What is an Operating System

Hi everyone, welcome to COS 417, our new and improved operating systems course. My name is XYZ, we're going to jump right in.

Today we're going to talk about what operating systems are and why we need them; we'll narrow in on the specific kinds of operating systems and computers we'll be focusing on this semester; and we'll cover some course administrativia including the first reading programming assignments.

What is an Operating System?

That's actually a hard question to answer, but to pharaphrase Justice Potter Stewart, at least to a degree, we know it when we see it.

Well hopefully you all have at least an intuition for what an operating system is or else you may not have chosen to take this class, so just to warm us up: who can give me some examples of operating systems?

Expecting:

- Windows
- Mac OS
- iOS
- Android
- Linux
- FreeBSD, etc…

Not expecting, but would be cool:

- Firefox/Chrome/whatever-browser
- AWS
- Java Virtual Machine

(If people mention both Android and Linux, good to hit on the note that Android is really an environment built atop a Linux kernel and userland, but Android applications can't generally run on Linux and most, particularly graphics, applications can't really run on Android. So are they different or the same operating system?)

So a good definition of an operating system is hard to come by. Let's try out some bad ones.

1. The stuff that an operating systems vendor (Microsoft, Apple, Red Hat, etc) provides when you buy an operating system at a store.

The first problem with this is that buying an operating system at a store is pretty rare these days. The second problem is that a whole variety of applications often come installed with the operating that don't pass the smell test. Is Garage Band part of the operating system? Is the calculator?

2. The program that runs first and is always running; that spawns and manages everything else.

The problem with this is that program we most associate with an operating system—a kernel—doesn't actually fit this definition. On most computers, the first program that runs is on a secondary chip on the motherboard which initializes much of the hardware including the main processor, followed by some firmware like a BIOS or UEFI, followed by a bootloader, such as GRUB or uboot, followed often by a version of a kernel that initializes more of the system but then replaces itself with another kernel.

There are also many other programs that always run on different parts of a modern computer, which is really composed not of one processor, but dozens or hundreds, typically at least one for each hardware peripheral.

For now, we'll get comfortable with the ambiguity of not really knowing what

Why do we need an Operating System?

Once upon a time (1950s-1960s), there were no operating systems and there were big giant computers that cost the equivalent of $20M and lived in whole wings of buildings. But much like today, a computer could only really do one set of things at a time. Also, much like today, there was more than one thing to do. So, they had human gatekeepers. If you wanted to run a program to, say, compute whether a rocket would end up in space or the ocean, you walk up to the gatekeeper, hand them your program on a big deck of cards, and they would decide whether and when to run it. When it ran, it was the only thing that ran until it finished (or the gatekeeper decided it was finished).

That gatekeeper was responsible for deciding how this expensive resource was shared, and making sure that programs didn't cause future programs to run incorrectly (typically manually "cleaning up" the states of various transistors between program runs). They also translated between the programming medium (generally cards) to the hardware's medium for interpreting programs (generally tape), as well as ensuring the computer ran as efficiently as possible.

Later, the advent of "mini-computers" which took up only a large part of a room rather than most of a building and cost a mere $1M (in todays dollars), meant that while each mini-computer was to be shared by dozens of people it no longer made sense to employ a whole entire extra human just to mediate who's program ran when. And thus, the job of the gatekeeper was automated away and the first generation of operating systems was born.

They were programs that largely did the same job as their human precessesors. A mini-computer at Bell labs, say, may have had a dozen scientists who showed up to work every day at 9am, and wanting to use it to compute something to do with electrons or switchboard circuits, or whatever it is that scientists at phone companies computed at the time. The operating system was responsible for mediating access to the computer's resources, ensuring that programs running concurrently didn't affect each other's correctness, providing some consistent interface to the hardware so programs didn't break when the VP of finance bought them the next generation of mini-computer, and otherwise tried to avoid making things too slow.

Fast forward to today. Computers are cheap and small enough that there are more than half as many desktops, laptops, smart-phones, and servers as there are humans in the world. In developed countries like the US, the vast majority of people have at least one personal computer. And none of this includes the hundreds of more invisible computers that the typical person interacts with regularly from dozens in each automobile, to HVAC systems, the power grid, medical devices, you name it.

Most of these computers are dramatically more powerful than the expensive shared mainframes of the 60s, many of them have only one user who has access to multiple computers at a time, and they are more often used to play Angrybirds than compute the velocity of a rocket.

Nonetheless, they still almost always run at least two programs at a time. And do that well, we need an operating system.

- Resource Sharing
- Safety / Security / Correctness
- Performance
- Portability

Here's the really magical bit—the operating systems we use on most of these computers are, in their very essense, the same as one those first generation operating systems from the 60s. And, in fact, this class of operating system is the one we'll be primarily focused on this semester.

Does anybody know which operating system I'm talking about?

UNIX

In 1969, Jimi Hendrix plays a legendary closing set at Woodstock, police raid the Stonewall Inn, Buzz Aldrin becomes the first person to walk on the moon, Gaddafi comes to power in Libya, and Ken Thomson and Dennis Richie begin working on the UNIX operating system.

The UNIX file system, kernel architecture, and process model, including the specifics of the core API for manipulating processes, are basically unchanged since the early versions of UNIX to todays Linux, Mac OS, the various BSDs and most other common operating systems.

What resources does a computer have?

So, a core job of the operating system is to mediate resource. But what are these resources?

To the students: what are the resources in a computer that programs might care to access?

Try to get to:

- CPU
  - One or more processors
  - Independent and shared caches
  - Accelerators (floating point, vector instructions, cryptography)
- Memory
- Peripheral computation units
  - GPUs
  - TPUs
  - TPMs
  - "HW" root-of-trust, etc
- Persistent storage devices
- Network
- Human-Machine-Interfaces (screens, input devices, speaker, microphone)
- Sensors
- Actuators

So What is an Operating System?

An operating system is a set of choices about what resources to expose to programs, the abstractions through which to present those resources, and, as importantly, how to guarantee these contracts.

We'll spend this semester learning and understading which choices the predominating operating systems have made, and we'll explore some alterative choices for each and their implications.

In addition you’re going to do a lot of systems programming. You’ll practice interacting with the interfaces an operating system provides as well as implementing a number of operating system kernel components in your programming assignments.

Course Administrativia