Introduction CS 111 Winter 2023 Operating System Principles

Peter Reiher

Outline

- Administrative materials
- Introduction to the course
  - Why study operating systems?
  - Basics of operating systems

Administrative Issues

- Instructor and TAs
- Load and prerequisites
- Web site, syllabus, reading, and lectures
- Exams, homework, projects
- Grading
- Academic honesty

Instructor: Peter Reiher

- UCLA Computer Science department faculty member
- Long history of research in operating systems
- Email: reiher@cs.ucla.edu
- Office: No in person office hours this quarter
  – Office hours: TTh 10-11 AM, via Zoom
    - Zoom ID for office hours to be announced later
  – Often available at other times

TAs

- Section 1A: Victor Zhang
  - Email: victorzhangyf@ucla.edu
- Section 1B: Rustem Can Aygun
  - Email: canaygun10@gmail.com
- Section 1C: Yadi Cao
  - Email: yadicao95@ucla.edu
- Section 1D: Salekh Parkhati
  - Email: salekh@cs.ucla.edu
- Section 1E: Shruthi Srinarasi
  - Email: shruthi223@g.ucla.edu
- Office hours to be announced later

Instructor/TA Division of Responsibilities

- Instructor handles all lectures, readings, and tests
  - Ask me about issues related to these
- TAs handles projects
  - Ask them about issues related to these
- Generally, instructor won’t be involved with project issues
  - So direct those questions to the TAs

Web Site

- We'll primarily use a web site set up for this class
  - https://bruinlearn.ucla.edu/courses/153928
  - Schedules for reading, lectures, exams, projects
  - Project materials
    - And uploads of completed projects
  - Copies of lecture slides
  - Also Zoom IDs for lectures and taped lectures
  - Announcements
  - Sample midterm and final problems

Prerequisite Subject Knowledge

- CS 32 programming
  - Objects, data structures, queues, stacks, tables, trees
- CS 33 systems programming
  - Assembly language, registers, memory
  - Linkage conventions, stack frames, register saving
- CS 35L Software Construction Laboratory
  - Useful software tools for systems programming
- If you haven't taken these classes, expect to have a hard time in 111

Course Format

- Two weekly reading assignments
  - Mostly from the primary text
  - Some supplementary materials available on web
- Two weekly lectures
- Four (10-25 hour) individual projects
  - Exploring and exploiting OS features
  - Plus one warm-up project
- A midterm and a final exam

Course Load

- Reputation: THE hardest undergrad CS class
  - Fast pace through much non-trivial material
- Expectations you should have

– lectures	4-6 hours/week
– reading	3-6 hours/week
– projects	3-20 hours/week
– exam study	5-15 hours (twice)

- Keeping up (week by week) is critical
  - Catching up is extremely difficult

Primary Text for Course

- Remzi and Andrea Arpaci-Dusseau: *Operating Systems: Three Easy Pieces*
  - Freely available on line at http://pages.cs.wisc.edu/~remzi/OSTEP/
- Supplemented with web-based materials

Course Grading

- Basis for grading:

– Class evaluation	1%
– 1 midterm exam	20%
– Final exam	26%
– Lab 0	9%
– Other labs	11% each

- I do look at distribution for final grades
  - But don't use a formal curve
- All scores available on MyUCLA
  - Please check them for accuracy
  - Scores on BruinLearn <u>not</u> authoritative

Midterm Examination

- When: 6th week (Tuesday, February 14)
  - Replacing that day's class
  - You can take it online during any two hour period that day
- Scope: <u>All</u> material up to the exam date
  - Approximately 60% lecture, 40% text
  - No questions on purely project materials
- Format:
  - On line, multiple choice, open book/notes
- Goals:
  - Test understanding of key concepts
  - Test ability to apply principles to practical problems

Final Exam

- When: Friday, March 24
  - You can take it online during any 3 hour period that day
- Scope: Entire course
- Format:
  - On line, multiple choice, open book/notes
- Goals:
  - Determining if you have mastered the full range of material presented in the class

