CS 370: OPERATING SYSTEMS
[INTRODUCTION]

Hiding in Plain Sight, the Operating System
Makes all you do possible
on toasters, tablets, PCs, or servers

With many a role
A referee, an illusionist, and the glue

Balancing competing needs,
resolving conflict, and targeted sharing

Be it
threading the concurrency needle
memory management, scheduling, or
circumventing deadlocks

Shrideep Pallickara
Computer Science
Colorado State University

COMPUTER SCIENCE DEPARTMENT

COLORADO STATE UNIVERSITY

1

Topics covered in this lecture

- Expectations
- Course Overview
- Introduction

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.2

2

EXPECTATIONS

What it takes to succeed

- You are required to work at least **6-8 hours** per-week outside of class
  - Coding and reviewing material from class
- If you miss a lecture?
  - Add about 3 hours per missed lecture

Pitfalls to avoid?

- Believing that you can learn via osmosis
- **Missing lectures**
  - If you don't have the discipline to come to class, you are unlikely to have the discipline to catch up
- **Procrastinating**
  - Get started on the assignments early

Why attend lectures if all the slides are posted?

- Slides are only part of the story
  - They anchor the discussion
- Any field has a *language* associated with it
- People who have worked in an area for a long time speak the language
  - Sitting in classes helps you learn how to frame questions and responses
- Often there are surprising questions
  - Some of these may be asked by interviewers

Help me help you

- We will have **surveys** at the end of every class
- You will provide a list of
  - 3 concepts you followed clearly
  - 3 concepts you had problems keeping up with
- Problem areas for the majority of the class will be addressed in the next class

Interactions

- You can have discussions with me, the TAs, and your peers
- There are two constraints to these discussions
  - No code can be exchanged under any circumstances
  - No one takes over someone else's keyboard
- Bumps are to be expected along the way
  - But you should get over this yourself
  - It will help you with the next problem you encounter

Communications

- Please DO NOT use Canvas messaging for communications
  - Please send communications to **compsci_cs370@colostate.edu**
- The e-mail account is checked by the entire team and allows us to respond to communications in a timely fashion
- Send e-mails from accounts that match your name
  - **No pseudonyms please**
- Do not post code on the MS Teams Channel

You are not allowed to take learning opportunities away from other students

- If you must use a laptop or tablet (in the keyboard mode) you should
  - Sit in the last row
  - Turn off wireless
  - Sign and turn in pledge forms
  - Use it only for taking notes
- If you are using a tablet in the stylus/pencil-mode, you may sit anywhere in class; also, sign the pledge form
- When the class is in session, put away your cell-phones!
- Please no cross-talking when the class is in-session

Course webpage

- All course materials will be accessible via the public-facing webpage (**https://www.cs.colostate.edu/~cs370**)
  - Schedule (Lecture slide sets for each lecture)
  - Assignments
  - Syllabus
  - Grading
- Grades will be posted on **Canvas**; assignment submissions will be via Canvas
- The course website, MS Teams Channel, and Canvas are all live now

Office Hours: Details on course webpage

- Professor
  - Shrideep Pallickara
  - Fridays 3:00-4:00 pm in CSB-364 and via Zoom
  - Focused on **course concepts**
- TA Office hours focused exclusively on **programming assignments**
  - Office Hours: CSB-120 and MS Teams
  - GTAs: Richi Rodriguez, William Scarbro, and Anindya Chowdhury
  - UTAs: Matthew Maloney, Henry Gates, Cameron Suess, and Hamad Alyami

TA Office Hours: Almost Finalized

****All changes will be reflected on the course webpage**

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Richi Rodriguez	1:00-5:00 pm		4:00-8:00 pm	1:00-5:00 pm	2:00-5:00 pm	
William Scarbro	12:00-5:00 pm	1:00-5:00 pm	10:00 am - 3:00 pm	2:00-4:00 pm		
Anindya.Chowdhury	5:00-6:00 pm	2:00-6:00 pm	5:00-6:00 pm	2:00-6:00 pm	12:00-5:00 pm	
Matthew Maloney	3:000-6:00 pm		3:00-7:00 pm			1-4:00 pm
Henry Gates	4:00-7:00 pm		4:00-7:00 pm		4:00-7:00 pm	
Cameron Suess	11:00-noon	10:00-11:59 am	11:00-noon	10:00-11:59 am	10:00-1:00 pm	
Hamad Alyami	6:00-8:00 pm	7:00-8:00 pm	6:00-8:00 pm	7:00-8:00 pm	3:00-6:00 pm	

