Operating Systems

CS3281 / CS5281

Spring 2026

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

VANDERBILT UNIVERSITY

Team

- Instructor
  - Mohammad Kavousi
- Graduate TAs
  - Weiheng Qiu, Ke Li
- Graders
  - Tobenna Udeze, Jiashu Zhu, Haowei Fu, Alex Galvez

Top Hat

- You will receive an email for your section. Please join.

- Visit https://www.tophat.com and enter the join code:
    - Section 1: 538765
    - Section 2: 121371
    - Section 3: 796689

Important Links

Textbook	http://pages.cs.wisc.edu/~remzi/OSTEP/
zybook	https://learn.zybooks.com/zybook/VANDERBILTCS3281-5281Spring2026
Discussion Forum & Announcements	https://piazza.com/vanderbilt/spring2026/cs32815281
Lectures	https://github.com/cs3281/lectures (PowerPoint lectures will be published on Top Hat)
Programming Assignments	https://classroom.github.com/classrooms/30844110-cs3281-classroom-spring2026
Announcements & Administration	https://vanderbilt.edu/brightspace/

Office Hours

- See on Github.

Programming Assignments

- Administered through GitHub Classroom
  - Requires admission to the `cs3281` repository using your GitHub username
  - Fill the following form to share your GitHub username. Use your **Vanderbilt account** to access the form

https://forms.office.com/r/DcaxBd2iC4

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

Programming Assignments

- 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
  - If you have any issues with logging, send a ticket to VUIT via https://tdx.vanderbilt.edu/TDClient/33/Portal/Home/

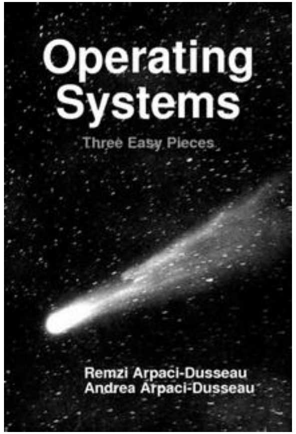
Programming Assignments

- 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“

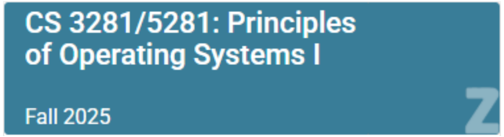
  - Log into your AWS virtual machine
  - Copy Personal Access Token and write it down or save it somewhere
  - In a terminal, run "git clone https://github.com/cs3281/lectures.git"
    - When prompted, enter you GitHub username and enter the above-generated Personal Access Token as your password
    - 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

Textbook

- Textbook: Operating Systems: Three Easy Pieces
  - Available for free at http://pages.cs.wisc.edu/~remzi/OSTEP/



- zybook
  - An online interactive book for understanding the lecture concepts
  - Activities will be assigned regularly. See **Brightspace** for the schedule of assigning the activities
  - Subscription code: VANDERBILTCS3281-5281Spring2026
  - Use your Brightspace name, Vanderbilt email address and VUNet ID



Textbook

- Other possible texts:
  - The Linux Programming Interface
  - Computer Systems: A Programmer’s Perspective
  - Linux Kernel Development

Course Assessment

- Programming assignments: 50%
- Take-home quizzes: 5%
- Exam 1: 20%
- Exam 2: 20%
- Zybook activities: 3%
- Attendance: 2%

Quizzes

- There will be regular quizzes throughout the semester
- You will do the quizzes outside class
- Each quiz covers topics covered after the previous quiz
- Expected to be conducted on Brightspace

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 programing 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 <u>not be accepted</u>
- **No collaborations unless explicitly permitted**

Late days

- Assignments submitted on or after 12:00am the next day are required to use late days or take the late-day penalty, 20% per day

- Do not wait until 11:59pm to make your first push!