Lab Projects

- Format:
  - 1 warm-up project
  - 4 regular projects
  - Done individually
- Goals:
  - Develop ability to exploit OS features
  - Develop programming/problem solving ability
  - Practice software project skills

Late Assignments & Make-ups

- Labs
  - Due dates set by TAs
  - ***NOTE: They may change from the dates listed on the syllabus***
  - TAs also sets policy on late assignments
  - The TAs will handle all issues related to labs
    - Ask them, not me
    - Don’t expect me to overrule their decisions
- Exams
  - Alternate times or make-ups only possible with prior consent of the instructor
  - If you miss a test, too bad

Academic Honesty

- It is OK to study with friends
  - Discussing problems helps you to understand them
- It is OK to do independent research on a subject
  - There are many excellent treatments out there
- But all work you submit must be your own
  - Do not write your lab answers with a friend
  - Do not copy another student's work
  - Do not turn in solutions from off the web
  - If you do research on a problem, cite your sources
- I decide when two assignments are too similar
  - And I forward them immediately to the Dean
- If you need help, ask the instructor

Academic Honesty – Projects

- Do your own projects
  - If you need additional help, ask your TA
- You must design and write <u>all</u> your own code
  - Do not ask others how they solved the problem
  - Do not copy solutions from the web, files or listings
  - Cite any research sources you use
- Protect yourself
  - Do not show other people your solutions
  - Be careful with old listings

Academic Honesty and the Internet

- You might be able to find existing answers to some of the assignments on line
- Remember, if you can find it, so can we
  - And we have, before
- It IS NOT OK to copy the answers from other people's old assignments
  - People who tried that have been caught and referred to the Office of the Dean of Students
- ANYTHING you get off the Internet must be treated as reference material
  - If you use it, quote it and reference it

Academic Honesty - Tests

- It shouldn’t be necessary to say this, but . . .
- The rules for the tests will be stated before the test
- You must take the test yourself
- You must follow general UCLA academic honesty principles

Introduction to the Course

- Purpose of course and relationships to other courses
- Why study operating systems?
- What is an operating system?

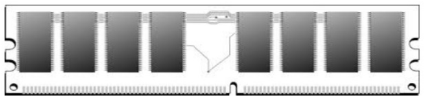
What Will CS 111 Do?

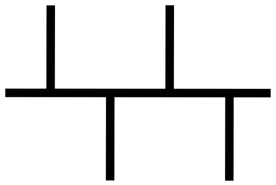
- Build on concepts from other courses
  - Data structures, programming languages, assembly language programming, computer architectures, ...
- Prepare you for advanced courses
  - Data bases, data mining, and distributed computing
  - Security, fault-tolerance, high availability
  - Network protocols, computer system modelling
- Provide you with foundation concepts
  - Processes, threads, virtual address space, files
  - Capabilities, synchronization, leases, deadlock

Why Study Operating Systems?

- Why do we have them, in the first place?
- Why are they important?
- What do they do for us?

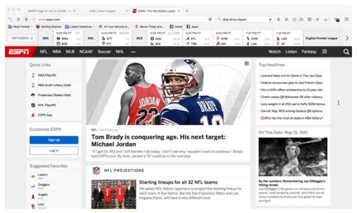
Starting From the Bottom





Here's what you've got

Here's what you want



NFL • Ian O'Connor

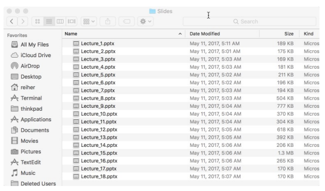
Tom Brady is conquering age. His next target: Michael Jordan

"If I get [to 45] and I still feel like I do today, I don't see why I wouldn't want to continue," Brady told ESPN.com. By then, Jordan's "6" could be in the rearview.

