

Introduction to CS350

Ali Mashtizadeh and Lesley Istead

David R. Cheriton School of Computer Science
University of Waterloo

Fall 2019

Welcome to CS350 - Operating Systems!



- Administrative Information
- Introduction to Operating Systems

Important links:

- <http://www.student.cs.uwaterloo.ca/~cs350>
Course personnel, office hours, readings, assignments, tutorials, previous midterms, review problems, etc.
- <https://piazza.com>
Piazza will be used for announcements, extra notes, questions, corrections, etc. Please check piazza regularly. **Do not post your code in public piazza posts; use private posts when appropriate.**

Course notes are **required**.

They are **NOT** designed to be standalone. Come to class, take notes. Notes are available online from the course website. You may also purchase a printed copy, if you desire.

Textbook is **NOT** required, but highly recommended.

Operating Systems: Three Easy Pieces

Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau

Textbook is available **FREE** on-line. Link to the text is available on course website. All recommended readings are linked on course website.

Grading Scheme

A0, *A1*, *A2a*, *A2b*, *A3*: Assignment marks as a percentage

M: Midterm exam grades as percentages

F: Final exam grade as a percentage

$$\text{Normal} = (0.02 * A0 + 0.1 * A1 + 0.11 * A2a + 0.11 * A2b \\ + 0.11 * A3) + (0.20 * M + 0.30 * F)$$

$$\text{Exams} = (0.20 * M + 0.30 * F) / 0.50$$

```
if (Exams < 50%) {  
    FAIL EXAMS, FAIL THE COURSE  
    Course Grade = min(Normal, Exams)  
} else {  
    Course Grade = Normal  
}
```

You **WILL FAIL** this course if you fail the weighted exam average, regardless of your assignment grades.

There are **5** assignments.

All assignments are to be done **individually**.

You will not be writing your own OS.

You will be adding/fixing features of an existing OS.

We use **OS/161** (~22,000 lines for kernel), which runs on **SYS/161** (MIPS simulator/VM)

Slip days:

- Allows flexibility in assignment deadlines
- Total of 5 slip days
- Can use maximum of 3 slip days per assignment

READ AND UNDERSTAND INFO ON COURSE WEB PAGE
This course has extra requirements and ignorance is no excuse!

Do not use code from other sources:

- Do not copy code from friends, web sites, or other sources
- Do not search for or look at other code for any reason
- Avoid blogs that provide instructions
- We use VERY GOOD cheat detection software
- Every term people are caught
- Often: 0 on assignment and -5% off final grade

Other than websites identified in the course, it is acceptable to use the web to

- understand the lecture material, learn how to use Git, bmake, GDB, and other tools used in this course

But it is **not acceptable to use the web** to

- get an idea of how to approach the assignment,
- copy or view code that may help you do the assignment

It is **acceptable to consule with other students** to

- get an idea of how to approach the assignment
- get an idea of how to overcome a stumbling block or fix a bug.

But it is **not acceptable** to

- view another student's code or have another student view your code.

IF you have taken this course before, you may reuse your previous code if:

- You ask your instructor for permission
- Your code was not subject to previous cheating penalties
- You understand it will be re-tested using our cheat detection software

What happens when you ...

- ... “double-click” a program icon?
- ... save a file “foo.txt”?
- ... push a key on the keyboard?
- ... use `malloc`?
- ... execute an assembly instruction?
- ... print a file?
- ... use `printf`?

You will discover the answer to these and more this term!

What is an Operating System?

Generally, an OS is a system that:



- manages resources
- creates execution environments
- loads programs
- provides common services and utilities

Operating Systems

- originated 1951, 'LEO I' from J. Lyons and Co.
- started as simple I/O libraries, batch processors



Application View: what services does it provide?

System View: what problems does it solve?

Implementation View: how is it built?

An operating system is part cop, part facilitator.

The OS provides an execution environment for running programs. The execution environment:

- provides a program with the **resources** that it needs to run, and
- provides **interfaces** through which a program can use networks, storage, I/O devices, and other system hardware components. Interfaces provide a simplified, abstract view of hardware to application programs.
- **isolates** running programs from one another and prevents undesirable interactions among them.

The OS:

- **manages the hardware resources** of a computer system.
Resources include processors, memory, disks and other storage devices, network interfaces, I/O devices such as keyboards, mice and monitors, etc.
- **allocates resources** among running programs.
- **controls the sharing of resources** among programs.

The OS itself also uses resources, which it must share with application programs.

The OS is a **concurrent, real-time** program.

- **Concurrency**, multiple programs/instructions running or appearing to run at the same time. Concurrency arises naturally in an OS when it supports concurrent applications.
- **Real-time**, programs that **must** respond to events within specific timing constraints. For example, hardware interactions impose timing constraints.

How does the OS implement these?

kernel: The operating system kernel is the part of the operating system that responds to system calls, interrupts and exceptions.

operating system: The operating system as a whole includes the kernel, and may include other related programs that provide services for applications. This may include things like:

- utility programs
 - task managers
 - disk defragmenting tools
- command interpreters
 - cmd.exe
 - bash
- programming libraries
 - POSIX
 - OpenGL

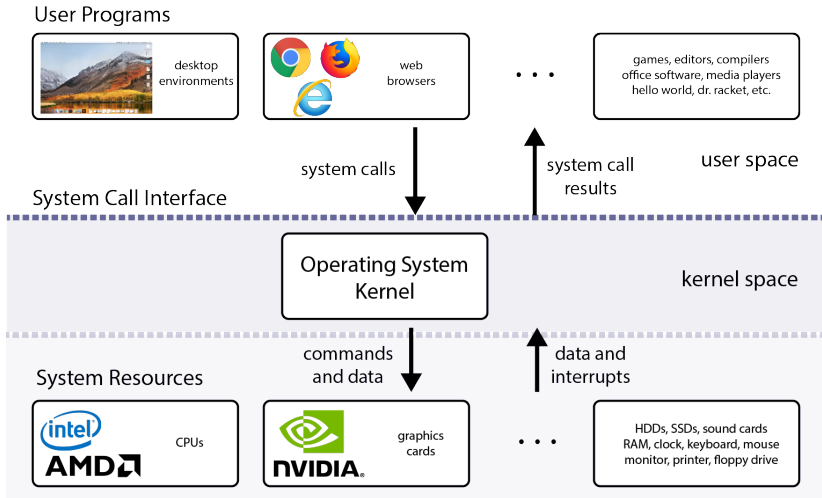
monolithic kernel: "everything and the kitchen sink" is a part of the kernel. This includes device drivers, file system, virtual memory, IPC, etc.

microkernel: only absolutely necessary components are a part of the kernel. All other elements are user programs.

real-time OS: an OS with stringent event response times, guarantees, and preemptive scheduling.

Windows, Linux, Mac OSX, Android and iOS are monolithic operating systems. They are **not** real-time. QNX is a real-time, micro-kernel operating system that originated here!

Schematic View of an Operating System



The **execution environment** provided by the OS includes a variety of **abstract entities** that can be manipulated by a running program. Examples of these abstractions:

files and file systems → secondary storage

address spaces → primary memory (RAM)

processes, threads → program execution

sockets, pipes → network or other message channels

This course will cover why and how these abstractions are:

- designed the way they are
- manipulated by application programs
- implemented by the OS

- Introduction
- Threads and Concurrency
- Synchronization
- Processes and the Kernel
- Virtual Memory
- Scheduling
- Devices and Device Management
- File Systems
- Virtual Machines