- You may commit and push incremental results and are suggested to do so early and often

- The last push before the deadline will be graded unless specified with late_days.md

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
- You can use ChatGPT and other AI tools however you want! Make sure to disclose how you used an AI tool
- ChatGPT can be a helpful aid, but it can’t totally do your homework for you! Sometimes, it can lead you off track, so think critically about its outputs

Tel (615) 343-7472 | Fax (615) 343-7440
1025 16th Avenue South Nashville, TN 37212
www.isis.vanderbilt.edu
VANDERBILT UNIVERSITY

Course expectations

- You are expected to attend class and participate in class discussions

- All assignments are released in GitHub Classroom. You will access the assignments via Brightspace

Course expectations: Office Hours

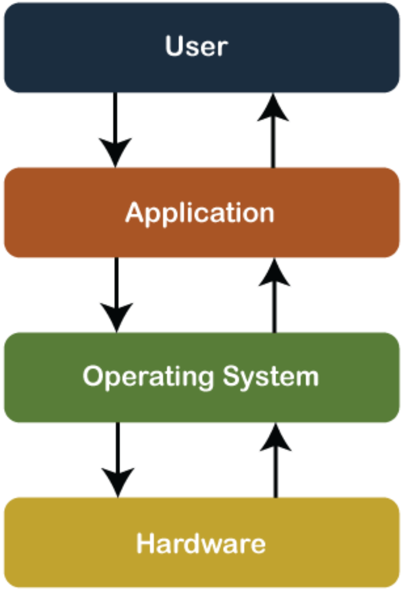
- We will use an office-hour policy similar to the one listed here: https://www2.seas.gwu.edu/~gparmer/resources/2021-09-20-Office-Hours-HOWTO.html
  - Please read this policy carefully and adhere to its guidelines
  - It will help you and us

Development Environment

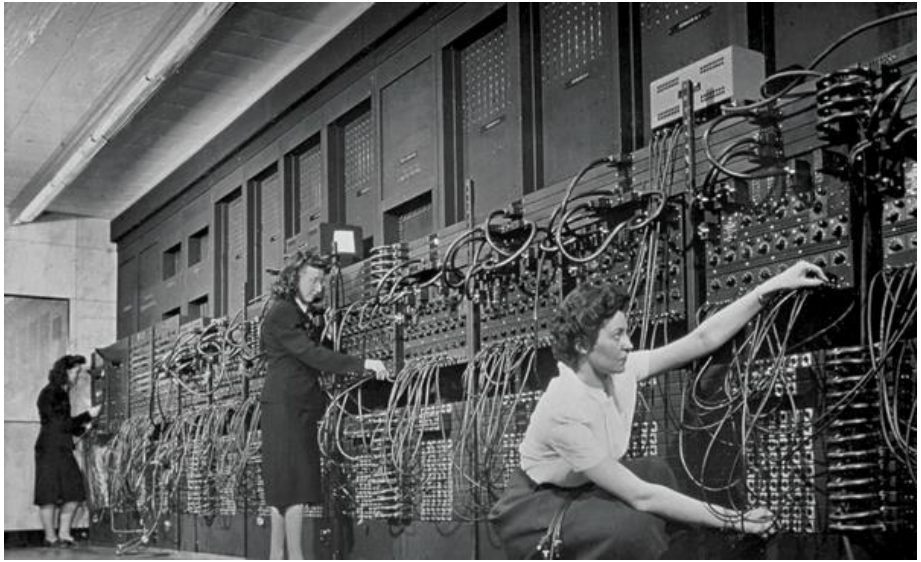
- This course will use C
  - This is an OS course, not a course on C
  - While some C material will be discussed, it is up to you to learn and build your proficiency in C if you aren't proficient already
    - Remember, you can ask ChatGPT for help!
- We will use Ubuntu (a Linux distribution) as the 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

Course Goals

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



Historical Perspective



Eniac (Electronic Numerical Integrator and Computer)– World's first programmable general-purpose computer