Topics that we will cover in CS 370

- Processes and Threads
- Process Synchronization (plus Atomic Transactions)
- CPU Scheduling: MFQ, CFS
- Deadlocks
- UNIX I/O
- Memory Management
- File System interface and management. Unix file system. NTFS.
- Storage Management including SSDs and Flash Memory
- Virtualization and Containers

Course Textbook

- *Operating Systems Concepts, 10th edition*
  Avi Silberschatz, Peter Galvin, and Greg Gagne Publisher - John Wiley & Sons, Inc.
  ISBN-13: 978-1119800361

COLORADO STATE UNIVERSITY | Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT | INTRODUCTION | L1.15

15

When I make slides, I usually refer to several texts. These include …

- *Andrew S Tanenbaum and Herbert Bos. Modern Operating Systems. 4th Edition, 2014. Prentice Hall. ISBN: 013359162X/ 978-0133591620*
- *Thomas Anderson and Michael Dahlin. Operating Systems: Principles and Practice, 2nd Edition. Recursive Books. ISBN: 0985673524/978-0985673529*
- *Remzi Arpaci-Dusseau and Andrea Arpaci-Dusseau. Operating Systems: Three Easy Pieces. 1st edition. CreateSpace Independent Publishing Platform. ISBN-13: 978-1985086593*
- *Kay Robbins & Steve Robbins. Unix Systems Programming, 2nd edition, Prentice Hall ISBN-13: 978-0-13-042411-2*
- I always list my references at the end of every slide set

COLORADO STATE UNIVERSITY | Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT | INTRODUCTION | L1.16

16

INFOSPACES (**https://infospaces.cs.colostate.edu**)

- **Knowledge repository** my lab has been building to enhance learning
- All videos are designed to be less than 2 minutes
- Improving INFOSPACES
  - Let us know what you would like to see
  - If you'd like to contribute to this repository let us know!

Plagiarism detection

- Use of generative AI (GitHub co-pilot, ChatGPT, Maude and their ilk) is expressly disallowed at stage (include ideation) for coding or the term project report
  - Will be considered plagiarism and cheating
- All programming assignments will be subject to pair-wise comparisons
  - Colluding, coping from the same source on the internet, and using paid-versions of GenAI for solutions will all be detected

GRADING

19

Grading breakdown

- Assignments: 45%
  - 5 programming assignments (3 C, 1 Java, and 1 C++)
- Quizzes: 10%
- Mid Term: 20%
- Comprehensive Final Exam: 25%
- Two sets of extra credit (optional) opportunities
  - Programming Exercises: 1.5% in total
  - Page faults assignment: 2% in total

20

Grading Policy I

- Letter grades will be based on the following standard breakpoints:
  - >= 90 is an A, >= 88 is an A-,
  - >=86 is a B+, >=80 is a B, >=78 is a B-,
  - >=76 is a C+, >=70 is a C,
  - >=60 is a D, and <60 is an F.
- I will not cut higher than this, but I *may* cut lower.
- There will be **no make-up exams**
  - Exceptions for extenuating circumstances with documentation

Grading Policy II

- Every assignment will be posted at least 2 weeks before the due date.
  - Every assignment will include information about how much it will count towards the course grade, and how it will be graded.
- Late submission penalty: 10% per-day for the first 2 days and a ZERO thereafter.
  - Detailed submission instructions posted on course website.
  - Programming assignments will be graded within 30-60 seconds of submission

For the Quizzes and Tests

- I will only ask questions about what I teach
  - If I didn't teach it, I won't ask from that portion
- If the concepts were covered in my slides
  - You should be be able to answer the questions
- I won't ask questions about arcane aspects of some esoteric device controller

Exams

- There will be one mid-term (20%)
- The final exam is comprehensive (25%)
- There will be 13 quizzes **in-class**
  - 3 quizzes where you had your lowest scores will be dropped
    - We will compute the average of your 10 highest scores
      - 10% of your course grade

