



# Operating Systems

CS3281 / CS5281

Spring 2024



Tel (615) 343-7472 | Fax (615) 343-7440  
1025 16th Avenue South Nashville, TN 37212  
[www.isis.vanderbilt.edu](http://www.isis.vanderbilt.edu)



# Team

- Instructors
  - Dr. Andy Gokhale, Dr. Abhishek Dubey, Dr. Shervin Hajiamini
- Graduate TAs
  - Sanjana Das, Saman Kittani
- Graders
  - Salomon Dushimirimana, Mohammed Khan, Rana Muhammad Shahroz Khan, Ilayda Koca, Chuci Liu, Xiaohan Liu, Binh Mai, Akash Munagala, Yurui Xu



# Top Hat

- You will receive an email for your section. Please join.
- Then you will receive a question about github id. Do join.



# Office Hours

- See on github.



# Important Links

Textbook	<a href="http://pages.cs.wisc.edu/~remzi/OSTEP/">http://pages.cs.wisc.edu/~remzi/OSTEP/</a>
Discussion Forum & Announcements	<a href="https://piazza.com/vanderbilt/spring2024/cs32815281">https://piazza.com/vanderbilt/spring2024/cs32815281</a>
Lectures	<a href="https://github.com/cs3281/lectures">https://github.com/cs3281/lectures</a>
Programming Assignments	<a href="https://classroom.github.com/classrooms/30844110-cs3281-classroom-spring2024">https://classroom.github.com/classrooms/30844110-cs3281-classroom-spring2024</a>
Announcements & Administration	<a href="https://vanderbilt.edu/brightspace/">https://vanderbilt.edu/brightspace/</a>



# Programming Assignments

- Administered through GitHub Classroom
  - Requires admission to the cs3281 repository using your GitHub ID



# Programming Assignments

- Administered through GitHub Classroom
  - Requires admission to the cs3281 repository using your GitHub ID

[TOPHAT GITHUB ID COLLECTION]



# Programming Assignments

- Administered through GitHub Classroom
  - Requires admission to the cs3281 repository using your GitHub ID
- Graded using the VUIT-provided AWS Virtual Machines
  - Access your WorkSpace on a virtual machine in a browser at <https://webclient.amazonworkspaces.com>

**or** download the AWS WorkSpaces client from:  
<https://clients.amazonworkspaces.com/>

- Enter the following registration code:  
SLiad+DGTNYL
- Log in with your VUNetID and password





# Programming Assignments

- Full instructions for using xv6 are at: <https://github.com/cs3281/lectures/wiki>
- Automate GitHub credentials:
  - Go to <https://github.com/settings/tokens>
  - Click "Generate new token (classic)"
    - Note: Development
    - Expiration: No expiration
    - Select scopes: repo, user, codespace
  - Click "Generate token"
  - Copy Personal Access Token and write it down or save it somewhere
  - In a terminal, run "git clone <https://github.com/cs3281/lectures.git>"
    - When prompted, enter you GitHub username and enter the above-generated Personal Access Token as your password
    - You should never be prompted to do enter these credentials again
  - In a terminal, run "git config --global credential.helper store"
    - Git should save your token in ~/.git-credentials file

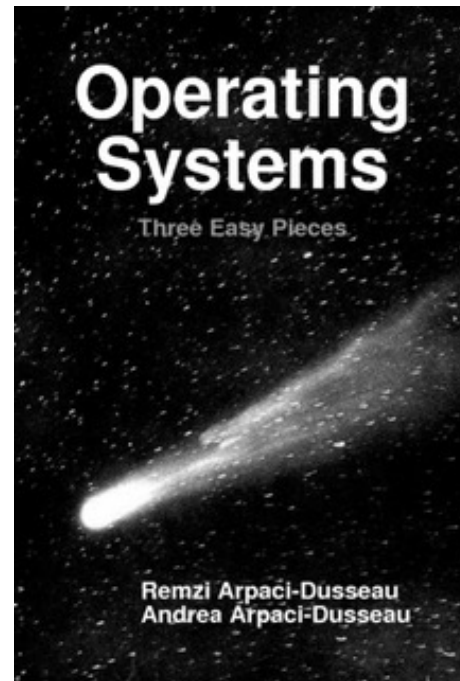
GDB reads gdbinit file from the current directory in your virtual machine