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

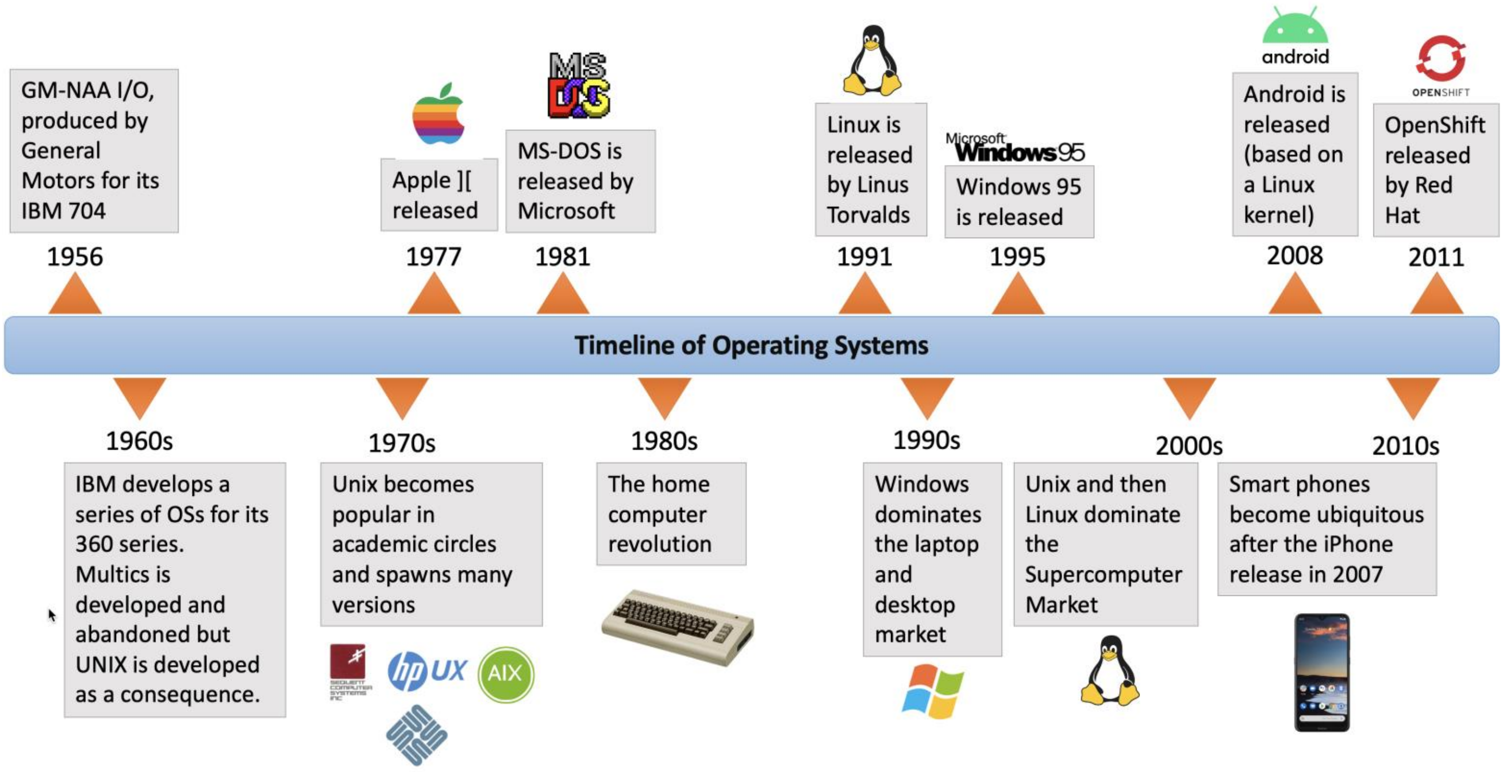
Before operating systems

- User could only run one program at a time.
- Had to insert the program disk before booting the machine.
- Program had to control the hardware directly
- An example (at right): 1983 "King's Quest" game for IBM PC Jr.

Embedded systems often run without OSes

- "Bare-metal" embedded systems
- Application must handle:
  - Boot, initialization and hardware
- Applications are not portable
  - Rewrite, mostly from scratch, for a new "Oven Controller"
- No memory allocations, no coding "errors"
  - Instead, invalid memory accesses would crash the whole system

