Lecture 1 —Introduction

Jeff Zarnett
jzarnett@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

April 26, 2015

Course Syllabus

As our first order of business, let's go over the course syllabus.

Collaborative Course

The source material for the ECE 254 notes and slides is now open-sourced via Github.

If you find an error in the notes/slides, or have an improvement, go to `https://github.com/jzarnett/ece254` and open an issue.

If you know how to use `git` and LaTeX, then you can go to the URL and submit a pull request (changes) for me to look at and incorporate!

Introduction to Operating Systems

Operating systems are those programs that interface the machine with the applications programs. The main function of these systems is to dynamically allocate the shared system resources to the executing programs.

- What Can Be Automated?: The Computer Science and Engineering Research Study, MIT Press, 1980

Introduction to Operating Systems

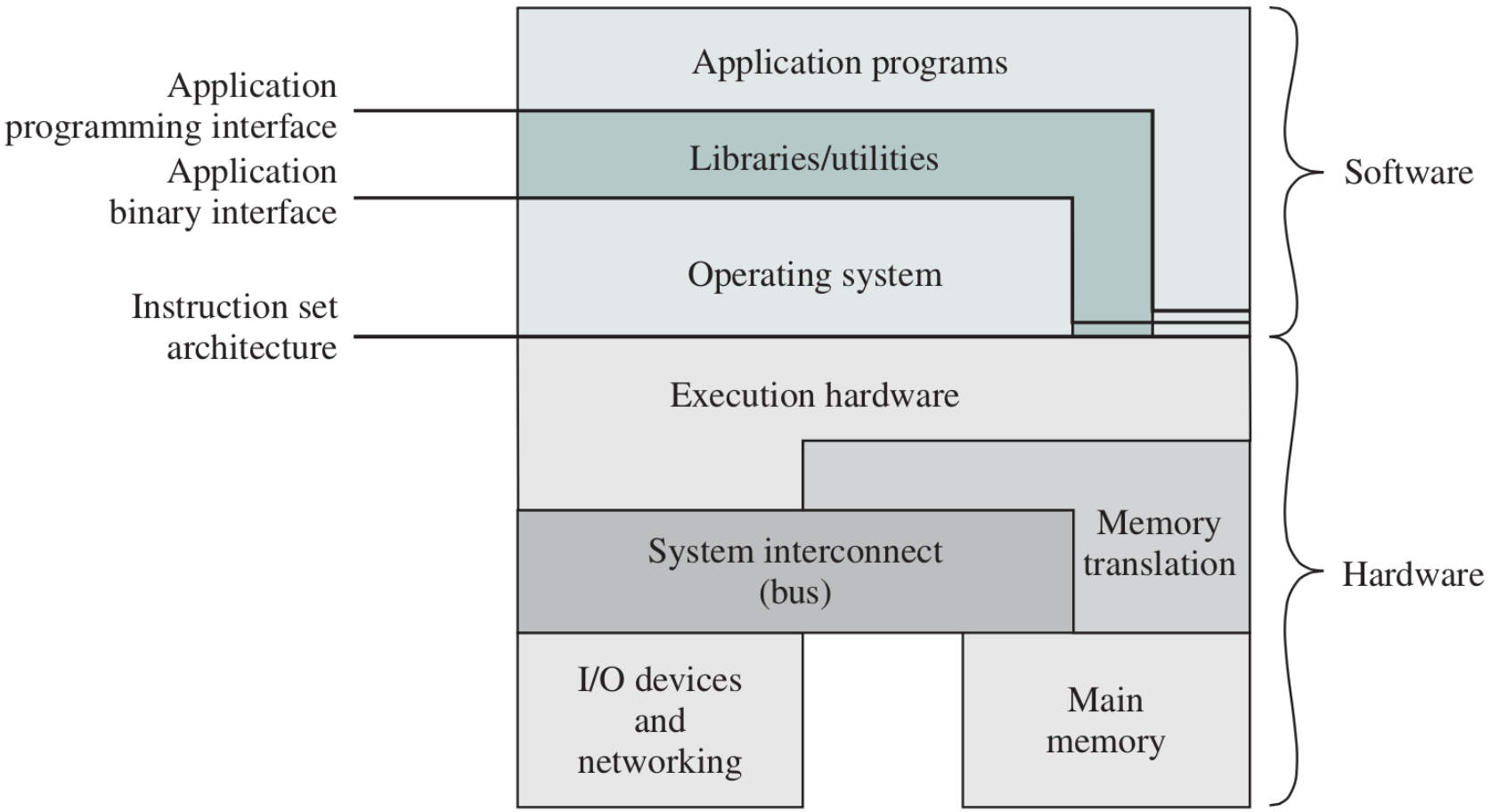
An operating system (OS) sits between the hardware and programs.