NFL PROJECTIONS

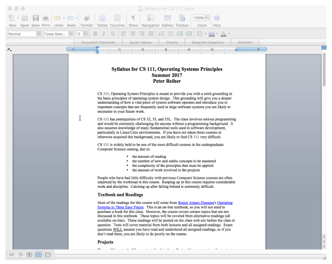
Starting lineups for all 32 NFL teams

We asked NFL Nation reporters to project the starting lineup for each team. A few teams, like the San Francisco 49ers and Los Angeles Rams, will have a very different look.



Slides

Name	Date Modified	Size	Kind
Lecture_1.pptx	May 11, 2017, 5:11 AM	189 KB	Micros
Lecture_2.pptx	May 11, 2017, 5:01 AM	175 KB	Micros
Lecture_3.pptx	May 11, 2017, 5:03 AM	169 KB	Micros
Lecture_4.pptx	May 11, 2017, 5:03 AM	181 KB	Micros
Lecture_5.pptx	May 11, 2017, 5:02 AM	211 KB	Micros
Lecture_6.pptx	May 11, 2017, 5:02 AM	196 KB	Micros
Lecture_7.pptx	May 11, 2017, 5:03 AM	194 KB	Micros
Lecture_8.pptx	May 11, 2017, 5:03 AM	504 KB	Micros
Lecture_9.pptx	May 11, 2017, 5:04 AM	777 KB	Micros
Lecture_10.pptx	May 11, 2017, 5:04 AM	370 KB	Micros
Lecture_11.pptx	May 11, 2017, 5:04 AM	304 KB	Micros
Lecture_12.pptx	May 11, 2017, 5:05 AM	618 KB	Micros
Lecture_13.pptx	May 11, 2017, 5:05 AM	392 KB	Micros
Lecture_14.pptx	May 11, 2017, 5:06 AM	206 KB	Micros
Lecture_15.pptx	May 11, 2017, 5:06 AM	1.3 MB	Micros
Lecture_16.pptx	May 11, 2017, 5:06 AM	265 KB	Micros
Lecture_17.pptx	May 11, 2017, 5:07 AM	170 KB	Micros
Lecture_18.pptx	May 11, 2017, 5:07 AM	170 KB	Micros



Syllabus for CS 111.docx

Syllabus for CS 111, Operating Systems Principles
Summer 2017
Peter Reiher

CS 111, Operating System Principles is meant to provide you with a solid grounding in the basic principles of operating system design. This grounding will give you a deeper understanding of how a vital piece of system software operates and introduce you to important concepts that are frequently used in large software systems you are likely to encounter in your future work.

CS 111 has prerequisites of CS 32, 33, and 35L. The class involves serious programming and would be extremely challenging for anyone without a programming background. It also assumes knowledge of many fundamental tools used in software development, particularly in Linux/Unix environments. If you have not taken these courses or otherwise acquired this background, you are likely to find CS 111 very difficult.

CS 111 is widely held to be one of the most difficult courses in the undergraduate Computer Science catalog, due to:

- the amount of reading
- the number of new and subtle concepts to be mastered
- the complexity of the principles that must be applied
- the amount of work involved in the projects

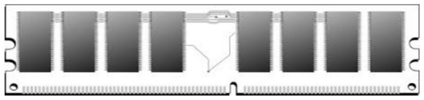
People who have had little difficulty with previous Computer Science courses are often surprised by the workload in this course. Keeping up in this course requires considerable work and discipline. Catching up after falling behind is extremely difficult.

Textbook and Readings

Most of the readings for this course will come from Remzi Arpaci-Dusseau's Operating Systems in Three Easy Pieces. This is an on-line textbook, so you will not need to purchase a book for this class. However, the course covers certain topics that are not discussed in this textbook. These topics will be covered from alternative readings (all available on-line). These readings will be posted on the class web site before the class in question. Tests will cover material from both lectures and all assigned readings. Exam questions WILL assume you have read and understood all assigned readings, so if you don't read them, you are likely to do poorly on the exams.