History of Operating Systems – a brief timeline

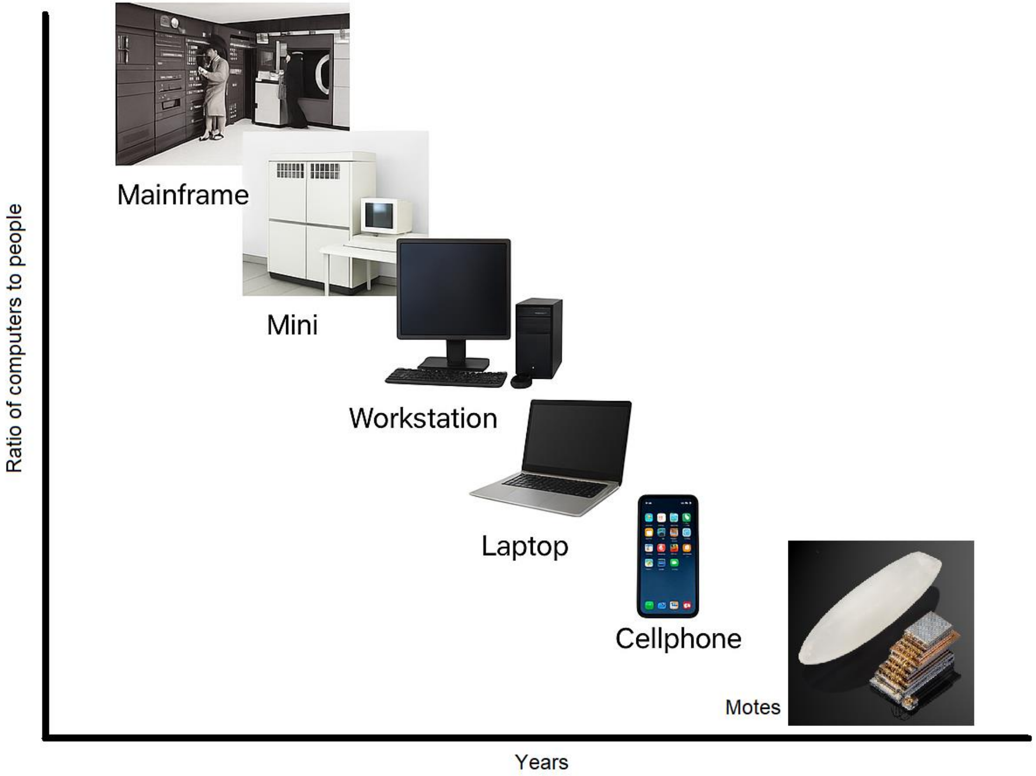


https://medium.com/@tanalpha-aditya/introduction-to-operating-systems-and-networks-dfa3611befcc

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

VANDERBILT UNIVERSITY

Computers come in diversity



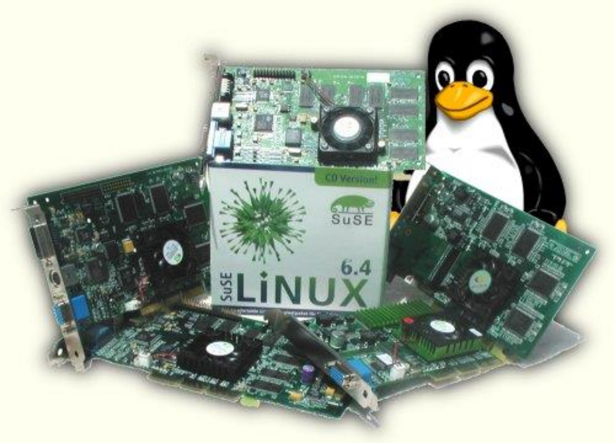
Computing timescales are increasingly large

Jeff Dean (Google AI): "Numbers Everyone Should Know"

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from disk	20,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

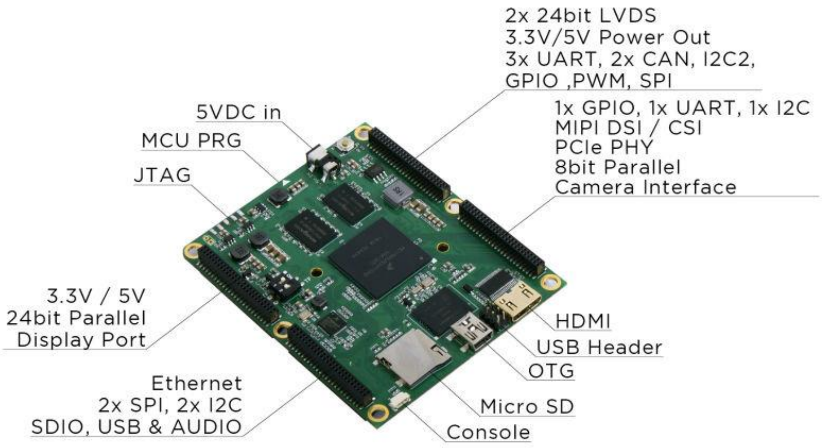
OSes at the heart of these challenges

- Operating system is a software that runs on the bare hardware of a computer
- Operating system is responsible for
  - Abstracting the hardware details for convenience and portability

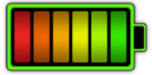


OSes at the heart of these challenges

- Operating system is responsible for
  - Abstracting the hardware details for convenience and portability
  - Share hardware among multiple applications. Through **virtualization**, OS creates multiple *virtual* objects from a real object e.g., CPU and memory








OSes at the heart of these challenges

- Operating system is responsible for
  - Abstracting the hardware details for convenience and portability
  - Share hardware among multiple applications
  - Isolating applications to contain bugs, enforce fair access

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

VANDERBILT UNIVERSITY

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.

