ECS 150 - Course Introduction

Prof. Joël Porquet-Lupine

UC Davis - 2020/2021

UCDAVIS

COMPUTER SCIENCE

Who am I?

Current

- 2018-present: **Assistant Professor of Teaching**, UC Davis
- 2017-18: **Lecturer**, UC Davis

Previously

At first, mostly hardware-oriented with some OS aspects:

- 2010: **PhD** at Sorbonne University, Paris, France
  (Co-sponsored by STMicroelectronics)
  Security architecture for multiprocessor Systems-on-Chip

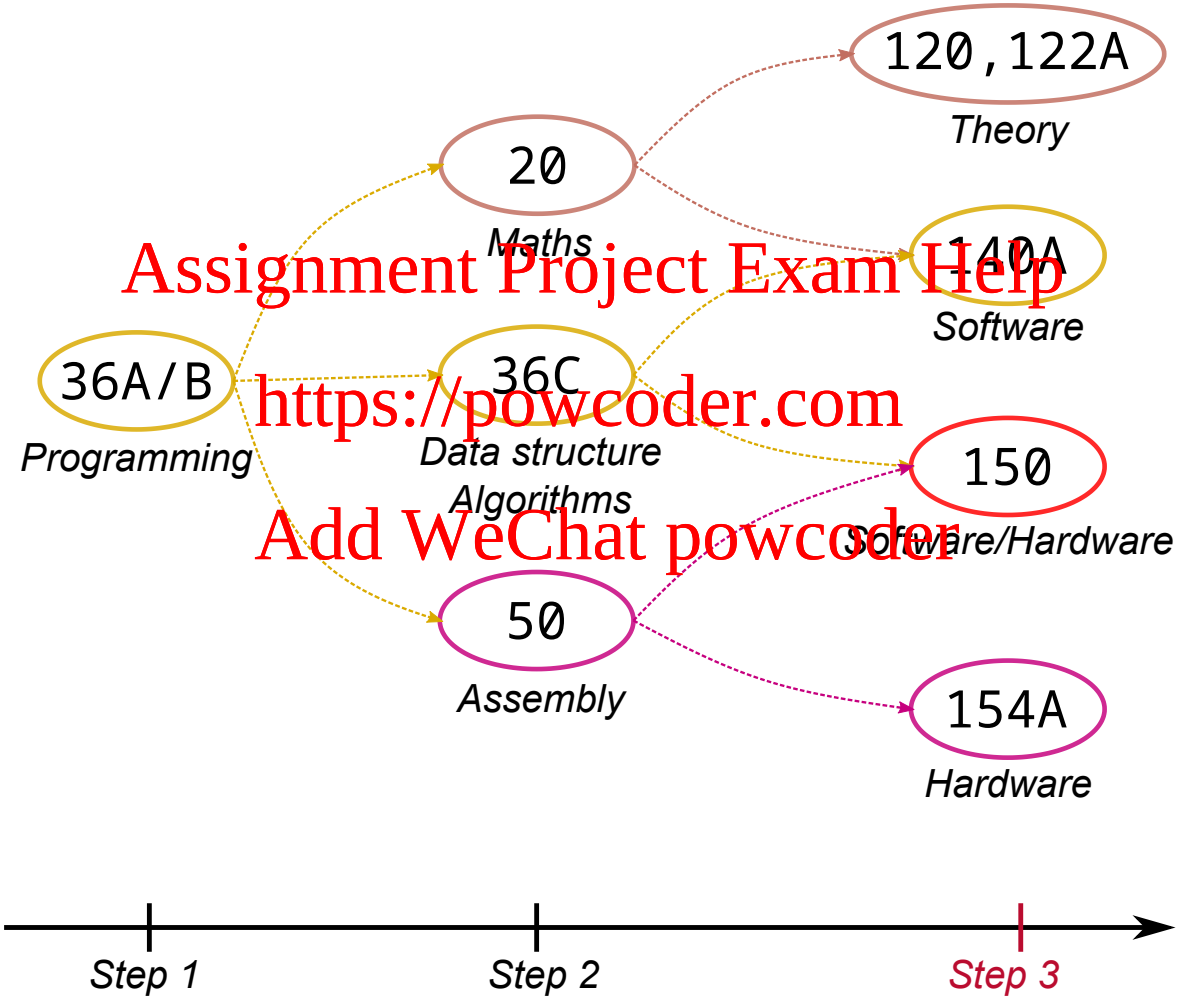
- 2011-12: **Postdoc** at Columbia University, NYC, NY
  Security mechanisms for heterogeneous multiprocessor Systems-on-Chip

