Welcome and Introduction

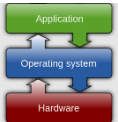
Lecture 1

Hartmut Kaiser

https://teaching.hkaiser.org/fall2025/csc7103/

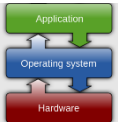
Introduction to Operating Systems

- Administrative introduction to course
- Why study Operating Systems?
- What is an Operating System?
- Principles to be covered in this course
- A (very) brief history of Operating Systems
- Operating System Goals



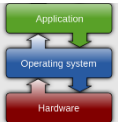
This Course

- This is a revised curriculum with new goals:
  - Understanding and exploiting OS services
  - Foundation concepts and principles
  - Common problems that have been solved in OS
  - Evolving directions in system architecture
- This is not a course in how to build an OS
  - You will not read or write any kernel-mode code
  - You will not study or build any parts of a toy OS



Learning Objectives

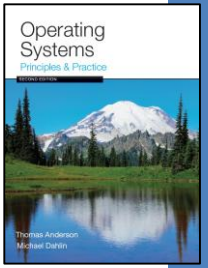
- We started with a list of learning objectives
  - Over 300 concepts, issues, approaches and skills
  - All activities in this course are based on them
- The reading has been chosen introduce them
  - The lectures are designed to reinforce them
  - The projects have been chosen to exercise them
  - The exams will test your mastery of them
- Study this list to understand the course goals
- Use this list to guide your pre-exam review

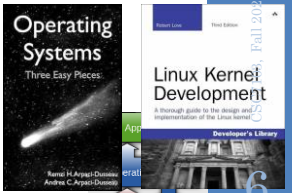


Administrativia

Infrastructure, Textbook & Readings

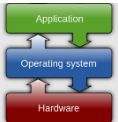
- Infrastructure
  - Website: https://teaching.hkaiser.org
  - Discord: https://discord.gg/FIXME
- Textbook: Operating Systems: Principles and Practice (2nd Edition) Anderson and Dahlin
  - Not required, get a copy if you feel that lectures are not sufficient
  - Suggested readings posted along with lectures
  - Try to keep up with material in book as well as lectures
- Supplementary Material
  - Operating Systems: Three Easy Pieces, by Remzi and Andrea Arpaci-Dusseau, available for free online
  - Linux Kernel Development, 3rd edition, by Robert Love
- Homework, project, quizzes, grading, honesty
  - Use Github classroom for assignments
  - Use VSCode as a tool





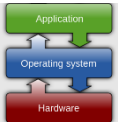
Class Expectations

- Lectures
  - Come! Cannot guarantee content will be identical to previous years
  - Electronic devices used only for note taking
  - Will explain many things related to assignments not otherwise explained
  - Attendance not mandatory but highly advised
  - If you decide to come, please be on time
  - Meet your fellow students, they are your future colleagues!
- Office Hours
  - Come and ask for help early. There are no stupid questions!
  - We like teaching and want to meet you!
- Communicate with course staff through Discord and Email.



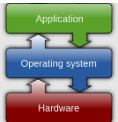
Important Dates

- Lectures
  - Tuesday and Thursday, 1:30pm to 2:50pm, 1236 PFT
- Grading
  - Homework 50%
  - Project 25%
  - Midterm exam 10%
  - Final exam 15%
- Exams
  - Midterm exam: TBD
  - Final exam: TBD

