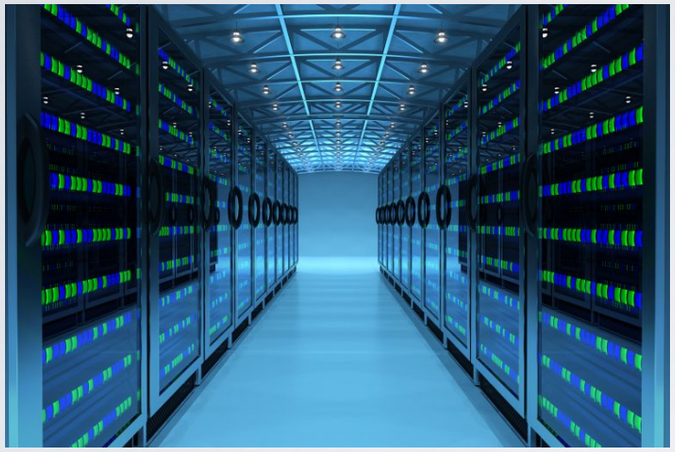
# CSCC69 Operating Systems

Thierry Sans

# Why do we need operating systems?






# Why learning about operating systems

- An exciting time for building operating systems
  New hardware, smart devices, self-driving cars, data centers, etc.
  Facing OS issues in performance, battery life, security, isolation

- Pervasive principles for systems in general
  Caching, concurrency, memory management, I/O, protection

- Understand what you use
  System software tends to be mysterious
  Understanding OS makes you a more effective programmer

- Complex software systems
  Many of you will go on to work on large software projects
  OSes serve as examples of an evolution of complex systems

# CSCC69

- An introductory course on Operating Systems' design principles
- A hands-on experience building an OS

➡ Theory and practice goes hand-in-hand

# New trends in OS
# same same but different

Different architectures but, in the end, the same concepts defined in the 70s

But there are some new trends

- Multicore
- Energy efficiency (mobile and IoT devices)
- Virtualization (cloud computing)

# New version of the course

- Better reflect on industry current standards and the newest trends

- A very hands-on approach to better understand x86 Operating Systems

# This is a tough course ...

- Lots of things to read, materials to digest before being able to produce something that looks like the solution
- Half-way through implementing your solution, you will need to refactor it
- In the end, it will be one of the most challenging and significant piece of code that you have ever written

... but you will gain valuable learning and experience

# Course Work

# Pintos

- Developed in 2005 for Stanford's CS 140 OS class and used by many universities since then
- Written in C, built for x86 hardware

✓ Can run on a real machine!

# Project Setup

Execute concurrent programs

- that run on Pintos
- that runs on Bochs/Qemu emulators
- that run on Linux
- that runs inside a Docker container
- that runs on your OS
- that runs on your personal computer

*"Turtle all the way down" - Wikipedia*

# 4 Projects

| | | | |
|---|---|---|---|
| Project 1 | **Threads** | Individual/Group | challenging |
| Project 2 | **User Programs** | Individual/Group | challenging |
| Project 3 | **Virtual Memory** | Individual/Group | very challenging |
| Project 4 | **File System** | Individual/Group | very challenging |

# Team spirit ... or not

"*Alone we can do so little; together we can do so much.*"
– Helen Keller

"*Coming together is a beginning; keeping together is progress; working together is success.*"
– Henry Ford

"*You rise as a team, you die as a team*"
– Me

# Projects deliverables and grading

1. Automated tests
   - All tests are given so you immediately know how well your solution performs
   - You either pass a test case or fail, there is no partial credit
2. Design document
   - Answer important questions related to your design for a lab
3. Coding style
   - Code must be easy to read and follow coding style guidelines
   - The TA will conduct a code review

# The red line between collaboration and plagiarism

✓ Collaboration is allowed

- you can explain a concept to someone in another group
- you can discuss algorithms/testing strategies with other groups

◉ Plagiarism is not allowed

- you cannot discuss specific implementation details
- you cannot look at other people's solutions, including solutions online (e.g., GitHub)
- you cannot publish your own solution online (even after the course term)

✦ Looking for materials is tolerated

- you can look for snippets of code online as long as this piece of code does not directly answer your a specific problem of the assignment
- if you copy more than 5 lines of code, put the source url as a comment in your code
- if you use more than 25 lines, do not copy it

# Late policies

Each team will have **4 days grace period** that can spread into 4 projects for interview, attending conference, errands, and so on, no questions asked

➡ **use it wisely**

# Let's look at the syllabus

- https://thierrysans.me/CSCC69/

# How to succeed in the course

- Stay on top of lecture materials and readings
- Start working on projects from day 1

# Acknowledgments

Some of the course materials and projects are from

- Ryan Huang - teaching CS 318 at *John Hopkins University*
- Ben Pfaff - creator of Pintos
  and researcher at *VMware Research Group*
- David Mazière - teaching CS 140 at *Stanford*