Projects

What Can You Do With What You've Got?

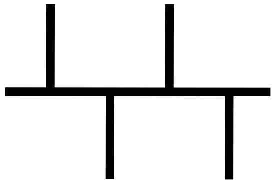


Read or write some binary words

MOV
ADD
JMP
SQRTPD

Report X and Y movements

Write to groups of pixels



READ
REQUEST SENSE

Read or write a block of data

And You Want This?

You’re Going to Need Some Help

- And that’s what the operating system is about
- Helping you perform complex operations
  - That interact
  - Using various hardware
  - And probably various bits of software
- While hiding the complexity
- And making sure nothing gets in the way of anything else

What Is An Operating System, Anyway?

- System software intended to provide support for higher level applications
  - Including higher level system software applications
  - But primarily for user processes
- The software that sits between the hardware and everything else
- The software that hides nasty details
  - Of hardware, software, and common tasks
- On a good day, the OS is your best computing friend

But Why Are You Studying Them?

- High probability none of you will ever write an operating system
  - Or even fix an operating system bug
- Not very many different operating systems are in use
  - So the number of developers for them is small
- So why should you care about them?

Everybody Has One

- Practically every computing device you will ever use has an operating system
  - Servers, laptops, desktop machines, tablets, smart phones, game consoles, set-top boxes
- Many things you don't think of as computers have CPUs inside
  - Usually with an operating system
  - Internet of Things devices
- So you <u>will</u> work with operating systems

How Do You Work With OSes?

- You configure them
- You use their features when you write programs
- You rely on *services* that they offer
  - Memory management
  - Persistent storage
  - Scheduling and synchronization
  - Interprocess communications
  - Security

Another Good Reason

- Many hard problems have been tackled in the context of operating systems
  - How to coordinate separate computations
  - How to manage shared resources
  - How to virtualize hardware and software
  - How to organize communications
  - How to protect your computing resources
- The operating system solutions are often applicable to programs and systems you write

Some OS Wisdom

- View services as objects and operations
  - Behind every object there is a data structure
- Interface vs. implementation
  - An implementation is not a specification
  - Many compliant implementations are possible
  - Inappropriate dependencies cause problems
- An interface specification is a contract
  - Specifies responsibilities of producers & consumers
  - Basis for product/release interoperability

More OS Wisdom

- Modularity and functional encapsulation
  - Complexity hiding and appropriate abstraction
- Separate policy from mechanism
  - Policy determines what can/should be done
  - Mechanism implements basic operations to do it
  - Mechanisms shouldn’t dictate or limit policies
  - Policies must be changeable without changing mechanisms
- Parallelism and asynchrony are powerful and vital
  - But dangerous when used carelessly
- Performance and correctness are often at odds
  - Correctness doesn’t always win . . .

What Is An Operating System?

- Many possible definitions
- One is:
  - It is low level software . . .
  - That provides better, more usable abstractions of the hardware below it
  - To allow easy, safe, fair use and sharing of those resources