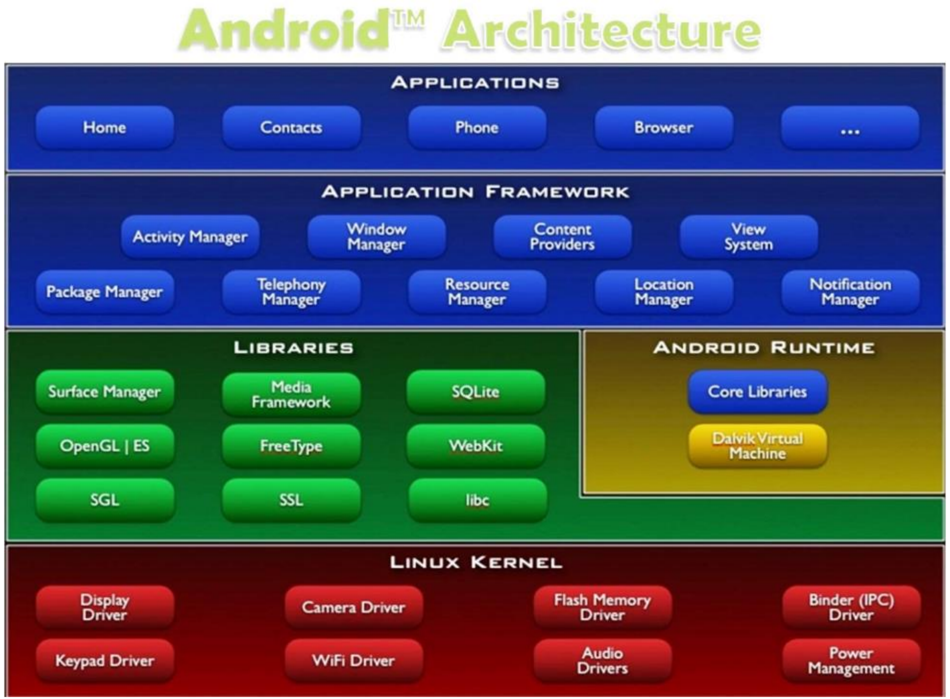
Tel (615) 343-7472 | Fax (615) 343-7440
1025 16th Avenue South Nashville, TN 37212
www.isis.vanderbilt.edu

What’s part of the Operating System?

- **OS Kernel**: the only code without security restrictions
- Process scheduling
- Memory Allocation
- Access hardware devices

- **OS Distribution**: the kernel + other stuff
- GUI
- Command shell
- Package manager
- Software Libraries
- Apps
  - – Text editor, browser, SSH, etc.