It does many different things.

It has many often-conflicting goals.

You might think of the OS as the “secretary” of the system.

Structural Diagram of a Modern Computer



OS: Resource Manager

The OS is responsible for resource management and allocation.

Resources like CPU time or memory space are limited.

The OS must decide how to allocate & to keep track of system resources.

In the event of conflicting requests, choose the winner.

OS: Environment Provider

The OS enables useful programs like Photoshop or Microsoft Word to run.

The OS is responsible for abstracting away the details of hardware.

This is so program authors do not have to worry about the specifics.

Imagine Hello World had to be written differently for different hardware.

OS: Multitasking

Multiple programs means some resources are shared.
$\rightarrow$ A source of conflicts!

OS creates and enforces the rules so all can get along.

Sometimes processes want to co-operate and not compete.
The OS can help them to do so.

OS: Efficiency

Another goal may be to use the computer hardware efficiently.

Any moment when the supercomputer is not doing useful work is a costly waste.

There are always programs eager to run.

OS: What is it, really?

Operating systems tend to be large and do a lot of things.

We expect now that an OS comes with a web browser, an e-mail client, some method for editing text, et cetera.

The part of the operating system we will study is the Kernel.

The kernel is the “core”; the portion of the OS that is always present in main memory and the central part that makes it all work.

OS: Evolution

Operating systems will evolve over time.

There will be new hardware released, new types of hardware, new services added, and bug fixes.

Evolution is constrained: a need to maintain compatibility for programs.

Microsoft Windows: strict devotion to not breaking binary compatibility.

Systems Programming

Operating systems themselves are not the whole picture.

Systems programming: the next layer above the OS itself.

Useful tools, but they are not part of the kernel.

Some examples of systems programming:

- **File Manipulation**
- **Status Information**
- **Programming-Language Support**
- **Communication**

Systems Programming

Programming at this level is more difficult than regular programs.

It may require knowledge of the hardware, or perhaps programming facilities like debugging are limited.

Systems programs must take concurrency into account.

A Short History of Operating Systems

OSes evolved over the years, typically in conjunction with hardware.

Let’s look at four generations.

First Generation

The vacuum tube generation.

In the 1940s/early 1950s, some devices recognizable as digital computers, but no recognizable operating systems.

Programming was done by wiring up the connections manually or perhaps connecting cables or flipping physical switches.

One program could run at a time.

Sign up sheet to reserve a block of time.

1950s: big improvement – the punch card!

Second Generation

1955 to 1965: the era of the mainframe.

Mainframes are extremely large and require professional operators.

Incredibly expensive: for governments, large companies, universities.

Programmers create a “job”.

Operator gets a deck of punch cards and puts it in.

Second Generation

An operating system, of sorts: an operator manages each job.

This is inefficient. Better idea: batch system.

Batch: group up a bunch of jobs and do them together.

Reduce the amount of time the mainframe is idle between jobs.

Use more limited systems to do input/output.

Third Generation

1965-1980: the era of integrated circuits (ICs) and multiprogramming.

IBM attempted to introduce the System/360.

This did not go well, prompting the writing of the book “The Mythical Man-Month”.

IBM was on the right track, at least.