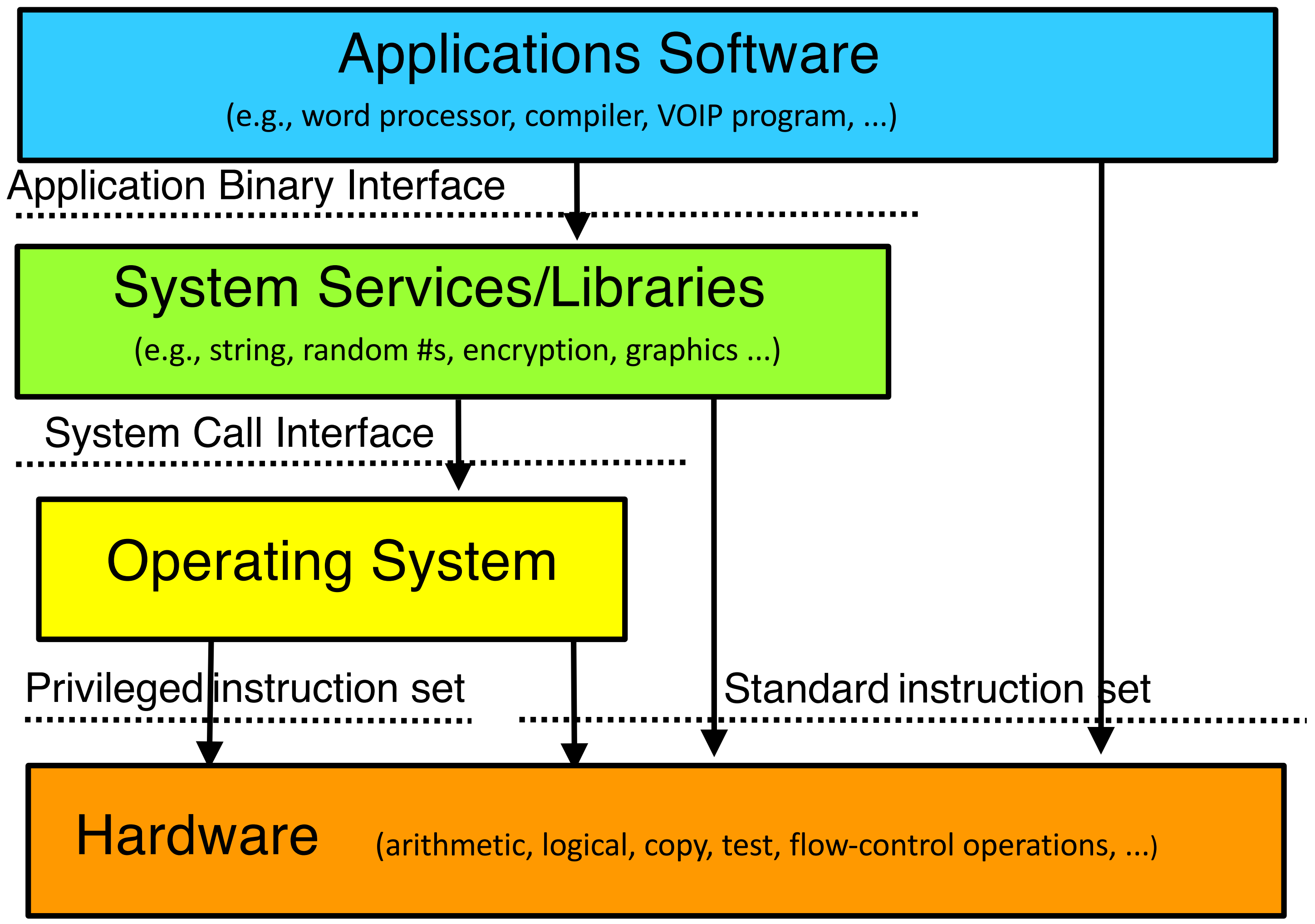
What Does an OS Do?

- It manages hardware for programs
  - Allocates hardware and manages its use
  - Enforces controlled sharing (and privacy)
  - Oversees execution and handles problems
- It abstracts the hardware
  - Makes it easier to use and improves SW portability
  - Optimizes performance
- It provides new abstractions for applications
  - Powerful features beyond the bare hardware

What Does An OS Look Like?

- A set of management & abstraction services
  - Invisible, they happen behind the scenes
- Applications see objects and their services
  - CPU supports data-types and operations
    - bytes, shorts, longs, floats, pointers, ...
    - add, subtract, copy, compare, indirection, ...
  - So does an operating system, but at a higher level
    - files, processes, threads, devices, ports, ...
    - create, destroy, read, write, signal, ...
- An OS extends a computer
  - Creating a much richer virtual computing platform
    - Supporting richer objects, more powerful operations

Where Does the OS Fit In?


