INFORMATION & COMPUTER SCIENCES
UNIVERSITY of HAWAI'I at MĀNOA

# Course Introduction

## ICS 332: Operating Systems

KYLE BERNEY
DEPARTMENT OF ICS, UNIVERSITY OF HAWAII AT MANOA

Slides adopted (with permission) from Henri Casanova

# Introductions

- Kyle Berney (Instructor)
  - Email: berneyk@hawaii.edu (Put “ICS 332” in the subject line)
  - Office: POST 314C
  - Office Hours: Tuesdays 10:30am - 11:30am (in TBA)
- Christian Moore (Teaching Assistant)
  - Email: moorechr@hawaii.edu (Put “ICS 332” in the subject line)
  - Office Hours: TBA

# Course Goal

- At this point in your life/studies, you:
  - Have used at least one OS
  - Know which OS runs on your computer
  - Know that without an OS, you couldn't use your computer
- Yet, for most of you, the OS is pretty mysterious
- Say your art major friend asks: "What really happens when I double click on an icon to run an application on my computer?"
- Could you give decent answer besides: "Amazingly, it all works?"
- As a Computer Scientist, it's okay to not know much about how a car/fridge/airplane works, but it's not okay to be clueless about how an OS works

# Motivation to Study OSes?

- After all, very few of you will develop an actual OS
- **But**, most of you will work on complex systems that couple together many components (not an ICS 111/211 assignment)
- **Obvious Motivation:** These systems all use OS heavily
  - Important to know how to use the OS as programmer
  - Important to know what the OS can and cannot do
  - Important to know what happens "under the hood" to understand bugs, security, performance, etc.
- **Meta Motivation (you have to trust me on this one):** Knowing OS principles makes you a better software architect and developer
  - OS concepts are massively re-usable in your own projects

# Disclaimer

- Wouldn't it be great for us to develop an OS during the semester?
- Sadly, it's not feasible:
  - Too hard, time-consuming, programming-heavy for most students at this stage in their education, especially students who may struggle with C programming
  - Or at least given the expected number of hours a student should put into an undergraduate course...

# Disclaimer

- As a result, undergraduate OS courses are often less “hands-on” than what some students expect
  - Typically, a few of you come into this course thinking we’re going to do awesome hacks in the Linux kernel... and will be disappointed
- If your dream is to get your hands dirty with the OS kernel, you can:
  - Do an internship at a company that does low level / OS-related work, do an OS-related Google Summer of code, etc.
  - Take the OS graduate course (ICS 612: Theory of Operating Systems)
  - Do it on your own (there are tons of online resources)

# Disclaimer

- I am not an “OS geek”, nor am I an OS expert
- If you’ve read the source code of some OS, worked on an OS (e.g., internships), or have any useful knowledge you are welcome to share with the class when appropriate
- That said, this class is more on general principles than specific implementations (since implementations change often)
  - You can learn a lot of OSes without necessarily spending hours looking at OS code

# Teaching OS is not easy

- A significant part of the material is of the “here is how it works and why it’s a good idea” kind
  - Precisely because it’s not feasible to have the full-fledged hands-on experience at the undergraduate level
  - Don’t fear, there will be plenty of programming / hands-on activities in this course
- Although I will try to make this course as interactive as possible, there is a limit to what any instructor can do for some of this material at the undergraduate level
- Bottom line:
  - The course is fun when students are engaged and ask questions
  - The course is dull when students are silent

# What we will learn

- Roles of an operating system
- Fundamental principles of operating system design and kernel implementation
- Key features of operating systems of practical importance
  - The course content is not specific to a particular OS
  - Many OSes do things in a similar way, but they also have key differences
  - We will often reference Unix derivatives (e.g., Linux, Mac OS, etc.) and Windows
  - We will mention “historical” OSes whenever relevant
  - We will not study special-purpose OSes (e.g., real-time network operating systems)

# What we will learn

- Fundamental components and principles of operating systems:
  - Processes and Threads Management Scheduling
  - Synchronization (barely scratching the surface here)
  - Memory and Virtual Memory Management
  - Storage and, if time permits, File Systems
- We will have several programming assignments that are more about “using” the OS than about “implementing” the OS
  - In other words, what most of you will need in your profession

# Course Website

- Located at:
  - https://courses.ics.hawaii.edu/ics332_f25/
- Website is organized as modules
  - All lecture notes as PDF files
  - Pointers/links to useful online material
  - All assignments
  - Syllabus
    - Which we're going over now in these slides
- Let's take a look at the website...

# Textbook

- Operating Systems: Three Easy Pieces
  - A.k.a. OSTEP
  - By Arpaci-Dusseau, R.H. and Arpaci-Dusseau, A. C.
  - Freely available at https://pages.cs.wisc.edu/~remzi/OSTEP/
- Lectures are tightly connected to particular chapters
- There will be reading assignments from this textbook, as indicated on the lecture notes
- Some exam questions and assignments will be directly from or inspired by the textbook
- There are several classic texts for Operating Systems (shown in the “syllabus” page on the course website)

# Course Content

- In spite of my best efforts, it happens that the course website could have small problems (e.g., typos, missing links, etc.)

