

Lecture 1 —Introduction, Operating Systems, Security

Jeff Zarnett
jzarnett@uwaterloo.ca

Department of Electrical and Computer Engineering
University of Waterloo

April 23, 2023

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

The source material for the ECE 350 notes and slides is open-sourced via Github.

If you find an error in the notes/slides, or have an improvement, go to
<https://github.com/jzarnett/ece350> 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!

Maybe we can avoid this situation?

**When a professor makes an
essay due at the end of the
semester and then gets mad at
you for starting it at the end of
the semester.**



Hold on.
This whole operation was your idea.

Thanks to Prof. Zahedi for this section!

Groups are always 4 members; not 5...
3 requires justification.

Group split-up is possible but discouraged.

Why Start Early?

Estimation and planning are hard.
New things pop up!

You don't want to be in this situation at the end of the term.



To complete the labs effectively, work needs to be divided up.



Overhead increases as you have more coordination.

Options for how to divide it up:

- Functional: by what you implement
- Task: Design, Implementation, Test...
- Time-based: Who has time to work on this right now?
- Maybe someone is your product (project) manager?

More people means more communication and more overhead.

Err on the side of over-communication!

Otherwise misunderstandings are possible: index starts at 0 or 1?

Write decisions and important information down!

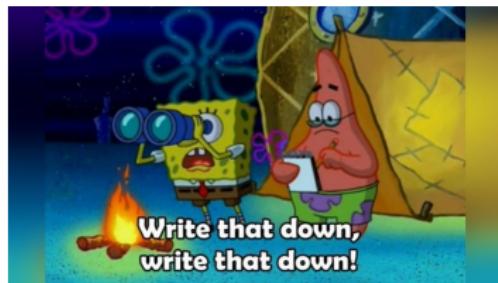
Individual decisions are fast, but group decisions may be best.



Let those who know the most about the problem/area have the final call!

Is someone your system architect? Big picture view.

Did I mention you should write down decisions?



When a decision is made, even if you disagree, go with it.

It's important that you choose group members with compatible work styles.

When I am consultant for a capstone group I ask if they agree on:

- Ahead-of-the-game or last-minute?
- Have you worked together before?
- How far do you want to take this?

Some general advice:

- Communication is key; write stuff down.
- Planning is valuable even if things don't go to plan
- Don't stay mad if someone makes a mistake or misses a meeting
- Put ego to the side
- Use good coding practices: git, automated tests, code review, etc.

Test all along the way, not just at the end.

Unit tests and integration tests.

Also try messing with the timing to ensure it doesn't break things!

The labs are not easy, but they are a very valuable learning experience.



We choose to go to the moon in this decade and do the other things. Not because they are easy, but because they are hard.

— John F. Kennedy —

AZ QUOTES

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

How likely are you to recommend Windows 10 to a friend or colleague?

1

2

3

4

5

Not at all likely

Extremely likely

Please explain why you gave this score.

I need you to understand that people don't have conversations where they randomly recommend operating systems to one another