Term project

- Team project
  - Team size is 2-3
- Based on the Raspberry Pi
  - Plus, a sensor and desktop: Released

Assignments schedule

	Release	Due Date
Programming Exercises [Extra Credit & Optional]	21-Jan	1/29, 2/5, and 2/12
HW1	21-Jan	5-Feb
HW2	29-Jan	19-Feb
HW3	5-Feb	26-Feb
HW4	19-Feb	12-Mar
HW5	24-Mar	16-Apr
Term Project	TP-D1	5-Feb
	TP-D2	26-Mar
	TP-D3	7-May
HW-Extra Credit [Optional]	2-Apr	23-Apr

ABOUT ME

About me

- I do research in the area of large-scale computing systems, Big Data, and GeoAI
- My research has been funded by agencies in the United States and the United Kingdom
  - These include the National Science Foundation, the Department of Homeland Security (including the *Long Range* program), the Environmental Protection Agency, the Department of Agriculture, the National Institute of Food & Agriculture, the National Endowment for the Humanities/Teagle and the U.K's e-Science program
  - Recipient of the National Science Foundation's CAREER Award
  - I direct the Center for eXascale Spatial Data Analytics and Computing (XSD) @ CSU [**https://spatial.colostate.edu**]

My research has been deployed in

- Urban sustainability
- Commercial internet conferencing systems
- Defense applications
- Precision Agriculture
- Earthquake sciences
- Epidemic modeling
- Healthcare
- Bioinformatics
- Brain Computer Interfaces
- High energy physics
- Visualizations

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.29

29

OPERATING SYSTEMS: A BRIEF OVERVIEW

COMPUTER SCIENCE DEPARTMENT COLORADO STATE UNIVERSITY

30

A modern computer is a complex system

- Multiple processors and co-processors
- Main memory and Disks
- Keyboard, Mouse and Displays
- Network interfaces
- I/O devices

Why do we need Operating Systems?

- If every programmer had to understand how *all* these components work?
  - Software development would be arduous
- Managing all components and using them optimally is a challenge

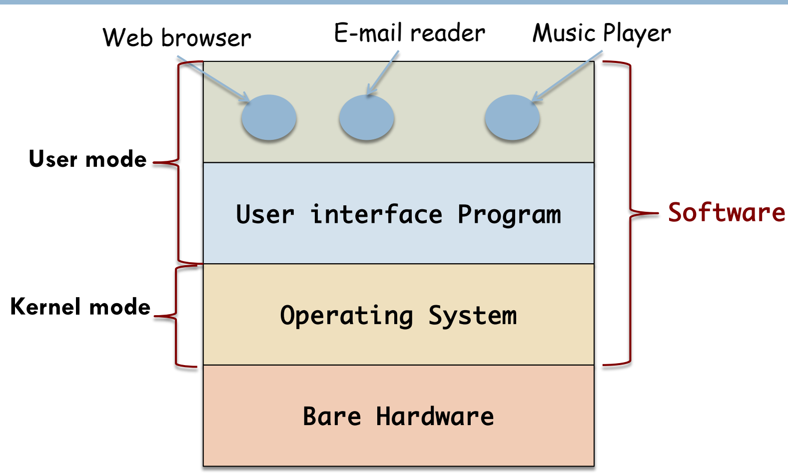
Computers are equipped with a layer of software

- Called the **Operating System**
- Functionality:
  - Provide user programs with a better, simpler, cleaner model of the computer
  - Manage resources efficiently

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.33

33

Where the operating system fits in [1/3]



COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.34

34

Where the operating system fits in [2/3]

- The OS runs on bare hardware in **kernel mode**
  - *Complete access* to all hardware
  - Can execute *any* instruction that the machine is capable of executing
- Provides the base for all software
  - Rest of the software runs in **user-mode**
    - Only a **subset** of machine instructions is available

Where the operating system fits in [3/3]

- Users interact with applications
  - Applications execute in an environment provided by the operating system
    - And the operating system mediates access to the underlying hardware

The application context is much more than a simple abstraction on top of hardware devices

- Applications execute in a virtual environment that is more **constrained** (to prevent harm)
- More **powerful** (to mask hardware limitations), and …
- More **useful** (via common services) than the underlying hardware

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.37

37

The OS as an extended machine

- The **architecture** of a computer includes
  - Instruction set, memory organization, I/O, and bus structure
- The architecture of most computers at the machine language level
  - Primitive and awkward to program especially for I/O

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.38

38

Let's look at an example of floppy disk I/O done using NEC PD765

- The PD765 has 16 commands
  - For reading and write data, moving the disk arm, formatting tracks, etc.
  - Specified by loading 1-9 bytes into the device register
- Most basic commands are for read and write
  - 13 parameters packed into 9 bytes
    - Address of disk block, number of sectors/track, inter-sector gap spacing etc.