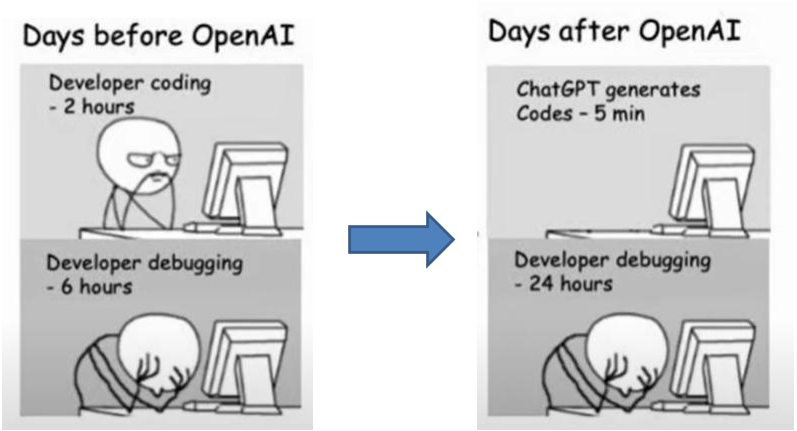


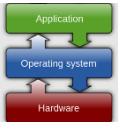
Honesty

- The LSU *Code of Student Conduct* defines plagiarism in Section 5.1.16:
  - "Plagiarism is defined as the unacknowledged inclusion of someone else's words, structure, ideas, or data. When a student submits work as his/her own that includes the words, structure, ideas, or data of others, the source of this information must be acknowledged through complete, accurate, and specific references, and, if verbatim statements are included, through quotation marks as well. Failure to identify any source (including interviews, surveys, etc.), published in any medium (including on the internet) or unpublished, from which words, structure, ideas, or data have been taken, constitutes plagiarism;"
- Plagiarism will not be tolerated and will be dealt with in accordance with and as outlined by the LSU Code of Student Conduct: https://www.lsu.edu/saa/students/codeofconduct.php



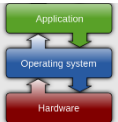
ChatGPT?


Days before OpenAI
Developer coding
- 2 hours
Developer debugging
- 6 hours
Days after OpenAI
ChatGPT generates
Codes - 5 min
Developer debugging
- 24 hours


Application
Operating system
Hardware

ChatGPT

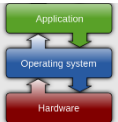
- The goal for this course is to train our brains, not ChatGPT
  - So, do yourself a favor and don't use it
  - There are too many software 'developers' out there who copy & paste their way to the next paycheck
- However, if you do use it:
  - Never use anything without carefully reviewing it
    - Assume what you got is wrong! Prove to yourself it is correct!
  - The skill of reading (and understanding) code becomes more important than ever
  - Cite and annotate the copied code
- Whatever you do, remember:
  - It's plagiarism if you submit the same code as your neighbor
  - No matter where you got it from, be it Google, ChatGPT, or your neighbors computer



How to Learn

- Ask yourself at the beginning of this semester
  - What scientific problem you want to solve?
  - What do you want to learn from this course that can prepare you?
  - Can you write a sequential program to solve it?
  - What is the performance of it?
  - …

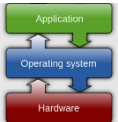
- Ask yourself at the end of this semester
  - Did you master the skillset to solve your scientific problem?
  - Can you write a high-performance program to solve it?
  - What is the performance of it?
  - What is the speedup?
    - Compare your sequential program with your high-performance program



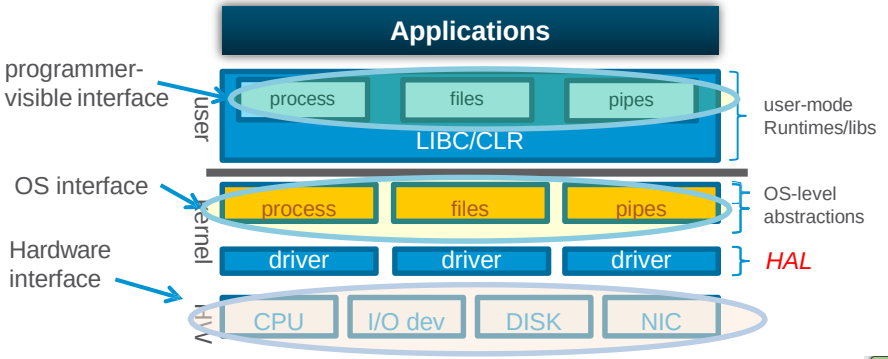
OS Research

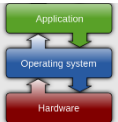
OS Research Actually

- Not really just about building Oses
- Any large code base converges on becoming an OS
  - Manages memory, programming for space/performance, hardware details
  - database, JVM, browser, parallel/distributed systems, internet services
- How to structure systems and deal with complexity
  - Modularize and encapsulate
  - Choose interfaces/abstractions carefully
- Themes in this class
  - Abstractions for managing/accessing resources: storage, replication, naming
  - Correctness: concurrency and sharing
  - Guarantees in real systems: security, fault tolerance, etc.