Then, mostly OS-oriented with some hardware aspects:

- 2013-15: **Postdoc** at Sorbonne University, Paris, France
  Porting the Linux kernel to a manycore processor architecture
- 2015-16: **Team leader** of the Firmware Development Kit for Google's modular smartphone *Project ARA* (discontinued)

Why ECS 150?

Important steps in CS (undergrad) education



Why ECS 150?

An OS for every kind of computer system

Laptops, cellphones, microwaves, washers, dryers, dishwashers, coffee makers, refrigerators, televisions, game consoles, cars, planes, ...

Even in toothbrushes!
(runs RTOS Zephyr)

Ever-evolving applications and technologies

- Self-driving cars, IoT (*Internet of Things*), smart homes/cities
- New processor architectures, memory topologies, networking technologies

Professional skills

- **Some** of you will actually design and build OSes or components of them
- **Many** of you will create systems that use the core concepts of OSes
- **All** of you will build applications that use OSes!

Why ECS 150?

Concrete objectives

Theory

- Operating systems: definition and structure
  - Syscall API, Kernel, Processes and threads
- Concurrency between processes/threads
  - Race conditions, synchronization, deadlocks
- Storage
  - Hard-drive/SSD, Filesystems
- Memory management and virtual memory
  - Memory segmentation, paged memory, on-demand paging

Practice

- Intensive programming!
  - Three projects

Course structure

	Monday	Tuesday	Wednesday	Thursday	Friday
Week #1	Jan 4, 2021 L1.1	5	6 L1.2	7	8 L1.3 D1
Week #2	11 L2.1	12	13 L2.2	14	15 L2.3 D2 *Drop-deadline*
Week #3	18 *Holiday*	19	20 L3.2	21	22 L3.3 D3

Theory

Lecture

- Focused on concepts
  - Supported by slides
  - Code examples and demo
- Asynchronous

Discussion

- Projects
  - Synchronous
- Tutorials (e.g., Makefile, GDB/Valgrind) or other OS-related topics
  - Asynchronous

Course structure

Practice

Programming projects

- 3 projects to cover the main class topics
  - P1: Simple shell
  - P2: User-thread library + synchronization
  - P3: Custom file system
- Fairly intense both conceptually and in terms of programming
- To be worked **in pairs**, in order to develop your collaboration skills
  - Choose your partner well!

Group work

- Self- and peer-evaluation
- Individualize each partner project score

Communication

- Interactive grading during the last week of class
- Very important training for job interviews, work meetings, etc.

Course structure

Assessment

Theory (50%)

- Class quizzes
  - To keep you on track with lecture/discussion material
  - 3 lowest scores are dropped
- Two midterms
  - Comprehensive understanding
  - Timed, open-book

Practice (50%)

- Three projects
  - Graded both on correctness *and* quality of implementation
- Group work evaluations
  - Individual contribution to projects
- Interactive grading
  - Graded on ability to explain your code, design logic, and project concepts

Conclusion

Disclaimer

This class is difficult!

> *In the end, you'll become more frustrated than ever, but you'll also learn more than you've ever learned in one class.*
>
> *TL;DR There's no more hand holding.*

Quote from student (WQ17)

But with hard work, it's possible to do well!

Grade	WQ17	SQ17	WQ18	SQ18	WQ19	SQ19	FQ19	WQ20	FQ20
A	22%	17%	24%	22%	24%	34%	28%	28%	22%
B	48%	46%	50%	46%	45%	39%	44%	34%	39%
C	25%	32%	22%	27%	18%	15%	21%	26%	26%
D	5%	4 %	4 %	3 %	7 %	6.5 %	3%	7%	10%
F	0%	1 %	0 %	2 %	5 %	5.5 %	3%	5%	3%