- In a terminal, run:
  - `mkdir -p $HOME/.config/gdb && echo "set auto-load safe-path /" > $HOME/.config/gdb/gdbinit`



# Textbook

- We will draw material from a variety of sources
- Primary textbook: Operating Systems: Three Easy Pieces
  - Available for free at <http://pages.cs.wisc.edu/~remzi/OSTEP/>
- Other possible texts:
  - The Linux Programming Interface
  - Computer Systems: A Programmer's Perspective
  - Linux Kernel Development



# Course Assessment

- Programming assignments: 50%
  - Learn by doing
- In-Class assignments/quizzes: 10%
- Mid-term exam: 15%
- Final exam: 25%
- Participation, Professionalism, Attendance:  $\pm 2\%$



# Late days

- You have a total of 4 late days that you can use across programming assignments as you wish
  - A maximum of two late days can be used on a given programming assignment
  - Example: assignment is due by 11:59pm Monday; you can use two late days to submit that assignment by 11:59pm Wednesday with no penalty
- To use late days: push a file named `late_days.md` to the top-level directory of your assignment repo with a line stating whether you're using one or two late days
- Assignments submitted more than two days late will not be accepted
- **No collaborations unless explicitly permitted**



# Regrade Requests

- Grading errors can and do happen
  - Not malicious!
- You can “challenge” if you believe an error was made
- If you are wrong, you may lose a late day



# ChatGPT and AI assistants



- We have entered a new era with AI assistants
- Our goal is to use them to enhance the learning experience
- We are adopting a new experimental policy
  - You can use ChatGPT and other AI tools however you want!
  - However, you must document your use
    - No points will be deducted for use of AI, so please document your use authentically
    - We want you to reflect on how these technologies are helpful and how they aren't. We want to know both how it failed, as well as how it succeeded!
- This policy is experimental, and is subject to change at the instructor's discretion

# Linux

- The Linux operating system is open source but very complex
  - Over 25 million source lines of code!
  - Many performance optimizations, which can obscure fundamental concepts
- You should learn to use Linux as it is widely used in industrial settings, and many applications, especially in research, use it exclusively
- It is unreasonable to implement many course concepts in Linux itself
- Watch this video: [https://www.youtube.com/watch?v=actgZVvs\\_Jk](https://www.youtube.com/watch?v=actgZVvs_Jk)



# xv6

- Instead, we will use the teaching-oriented operating system xv6
- xv6 developed and used in OS class at MIT
  - Only ~5,000 source lines of code
  - Also used at many other universities
- We will use some existing xv6 assignments and some of our own
  - Posting or sharing solutions is a serious violation of the honor code and will be treated as such
  - Copying solutions from the web is also an honor-code violation and will be directed to the honor council





# Course expectations

- You are expected to read the material for a lecture beforehand and participate in class discussions
  - We may occasionally assign videos to watch; please watch them ahead of time so you can participate in class discussions
- All assignments can be found on Brightspace
  - Check back frequently
  - Brightspace may be updated up to a week before any given lecture based on class progress and timeline



# Course expectations: Office Hours

- We will use an office-hour policy similar to the one listed here:  
<https://www2.seas.gwu.edu/~gparmer/resources/2021-09-20-Office-Hours-HOWTO.html>
  - Please read this policy carefully and adhere to its guidelines
  - It will help you and us



# Development Environment

- Most of this course will use C
  - This is an OS course, not a course on C
  - While some C material will be discussed, it is up to you to learn and build your proficiency in C if you aren't proficient already
    - Remember, you can ask ChatGPT for help!
- We will assume Ubuntu as the development environment
  - VMs provided through VUIT
  - You're free to choose your development environment, but your work will be tested and evaluated on the Ubuntu VM environment
  - We do not support alternative environments
- We will use GitHub and git for content and assignment management