But that's not the end of it …

- When the operation is completed
  - Controller returns 23 status and error fields packed into 7 bytes
- You must also check the status of the **motor**
  - If it is off? Turn it on before reading or writing
  - Don't leave the motor on for too long
    - Floppy disk will wear out
  - TRADEOFF: Long start-up delay *versus* wearing out disk

Of course, the average programmer does not want to have any of this

- What they would like is a simple, high-level **abstraction** to deal with
- For a disk this would mean a collection of named **files**
  - Operations include open, read, write, close, etc.
  - BUT NOT
    - Whether the recording should use frequency modulation
    - The state of the motor

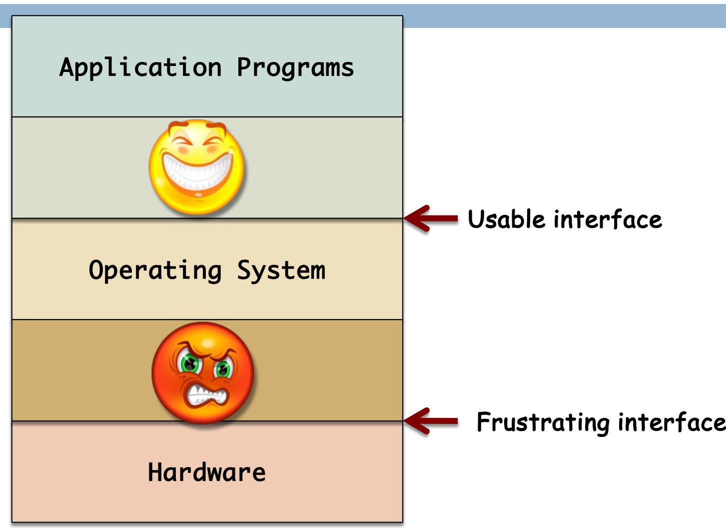
Why do processors, disks, etc. present difficult, awkward, idiosyncratic interfaces ?

- Backward compatibility with older hardware
- Desire to save money
- Sometimes hardware designers don't realize (or care) how much trouble they cause!

Why abstractions are important

- Abstraction is the key to managing **complexity**
- Good abstractions turn a nearly impossible task into two manageable ones
  1. Defining and implementing abstractions
  2. Using abstractions to solve problem
- Example
  - File

Operating systems turn frustrating hardware into usable interfaces



Roles of an Operating System

45

The three roles of an Operating System

- **Referee**
  - Isolate applications from each other
- **Illusionist**
  - Provide an abstraction of physical hardware to simplify application design
  - Because applications are written to a higher level of abstraction, the OS can invisibly change the amount of resources assigned to each application
- **Glue**
  - Provides a set of common services to facilitate sharing among applications
  - As a result, *cut-and-paste* works uniformly across the system; a file written by one application can be read by another

46

Referee: Facilitating resource sharing

- Provide **orderly** and **controlled** allocation of resources to programs competing for them
  - Processors, memories, and I/O devices

Referee: The OS a Resource Allocator

- An OS may receive numerous & conflicting requests for resources
  - Prevent errors and improper use
- Resources are scarce and expensive
- The OS allocates resources to specific programs and users
  - The allocation must be efficient and fair
  - Must increase overall system throughput
- Seemingly trivial differences in how resources are allocated can impact user-perceived performance

Referee: Providing isolation

- An operating system must protect itself and other applications from programmer bugs
  - Debugging would be vastly harder if an error in one program could corrupt data structures in other applications
- **Fault isolation** requires restricting the behavior of applications to less than the full power of the underlying hardware

Referee: Facilitating Communications

- The flip side of isolation is the need for **communication** between different applications and different users
- In setting up boundaries, an OS must also allow those boundaries to be crossed in **carefully controlled ways** when the need arises!