Third Generation

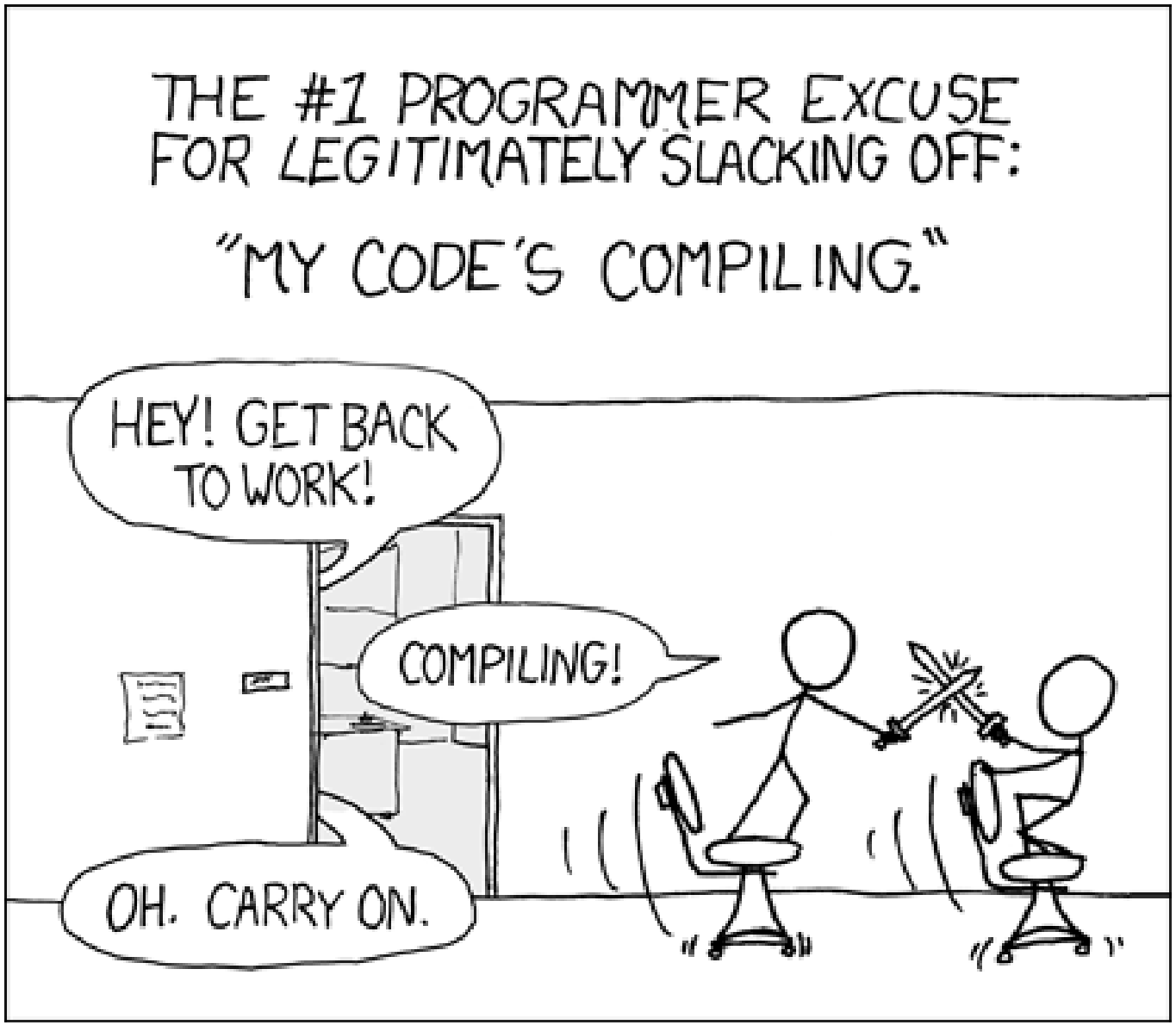
These systems were still operating on the batch principle.
    Keep the CPU busy.

In many older mainframes the CPU was the limiting factor.

In commercial applications, input/output was often the limiting factor.

Programmers submitted a job; it could take a long time to get results.

Third Generation



Third Generation

Next major innovation: timesharing.

Each user gets a terminal connected to the central computer.

More interactive and responsive system.

Compilation might be quick if it's a small block of code.

While I am reading the output of the compiler (`SYNTAX ERROR LINE 5`), the CPU can be working on something else.

New problem as multiple programs are running at once: the various programs need to be protected from one another.

Third Generation: UNIX

UNIX emerged in the third generation.

The history of UNIX is long and varied, but it is arguably the most successful operating system ever.

Almost all PC operating systems and many mobile ones are either descended from UNIX or an (often poor) attempt to reinvent it.

IEEE developed a standard, called *POSIX* that versions of UNIX support.

Linux is a clone of UNIX intended to be compatible with and implement the POSIX standard.

Fourth Generation

From 1980: the age of personal computers.

The IBM PC and the MS-DOS own computers on their desks and in their homes.

Up to here interaction with the computer was all text based.

Apple capitalized on an invention by Xerox called the GUI.

OSes became more “user-friendly”.

Fourth Generation

The GUI is more than just window dressing.

GUI: single user starts to have multiple programs running.

Protection between the various programs is needed.

Wrapping Up

The history of the OS is obviously more complex than this.

In the last 35 years, much has changed, but much has stayed the same.

We will focus on UNIX-like systems.

- This includes Linux, Mac OS X, Android, Solaris...
- Windows is different, but has a wide install base.