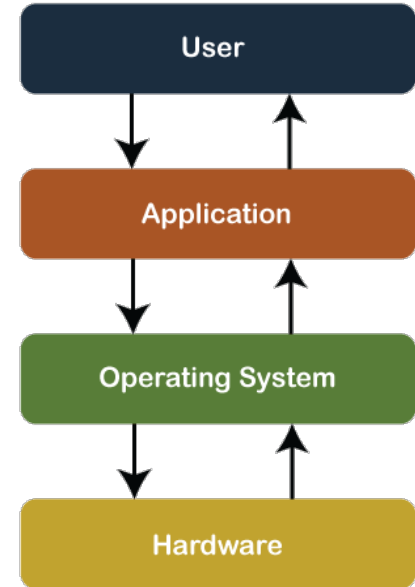
# Development Environment

- We will use the MIT-sponsored “xv6” teaching OS for most assignments
  - Based on the original UNIX kernel
  - Allows us to actually hack on a real kernel
  - Runs on a RISC-V architecture which will be emulated on your VM through the QEMU application
  - Good idea to read relevant sections of the “xv6 book” while doing homework assignments:  
<https://pdos.csail.mit.edu/6.S081/2023/xv6/book-riscv-rev3.pdf>
  - Exit xv6/QEMU by pressing “Ctrl+a” followed by “x”

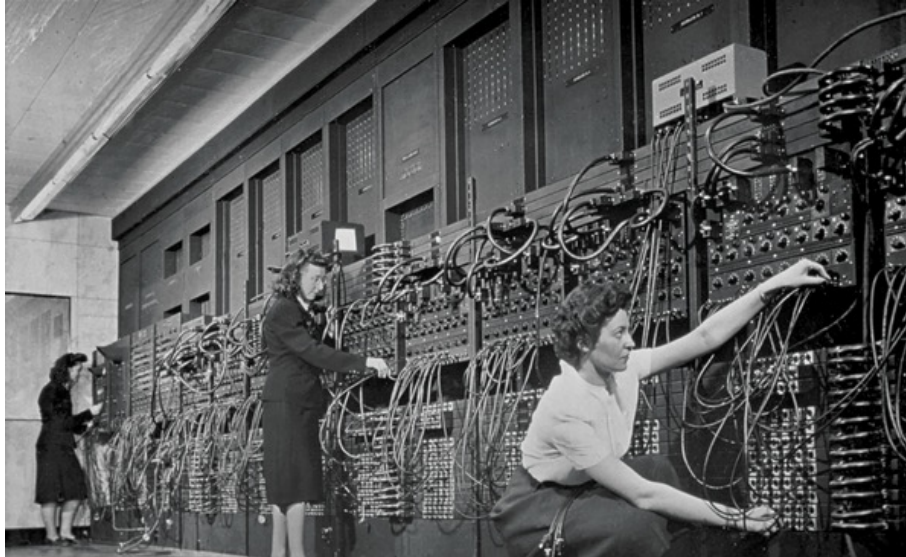


# Course Goals

- Understand operating systems by learning their architecture and services
- Experience with low-level systems development