In its role as referee, an OS is like a particularly patient kindergarten teacher.
It balances needs, separates conflicts, and facilitates sharing.

The OS as an Illusionist: Masking Limitations

- **Physical constraints limit hardware resources** — a computer has only a limited number of processors and a limited amount of physical memory, network bandwidth, and disk
- Since the OS must decide how to *divide its fixed resources* among the various applications running at each moment …
  - A particular application can have differing amounts of resources from time to time, even when running on the *same* hardware

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.51

51

The OS as a Glue: Providing Common Services

- Providing a set of common, standard services to applications to simplify and standardize their design
- The OS serves as an **interoperability layer** so that both applications and devices can evolve independently
- OSes provide a set of standard user interface widgets
  - Facilitates a common "look and feel" to users so that frequent operations — such as pull-down menus and "cut" and "paste" commands — are handled consistently across applications

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.52

52

Defining Operating Systems

- Solves the problem of creating a usable computing system
  - Makes solving problems easier
- Control, allocate and mediate access to resources
- It is the one program that is running all the time: **kernel**

COLORADO STATE UNIVERSITY Professor: SHRIDEEP PALLICKARA COMPUTER SCIENCE DEPARTMENT INTRODUCTION L1.53

53

A (VERY) BRIEF HISTORY OF OPERATING SYSTEMS

COMPUTER SCIENCE DEPARTMENT COLORADO STATE UNIVERSITY

54

The first true digital computer was designed by Charles Babbage (1792-1871)

- Spent most of his life and fortune trying to build the analytical engine
- Never got it working properly
  - Purely mechanical
  - Technology of the day could not produce wheels, cogs, gears to the required precision
- Did not have an operating system

Babbage realized he would need software for his analytical engine

- Hired Ada Lovelace as the worlds first programmer
  - Daughter of British poet Lord Byron
- The programming language Ada® is named after her

The First Generation (1945-55) Vacuum Tubes

- First fully functioning digital computer built at Iowa State University
  - Prof. John Atanasoff and grad student Clifford Berry
- All programming in absolute machine language
  - Also, by wiring up electrical circuits
    - Connect 1000s of cables to plug boards to control machine's basic functions
  - Operating Systems were unheard of
- Straightforward numerical calculations
  - Produce tables of sines, cosines, logarithms

The Second Generation (1955-1965): Transistors and Batch Systems

- **Separation** between designers, builders, operators, programmers, and maintenance
- Machines were called **mainframes**
- Write a program on paper, then punch it on cards
  - Give card deck to operator and go drink coffee
  - Operator gives output to programmer

The Third Generation (1965-1980) ICs and Multiprogramming

- Managing different product lines was expensive for manufacturers
  - Customers would start with a small machine, and then outgrow it
- IBM introduced the Systems/360
  - Series of **software-compatible** machines
  - All machines had the same instruction set
    - Programs written for one machine could run on all machines

The Fourth Generation (1980-Present) Personal Computers

- Large Scale Integration circuits (LSI)
  - Thousands of transistors on a square centimeter of silicon
- 1974: Intel came out with the 8080
  - General purpose 8-bit CPU
- Early 1980s IBM designed the IBM PC
  - Looked for an OS to run on the PC
  - Microsoft purchased Disk Operating System and went back to IBM with MS-DOS

The contents of this slide-set are based on the following references

- *Andrew S Tanenbaum and Herbert Bos. Modern Operating Systems. 4th Edition, 2014. Prentice Hall. ISBN: 013359162X/ 978-0133591620* [Chapter 1]
- *Avi Silberschatz, Peter Galvin, Greg Gagne. Operating Systems Concepts, 9th edition. John Wiley & Sons, Inc. ISBN-13: 978-1118063330.* [Chapter 1]
- *Thomas Anderson and Michael Dahlin. Operating Systems: Principles and Practice, 2nd Edition. Recursive Books. ISBN: 0985673524/978-0985673529. [Chapters 1-2]*
- *Kay Robbins & Steve Robbins. Unix Systems Programming, 2nd edition, Prentice Hall ISBN-13: 978-0-13-042411-2.* [Chapter 1]