

EECS 482
Introduction to Operating Systems

Professor Peter Chen

A complete picture of how programs execute?

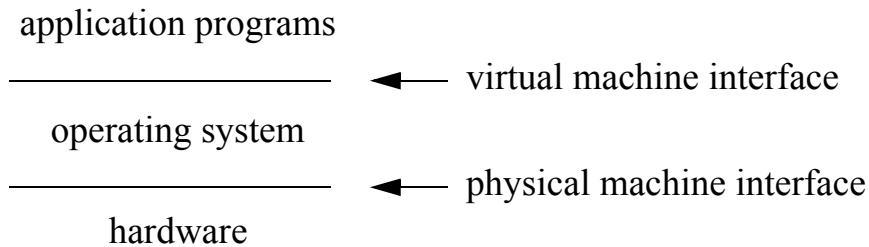
EECS 280, 281: ideas into high-level programming language
(e.g. C++)

EECS 370: high-level program into machine-language program;
how processor executes machine-language program

Any missing pieces?

What's 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 application by presenting nicer abstractions (i.e. the virtual machine)

For any area of OS (e.g. threads, virtual memory, file systems, networking, security) 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 program is the main program and gets services by calling the kernel

Second perspective: OS is the main program, calls user programs as subroutines

Two main functions of an operating system

Illusionist

- makes computer appear to be more or better than it really is
- examples?

Government

- parcels out the single shared hardware resource to multiple competing applications efficiently and fairly
- what hardware resources does OS manage?

Why study operating systems?

You may write part of one

The functions of an operating system (illusionist, government) appear in lots of domains (not just OS), e.g. web server farm, cloud computing, concurrent applications

- lots of techniques used in OS are used in other domains
- designing and implementing an abstraction
- caching, indirection, concurrency, authentication, naming, atomicity, resource allocation

Fun to “open the hood” and understand how the thing you’ve been using really works

History of operating systems

OS history is dominated by two trends

- hardware started out VERY expensive, but cost decreased dramatically
- increasing complexity of OS

Single operator at 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
- poor utilization of computer (idle while person thinks and while I/O device works)

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 execution

- still only one job at a time
- OS is batch monitor and library of standard services
- batch monitor loads program, runs it, prints results, starts next program
- protection starts to become an issue. Must ensure that batch monitor can run the next program in the queue.

Why wasn't this an issue for "single operator at console"?