- Anytime you see anything strange/broken on the website, please let me know

# Grading

- Two exams (40%)
  - One Midterm (20%)
  - One Final (20%)
- Quizzes (10%)
  - Roughly every week
  - Announced beforehand
- Homework assignments (50%)
- Read the syllabus' statement about "academic dishonesty"

# Homework Assignments

- All assignments must be turned in electronically via Lamaku by 11:55pm HST on the day the assignment is due
  - **Scanned hand-written assignments are not allowed**
- Late Assignments
  - 10% penalty for up to 24 hours of lateness
  - A grade of 0 for more than 24 hours of lateness
  - E.g., if the due date is 10/10, an assignment turned in at 1am on 10/11 will be penalized by 10%, and given a zero if turned in at 5pm on 10/12

# Homework Assignments

- If Lamaku is down, email your submission to both myself and the TA immediately
  - Do not send an email that says “Lamaku is down, what should I do?”, which we’ll only see the day after
- After submitting, double-check what you submitted!
  - “Oops, I submitted and empty file... here is what I really meant to submit yesterday” will not be accepted

# Homework Assignments

- All assignments are individual (not group) assignments
- Some assignments will be pencil-and-paper that require no programming
  - But may require the use of a Linux box to observe things
- Some assignments are programming assignments
  - Write code and/or report on code execution
- Pencil-and-paper and programming assignments can overlap in time

# Homework Assignments

- Instructor and TA will not answer assignment related emails on the day the assignment is due!
  - Start early
  - Come to office hours for questions

# Progamming Assignments

- Java 8 or later, some C, some scripting
- Some programming assignments can be implemented on Windows, Mac OS, or Linux
- Other programming assignments will require Linux
- You can use your own machine (or a machine in a lab) for the assignments, using whatever text editor or IDE you prefer
- BUT, you must test/run your code on a Linux machine (real or virtual)
  - Because that's what we'll use to for testing/grading
  - One common source of errors is Windows ("/", ";") vs. Linux ("/", ":")

# Progamming Assignments

- Each programming assignment has specifications and examples
  - Command-line arguments, file names, etc.
  - If command-line arguments are not correct, then your programs have to exit gracefully
  - If you find specifications unclear, let us know
- Nonconforming with these specifications makes us less tolerant (and sometimes intolerant) in terms of grading
  - We will not go into your code to fix it
  - You will lose points for not following the specifications
- Following specifications and writing robust code will be a HUGE part of your professional lives

# The Linux CLI (Shell)

- Knowledge of the UNIX/Linux CLI (Shell) is needed in this course
- Show of hands: who feels somewhat familiar with the UNIX/Linux CLI?
- A huge potential side-benefit of taking this course is to become decent/good with the command line
  - About 25% of students passing this course tell me that they are forever grateful that they forced themselves to learn/use the Shell
  - This is also something who hear from alumni who, once in a job, “discover” that 99% of back-end systems are Linux-based and use the Shell daily
- This module (Getting Started) contains some pointers

# On the Importance of Terminology

- As we learn about Operating Systems, we will encounter a lot of terminology
- Recognizing and using the correct terminology is part of what we learn in this course
- Knowing the terminology is very important (e.g., in a job interview or in a professional context)
- So, in class, whenever a student asks a question, I may rephrase the question using proper terminology
  - This is not to be annoying or belittling, it's to make sure we all come out of this course speaking the language of Operating Systems (and of Computer Science)
- Of course, terminology will be part of the quizzes/exams

# How to not do well in this course?

- **Don't come to class ("the slides are nice")**
  - We do a LOT of stuff in class, including coding, additional explanations, and examples
- **Start assignments late ("I work better under pressure")**
  - OSes are a difficult topic
  - Starting late seems to be a growing trend, and it's a problem
  - Read the assignment early to subconsciously start thinking about it

# How to not do well in this course?

- **Don't turn in assignments**
  - Every semester, some students do not turn in assignments and then seem surprised that they fail
  - Just count your points to know what your current grade is
- **Don't come to office hours ("the instructor is scary because he shows 'how to not do well in this course' slides")**
  - After you struggle for a while on something, drop by
  - Don't expect to "camp" in the office hours for the solutions to be given out
  - Instructor and TA office hours are an amazing service provided to you, and yet, they go mostly unused

# How to not do well in this course?

- **Cheat**
  - Almost every semester students are caught cheating
    - Cheating is bad for many reasons, including hurting the reputation of ICS graduates
    - This is part of the reason for 50% of the course's points being exams
  - If you are caught cheating or enabling cheating:
    - Zero on the assignment/exam
    - Overall grade lowered by a letter grade (e.g., a "B" becomes a "C")
    - Reported to UH's Office of Judicial Affairs (as required)
- **Expect that "what can I do for extra credit?" will be met with a positive response... it won't**

# How to not do well in this course?

- **Expect that "what can I do for extra credit?" will be met with a positive response... it won't**
- **Don't study for quizzes**
  - "It's only 10% of the grade"
  - Studying for quizzes is a HUGE help to prepare for exams
  - When I don't do quizzes, the average grade drops!

# Questions?

- Any questions on the syllabus?
- Any questions on the course in general?

# Conclusion

- Let's take a look at (ungraded) Homework #0