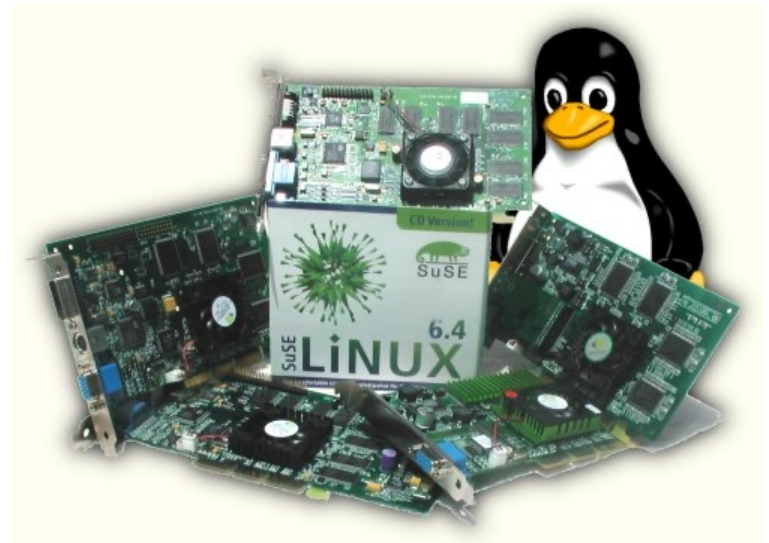
# Historical Perspective



Early computers did not have an operating system. People manually performed functions that are now controlled in software systems that operate the machine

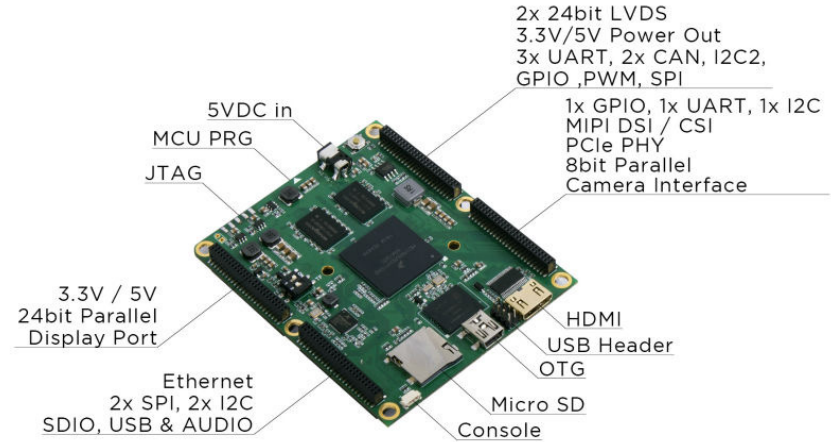
# Why is this important?

- Operating system is responsible for
  - Abstracting the hardware details for convenience and portability



# Why is this important?

- Operating system is responsible for
  - Abstracting the hardware details for convenience and portability
  - Multiplexing the hardware among multiple applications





# Why is this important?

- Operating system is responsible for
  - Abstracting the hardware details for convenience and portability
  - Multiplexing the hardware among multiple applications
  - Isolating applications to contain bugs



# Example: USB device insertion

- Consider what happens when you plug-in a USB device to your laptop
  - USB controller informs its driver
    - The driver is part of the kernel
  - The driver asks the device to identify itself
    - Device sends back an ID
  - Controller uses ID to match a driver to the new device



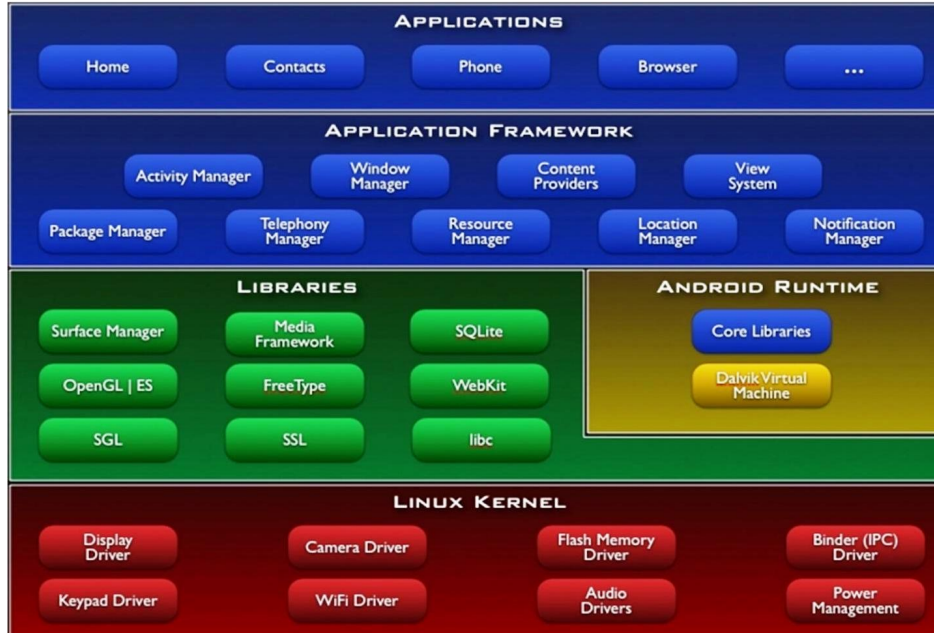
A typical USB connector, called an "A" connection