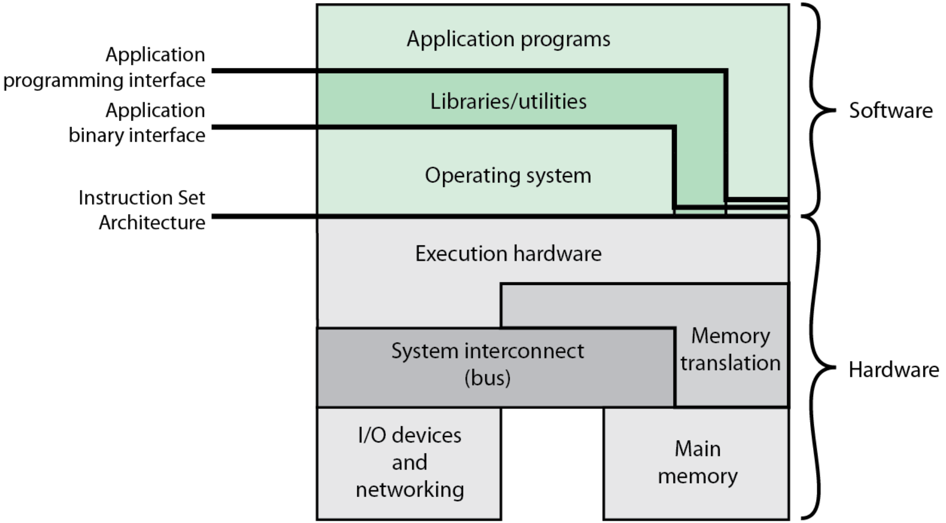
Layers of a modern computing system



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

Layers of a modern computing system

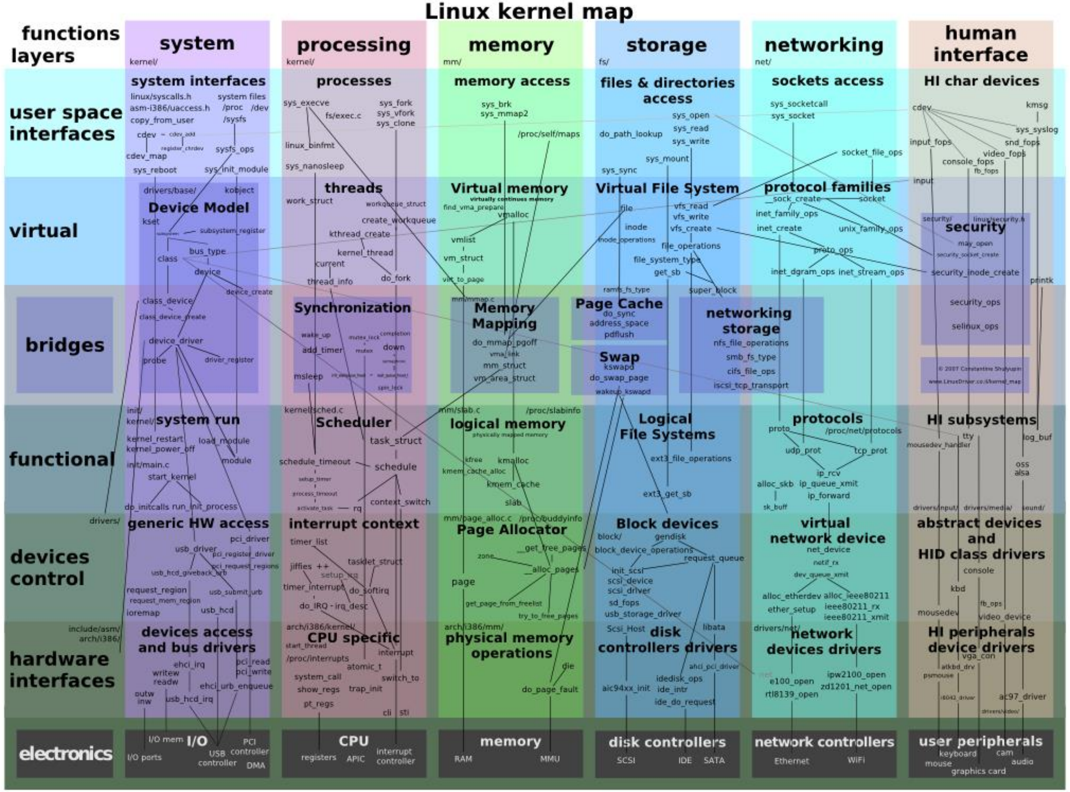
- Application – the user program.
- System Libraries – the core services of OS are encapsulated by these libraries, e.g. libc
- The operating system has full access to hardware

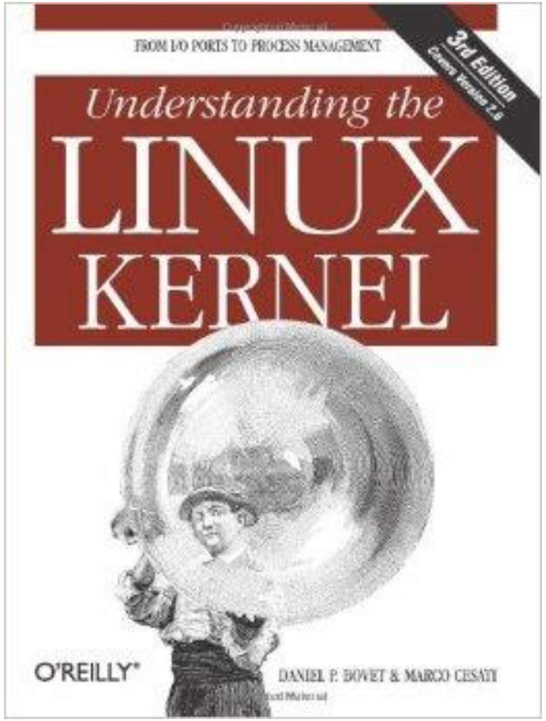


Another view of the layers

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

The OS Kernel





The OS as a Resource Manager

- All applications should have good performance. They should have fast response time for interactive tasks.