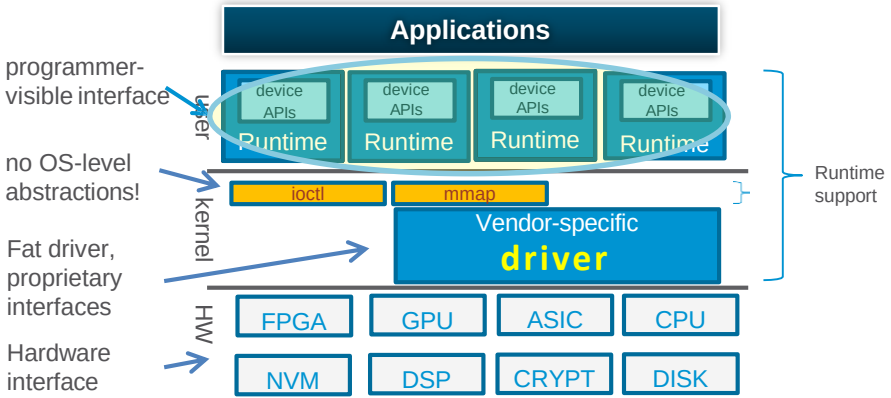


How to “structure” a System

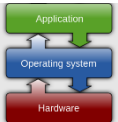




How to “structure” a System



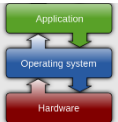
- OSes designed for old hardware
  - Old OS assumptions: CPUs, disks, networks, OS is the only resource manager



Today's Readings

Today's Readings: Motivation & Professional Development

- Levin & Redell: "How (and How Not) to write a Good Systems Paper"
  - Summary:
    - Rules of thumb for writing good systems papers
    - Most papers are limited
    - A few researchers consistently publish at top conferences
    - Learn from them or risk a second-rate career
    - A good framework to use for thinking about the papers we read in this class
- Haldane '28: On being the right size
  - Summary: scale and use case are determinants for the fitness of a system
- (Optional) Hamming: "You and Your Research"
  - Summary:
    - Stay open to people and ideas
    - Research is both time-consuming and exciting
    - Research inspiration comes from unexpected directions



How (and How Not) to write a Good Systems Paper (Levin & Redell)

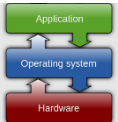
- Categories
  - Real system
  - Unimplemented system
  - Theoretical topic
- Most SOSP/OSDI papers are about prototypes
  - Some differences by field
    - E.g. architecture: more simulation (why?)
- Prototype is not enough
- Why build a system?
  - "The purpose of (scientific) computing is…
- insight, not numbers." – Richard Hamming
  - Adapt old techniques to new technology

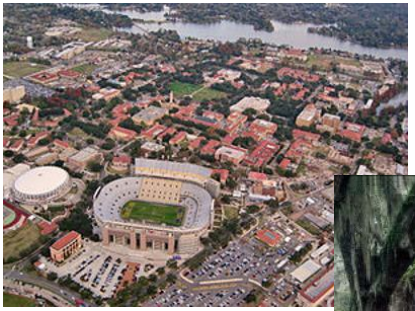
		Effort	SOSP Accept	SOSP Reject
Real system	Used by others	Huge	Occasional	Occasional
Worked once	Benchmarks ran by deadline	Large	Common	Occasional
Simulated	Simulations run	Large/med	Occasional	Common
Paper design	*Sounds* good	minimal	Rare	Common

On being the right Size (Haldane)

- Why did we read this paper?
  - Bold statements that are unusual in scientific literature
  - Intellectual pursuits require breadth and depth
- Can you draw out lessons relevant to computer systems?

- Incommensurate scaling: eye size, jumping height
  - Hardware constraints influence system design: strength of femur limits height
- Distribution of complexity and even basic assumptions change as a function of scale
  - E.g. MapReduce algorithm is a handful of lines of code
  - Apache Hadoop: 2,422,127 SLoC (as of 9/5/17)





LSU CENTER FOR COMPUTATION & TECHNOLOGY

STE||AR GROUP