Inside a USB cable: There are two wires for power -- +5 volts (red) and ground (brown) -- and a twisted pair (yellow and blue) of wires to carry the data. The cable is also shielded.

# Layers of a modern computing system

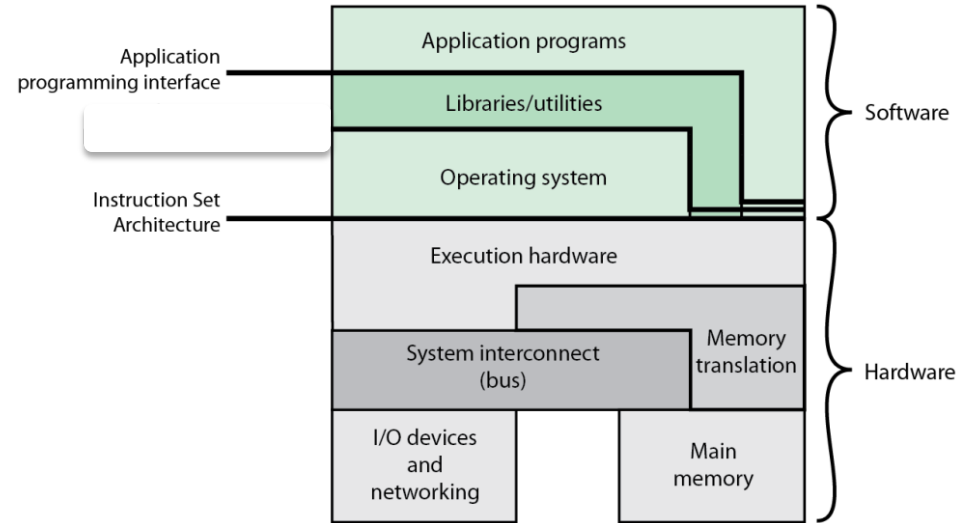
## Android™ Architecture



- Application – the user program
- Application framework
  - Helpful libraries for providing modularity and reuse
- System Libraries – the core services of the OS are encapsulated by these libraries, e.g. libc
- The operating system kernel

