principles we study are applicable outside of operating systems
- Design and implementation of abstractions
- Caching, indirection, concurrency, authentication, naming, atomicity, protection
- Resource multi-plexing

Fun to open the hood and understand how the machine you’ve been using actually works.
**History of operating systems**

OS history is dominated by two trends

- Hardware used to be really expensive and its cost has decreased dramatically
- The OS has become increasingly complex

Single operator at a console

- Goal: basic functionality
- Interactive

- OS is simple: one thing happening on the computer at a time. OS is just a library of standard services.
- Leads to poor utilization of computer (idle while person thinks and while I/O devices are working).
- Hardware used to be really expensive, so utilization mattered.

**Batch processing**

- Goal: improve CPU and I/O utilization by removing user interaction
- Solution: submit job and wait for final answer, no human interaction during session

- Still only one job at a time
- Now OS is batch monitor and library of standard services
- Batch monitor:
  - Load program
  - Run it
  - Print results
  - Load next program
- Protection starts to matter. Must ensure that batch monitor can run the next program in the queue

Why wasn’t this an issue when there was a single operator at the console?
Multi-programmed batch

- Goal: improve CPU and I/O utilization by overlapping CPU and I/O
- Solution: overlap disk I/O with CPU

- Allows multiple I/Os to happen simultaneously
- Allows CPU and disk to work simultaneously
- OS getting more complex: OS switches between multiple processes, manages multiple I/O devices
- OS needs to protect processes from each other

Time-sharing

- Goal: restore interaction between humans and programs
- Insight: humans can be modeled as a (very slow) I/O device
- Solution: switch between programs when waiting for users, switch back in time to get the next keystroke
- OS getting more complex: lots of simultaneous jobs, multiple sources of new jobs

Personal computing

- Computer hardware gets cheaper, making it economically feasible to go back to a single operator at the console
- OS becomes a library of routines again
- Do we need to time-share between multiple simultaneous jobs?

- Do we need protection between jobs?

- PC operating systems have gradually added back most of the features of time-sharing systems