THIS IS NOT
TRUE

have you ever seen a linux user

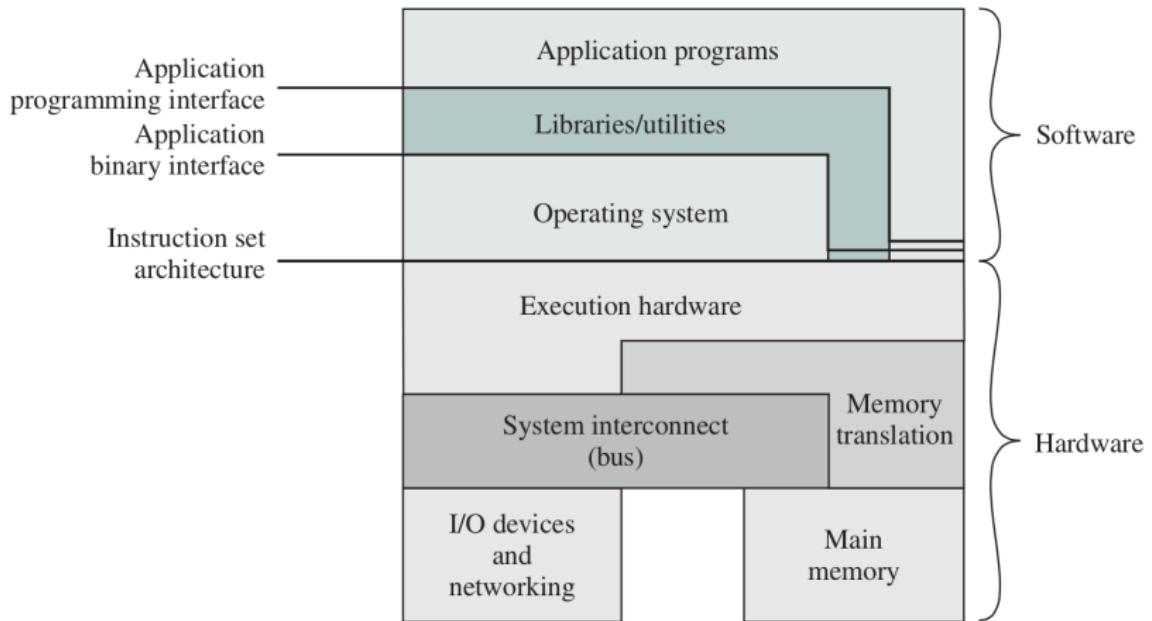
Introduction to Operating Systems

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

It has many goals, that often conflict with one another.

Its job is to make it so other programs can run efficiently.

Structural Diagram of a Modern Computer



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.

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.

Multiple programs means some resources are shared.

→ 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.

Another goal may be to use the computer hardware efficiently.



Image Credit: Argonne National Laboratory

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

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.

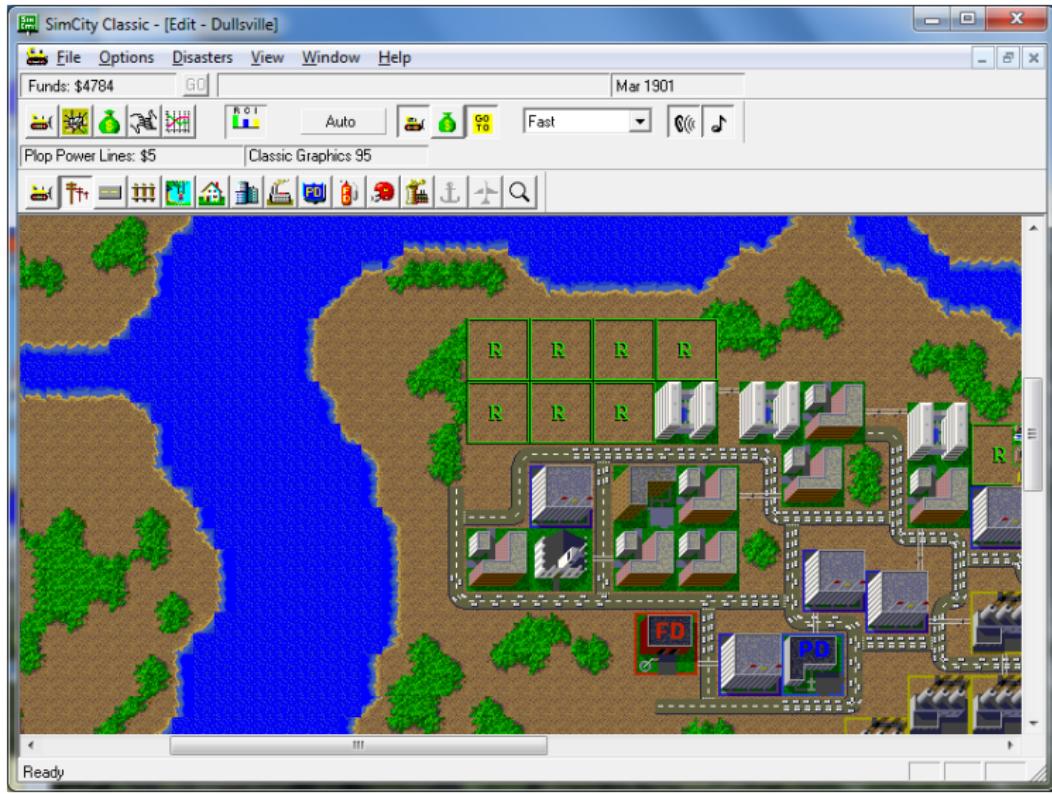
Operating systems will evolve over time.