Applications Software
(e.g., word processor, compiler, VOIP program, ...)
Application Binary Interface
System Services/Libraries
(e.g., string, random #s, encryption, graphics ...)
System Call Interface
Operating System
Privileged instruction set
Standard instruction set
Hardware
(arithmetic, logical, copy, test, flow-control operations, ...)

What's Special About the OS?

- It is always in control of the hardware
  - Automatically loaded when the machine boots
  - First software to have access to hardware
  - Continues running while apps come & go
- It alone has <u>complete access</u> to hardware
  - Privileged instruction set, all of memory & I/O
- It mediates applications' access to hardware
  - Block, permit, or modify application requests
- It is trusted
  - To store and manage critical data
  - To always act in good faith
- If the OS crashes, it takes everything else with it
  - So it better not crash . . .

Instruction Set Architectures (ISAs)

- The set of instructions supported by a computer
  - Which bit patterns correspond to what operations
- There are many different ISAs (all incompatible)
  - Different word/bus widths (8, 16, 32, 64 bit)
  - Different features (low power, DSPs, floating point)
  - Different design philosophies (RISC vs. CISC)
  - Competitive reasons (x86, ARM, PowerPC)
- They usually come in families
  - Newer models add features (e.g., Pentium vs. 386)
  - But remain upwards-compatible with older models
    - A program written for an ISA will run on any compliant CPU

Privileged vs. General Instructions

- Most modern ISAs divide the instruction set into *privileged* vs. *general*
- Any code running on the machine can execute general instructions
- Processor must be put into a special mode to execute privileged instructions
  - Usually only in that mode when the OS is running
  - Privileged instructions do things that are "dangerous"

Platforms

- ISA doesn't completely define a computer
  - Functionality beyond user mode instructions
    - Interrupt controllers, DMA controllers
    - Memory management unit, I/O busses
    - BIOS, configuration, diagnostic features
    - Multi-processor & interconnect support
  - I/O devices
    - Display, disk, network, serial device controllers
- These variations are called "platforms"
  - The platform on which the OS must run
  - There are lots of them

Portability to Multiple ISAs

- A successful OS will run on many ISAs
  - Some customers cannot choose their ISA
  - If you don’t support it, you can’t sell to them
- Which implies that the OS will abstract the ISA
- Minimal assumptions about specific HW
  - General frameworks are HW independent
    - File systems, protocols, processes, etc.
  - HW assumptions isolated to specific modules
    - Context switching, I/O, memory management
  - Careful use of types
    - Word length, sign extension, byte order, alignment
- How can an OS manufacturer distribute to all these different ISAs and platforms?

Binary Distribution Model

- Binary is the derivative of source
  - The OS is written in source
  - But a source distribution must be compiled
  - A binary distribution is ready to run
- OSes usually distributed in binary
- One (or more) binary distributions per ISA
- Binary model for platform support
  - Device drivers can be added, after-market
    - Can be written and distributed by 3rd parties
    - Same driver works with many versions of OS

Binary Configuration Model

- Good to eliminate manual/static configuration
  - Enable one distribution to serve all users
  - Improve both ease of use and performance
- Automatic hardware discovery
  - Self-identifying busses
    - PCI, USB, PCMCIA, EISA, etc.
  - Automatically find and load required drivers
- Automatic resource allocation
  - Eliminate fixed sized resource pools
  - Dynamically (re)allocate resources on demand

What Functionality Is In the OS?

- As much as necessary, as little as possible
  - OS code is <u>very expensive</u> to develop and maintain
- Functionality must be in the OS if it ...
  - Requires the use of privileged instructions
  - Requires the manipulation of OS data structures
  - Must maintain security, trust, or resource integrity
- Functions should be in libraries if they ...
  - Are a service commonly needed by applications
  - Do not actually have to be implemented inside OS
- But there is also the performance excuse
  - Some things may be faster if done in the OS

Conclusion

- Understanding operating systems is critical to understanding how computers work
- Operating systems interact directly with the hardware
- Operating systems rely on stable interfaces