# Layers of a modern computing system

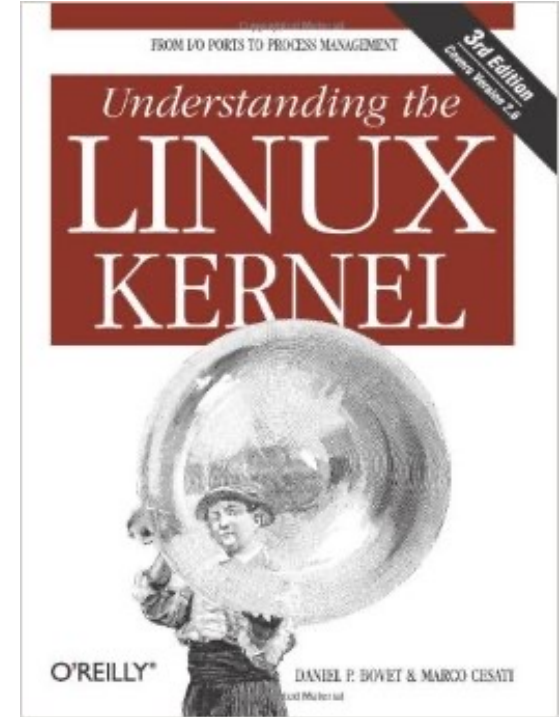
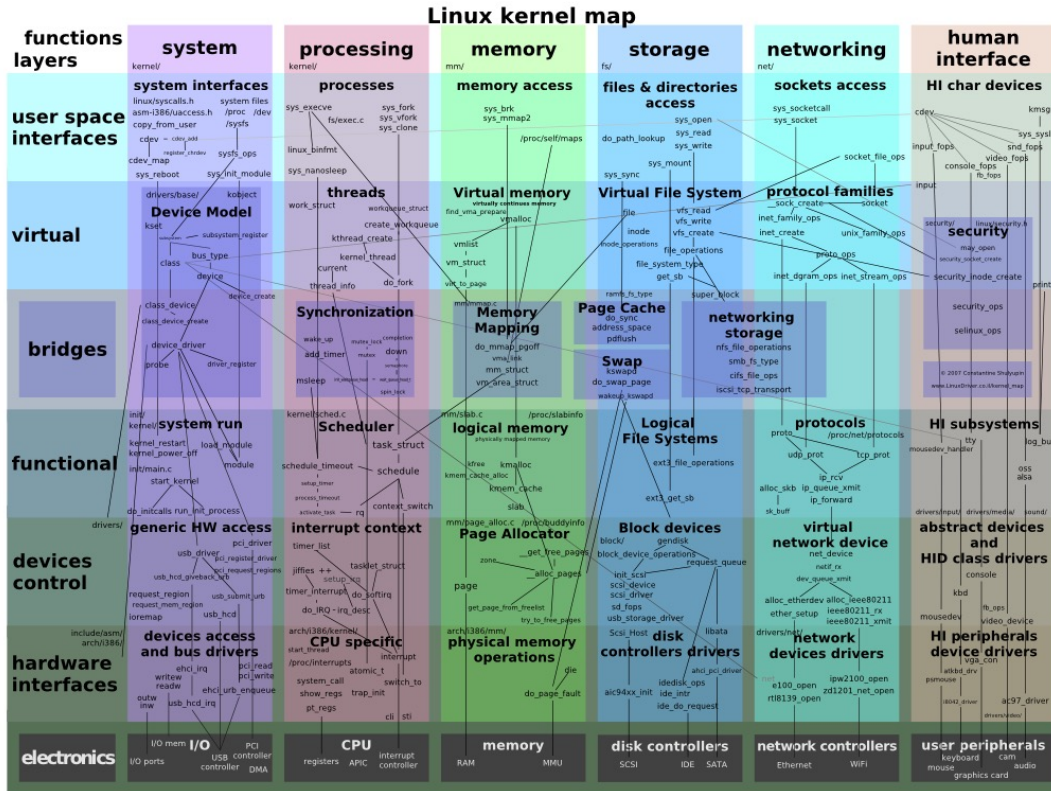
- Application – the user program.
- Application framework
  - Helpful libraries for providing modularity and reuse
- **System Libraries – the core services of OS are encapsulated by these libraries, e.g. libc**
- **The operating system kernel**