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.



Example: SimCity and Windows 95





People assume that time is a strict progression from cause to effect, but actually from a non-linear, non-subjective viewpoint, it's more like a big ball of wibbly-wobbly, timey-wimey stuff.

Real-Time vs. Non Real-Time

Real-time systems are the ones where wall-clock deadlines matter.

Examples: aviation, industrial machinery, video conferencing, satellites...

There are deadlines, and there are consequences for missing deadlines.



Fast is not as important as predictable.

Hard real-time: it has a deadline that must be met to prevent an error, prevent some damage to the system, or for the answer to make sense.

If a task is attempting to calculate the position of an incoming missile, a late answer is no good.

A **soft real-time** task has a deadline that is not, strictly speaking, mandatory; missing the deadline degrades the quality of the response, but it is not useless.



In many textbooks, security and protection are left to the end.

Security has to be designed from the beginning; cannot be bolted on.

An OS supports multiple users with multiple tasks.

OS designers create policies and policy tools.

There is a tradeoff with usability.

You do NOT want to have to report a data breach.



The OS must enforce the configured policies.
Otherwise, malicious users exploit it.

Three desirable properties: Confidentiality, Integrity, Availability.

Protection is about internal threats.

Security is about external threats.

We'll take them in order.

PROTECTION

3rd-level abjuration

Casting Time: 1 action

Range: 30 feet

Components: V, S, M (a miniature silver shield worth 10gp)

Duration: Concentration, up to 1 hour

Class: Bard, Cleric, Druid, Paladin

You trace a warding sigil in the air, sending it to slowly circle a target you can see within range for the next hour. When the target next takes damage, the ward triggers, granting resistance to all damage types except psychic until the end of the target's next turn.

At Higher Levels. When you cast this spell with a spell slot of 4th level or higher, you may target one additional creature for each spell slot level above 3rd.

Design by Humperdink's Wares. "Protection" is unofficial Fan Content permitted under the Fan Content Policy. Not approved/endorsed by Wizards. Portions of the materials used are property of Wizards of the Coast. © Wizards of the Coast LLC.

Art Credit: TES:L



Most of the discussion in this course is about protection.

Following the rules is important to a functioning system or society.

Even in the absence of malicious intent.

Enforce policies about responsible usage.

Responsible and reasonable vary across systems.

Examples of access control: file permissions, walls between processes.

Rules take effort to enforce.

Exceptions are allowed and administrators can override policies!



Other kinds of rules can exist.

If I edit a lecture video, it consumes a lot of CPU and RAM.

Is that bad?

It's legitimate – but might set off alarms.

So we may be lax – which gives an opening to malicious users...



Let's consider a few bad things attackers can do.

Why? We need this in mind when designing the system.

- Breach of Confidentiality
- Breach of Integrity
- Breach of Availability
- Theft of Service
- ... and others.

Most likely the weak link is people.

A few potential specific attacks:

- Excessive Requests
- Malformed Requests
- Back Door
- Intercepting Messages
- Trojan Horse

Security is an arms race.

Can we break things if it's a security problem?

No easy answers.

This is not a course in security, but we can't ignore it.

Security needs to be included from the beginning!

A broad overview of the major topics of the course:

- 1 Introduction, Security, Review from ECE 252
- 2 Memory
- 3 Scheduling
- 4 I/O Devices, File Systems
- 5 Reliability, Virtualization/Containers