

# Introduction

## ICS412 - Fall 2009 Operating Systems

Henri Casanova (henric@hawaii.edu)

## Course Goals

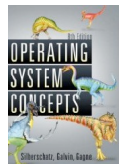
- Gain fundamental understanding of OS principles (kernel, processes, threads, memory management, file system, etc.)
- Understand some of the key features/differences of Linux and Windows
- Learn how to program using system calls
- Learn how to implement/modify part of an OS
  
- This is very much a “hand on” class
  - we'll look at code
  - we'll write code

## Course Website

- Located at:
  - [http://navet.ics.hawaii.edu/~casanova/courses/ics412\\_fall09](http://navet.ics.hawaii.edu/~casanova/courses/ics412_fall09)
  - Linked from my homepage
    - Accessible from the ICS homepage
    - Google “Henri Casanova”
- Contains
  - All lecture notes
  - Pointers to useful on-line material
  - All assignments
  - Announcements
  - A link to the PDF Syllabus
    - the content of which is summarized in these notes

## Textbook

- Operating Systems Concepts, *8th Edition*, by Silberschatz, Galvin, and Gagne
- Most of the lectures will be very tightly connected to particular book chapters
- There will be reading assignments
  - In the interest of time, I will at times say things like “this is explained in the book” and move on, with the understanding that you will go read the book (and possibly come back with questions)
- When to do the reading assignments?
  - Some may prefer to read them before the corresponding lectures
  - Some may prefer to read them after
  - It's up to you...
- We will have homework questions out of the book as well



## Lectures and Office Hours

- Lectures: Tue & Thu, 3PM-4:15PM
- Office Hours: Wed, 1PM-3PM
  - POST 310C
  - phone: 956-2649
  - e-mail: [henric@hawaii.edu](mailto:henric@hawaii.edu)
  
- Teaching Assistant: None

## Assignments

- Homework Assignments
  - “Pencil and Paper” assignments that require thought, but no programming
  - (turning in electronic PDF assignments rather than oddly hand-scribbled ones is appreciated)
- Programming Assignments
  - Write code
    - Either using the OS or modifying a “toy” OS
  - Report on code's behavior
  - Code written in:
    - C or C++ (up to you)
    - Java
    - Some doable on Windows, but mostly on Linux
  - Some individual, some in teams

## Assignments Due Dates

- Assignments are due at 11:59PM on the due date
- **Late assignments:** 20% grade penalty for each additional day of lateness
  - e.g., if the assignment is due the Tuesday at 11:59PM, assignments turned in between 0:00AM and 11:59PM Wednesday will get a 20% penalty, assignments turned in between 0:00AM and 11:59PM Thursday will get a 40% penalty. etc.

## Exams, Grading

- Exams
  - One midterm exam
  - One comprehensive final exam
- Grading
  - Homework Assignments 20%
  - Programming Assignments 30%
  - Midterm 20%
  - Final 30%

## How not to Pass 412?

- Do not come to lectures
  - It's nice out, slides are on-line, there is a textbook
  - Many things are said in class that directly relate to assignments and exams
- Do not do the homework assignments
  - It's only 20% of the grade
  - Homework is the best practice for the exams
  - 20% takes you from an A to a C

## How not to Pass 412?

- Do not ask questions in class, office hours, or e-mail
  - It's too scary, the instructor speaks with a funny accent
  - Asking questions is the way to get the most out of lectures
  - **Well thought out questions** (especially via e-mail) will avoid your wasting time on assignments
- Start late on the programming assignments
  - I work better under pressure, and the later I start the less time I'll spend on it
  - For 99% of the students, assignments are just not doable in the last few days, under pressure
  - I will not answer "I wrote some code but it doesn't work" or "I don't understand <concept taught in class>" questions on the assignment's due day

## Questions about the Syllabus?

## Course Content

- This course focuses on **OS Principles**
  - Structure of the OS
  - Services provided by the OS
  - OS implementation on modern hardware
  - Techniques for implementing complex software systems and make them evolve
- System Software is often seen a very mysterious
- Our goal is to reveal most mysteries

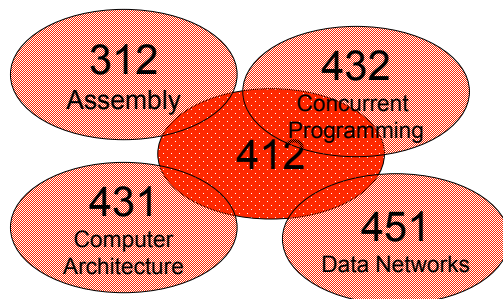
## Fundamental OS Issues

- **Structure**: how is an OS organized
  - **Sharing**: how are computer resources shared among users
  - **Naming**: how are computer resources named/referenced by users and programs
  - **Protection**: how are users protected from each other
  - **Security**: how can information flow be restricted
  - **Communication**: how can programs and computers exchange data
  - **Reliability and fault-tolerance**: how to mask failures
  - **Extensibility/Compatibility**: how to add features
  - **Concurrency**: how to control activities
  - **Performance**: how to do all this fast
- The “principles” in this course are the design methods, implementation approaches, and current solutions to the above issues

## Specific Topics in this Course

- What is an OS, how does one design on?
- Processes, Threads, Synchronization, Communication, and Scheduling
- Memory Management
- File Systems
- Virtual Machine Technology
- Linux, Windows, and references to others
  - Oses are extremely diverse
- and perhaps a little bit on Distributed Systems

## ICS412 and the ICS Curriculum



## Software/Hardware for ICS412

- We will use freely available software in the class
- You can install the software on your own (virtualized) Linux machine
- If you really need to, you can get an account on a server on which you can do all your work
  - connecting via Ssh
- Details forthcoming...

## Disclaimer

- I am not an “OS geek” and questions like “How has the CPU scheduler been modified between Linux kernel 2.2.19 and 2.2.24 in terms of its handling of process priorities on an SMP?” will likely not be answered in class right away
  - But hopefully the following lecture, if the question is interesting
- Typically, in a class like this, there could be a few students who, e.g., have read the entire source code of some open-source OS
  - Something that professors typically never do
- If you have tons of useful knowledge about particular Oses, please share with the class
- If you catch me saying something that you think is not right, let us know right away and start a discussion
- However, the class is more on general principles than specific implementations (since those change constantly)

## A few questions

- Who has never written programs in C before?
- Who does not feel comfortable in C?
- Who has never programmed in a Linux environment before?
- Who has never written a Makefile?
- Who thinks that a short Linux/UNIX refresher would be useful?



Any Questions?