Another view of the layers

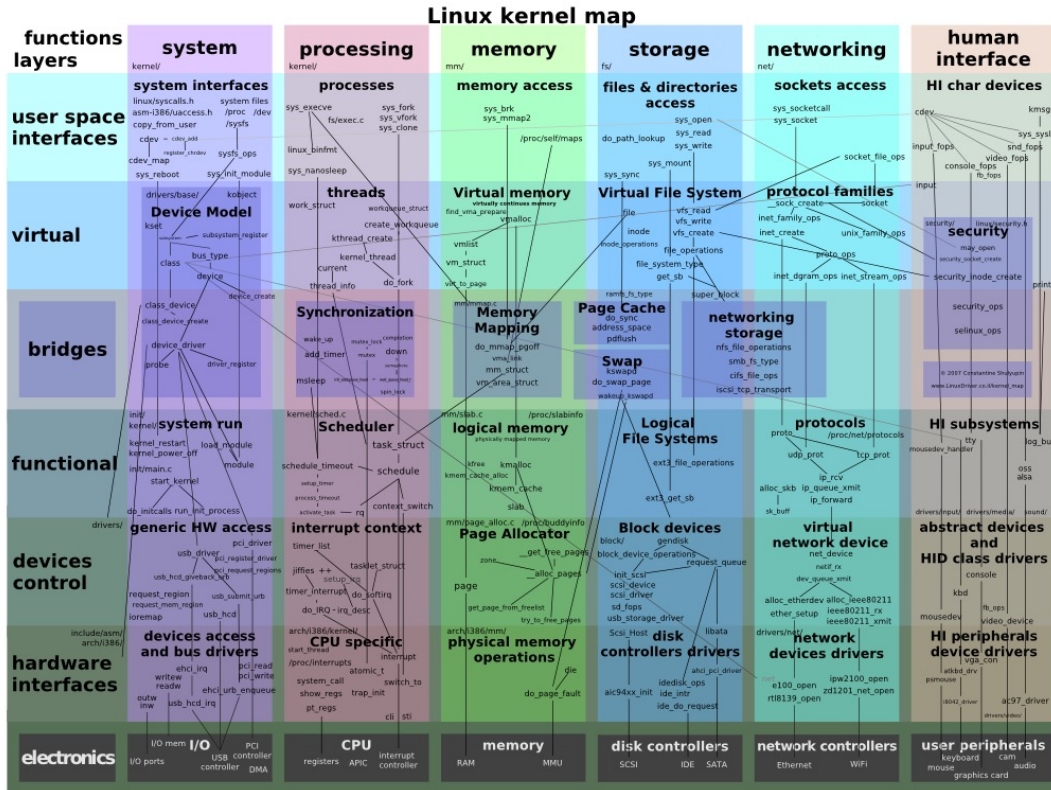
In this course we will primarily focus on  
The kernel and the system libraries.

# The OS Kernel





# The OS Kernel



- Process management
- Memory management
- File-system management
- Security
- Communication and networking
- Time Synchronization
- Many others: users, IPC, network, time, terminals

# How do we interact with the kernel?

- Applications only see them via system calls (system calls are the API of the kernel)
- Examples, from UNIX / Linux:

```
pid_t pid = getpid();  
printf("mypid is %d\n", pid);
```



# How do we interact with the kernel?

- Applications only see them via system calls (system calls are the API of the kernel)
- Examples, from UNIX / Linux:

```
pid_t pid = getpid();  
printf("mypid is %d\n", pid);
```

```
brk(0x18c9000)           = 0x18c9000  
clone(child_stack=0,  
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_  
SETTID|SIGCHLD,  
child_tidptr=0x7fbf4bda1a10) = 3710  
getpid()                 = 3709  
fstat(1, {st_mode=S_IFCHR|0620,  
st_rdev=makedev(136, 6), ...}) = 0  
write(1, "mypid is 3709\n", 14mypid is 3709  
)      = 14  
exit_group(0)
```

Strace output (system calls in bold)





# Why OS design is challenging

- The environment is unforgiving: weird h/w, hard to debug
- It must be efficient (thus low-level?)
  - but abstract/portable (thus high-level?)
- Powerful (thus many features?)
  - but simple (thus a few composable building blocks?)
- Features interact: ``fd = open(); ...; fork()```
- Behaviors interact: CPU priority vs memory allocator
- Open problems: security, multi-core



# Before Next Class

- Ensure that you have full access to your VUIT Amazon AWS Virtual Machine
- Ensure that you can access the CS 3281 GitHub repo, including
  - <https://github.com/cs3281/lectures>
  - <https://github.com/cs3281/lectures/wiki>
- Verify access to Brightspace and Piazza



# Additional Reference Material

- <https://github.com/cs3281/coding-interview-university>
- <https://missing.csail.mit.edu/> - The Missing Semester of Your CS Education
  - Videos and material which will be very useful for you.