- One of the main tasks of an OS is to optimize the use of computational resources. The OS strives to keep CPU, memory, and I/O devices busy continuously

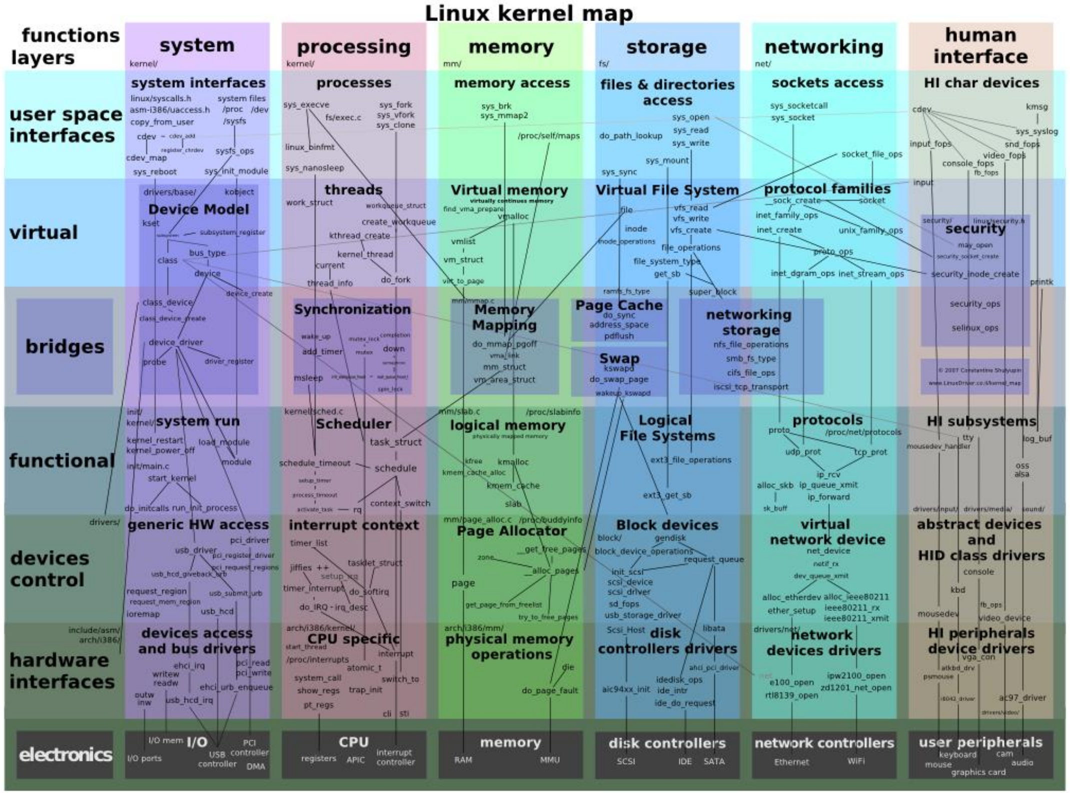
Real-world roles of the Operating System

- Referee
  - Manage protection, isolation, and sharing of resources
  - Resource allocation and communication

- Illusionist
  - Provide clean, easy-to-use abstractions of physical resources
    - Infinite memory, dedicated machine
    - Higher level objects: files, users, messages
    - Masking limitations, virtualization

- Glue
  - Common services
    - Storage, Window system, Networking
  - Sharing, Authorization
  - Look and feel

The OS Kernel



- Process management
- Memory management
- Scheduling
- File-system management
- Communication and networking
- 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)

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

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?)
- Resource sharing: CPU and memory
- Open problems: security

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. Skim through the syllabus: https://github.com/cs3281/lectures
- Verify access to Brightspace and Piazza

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