Linux Kernel Programming

Xiaoguang Wang

UIC COMPUTER SCIENCE

About me

Assistant Professor @ UIC CS

Email: xgwang9@uic.edu (preferred way to communicate)

Office: SEO 1331

Homepage: https://xiaoguang.wang/

My research interests

Systems software (operating systems, compiler, runtime, …)

- How to efficiently execute programs on CPUs of heterogeneous architecture (e.g., SmartNIC, IoT/Edge)?

Software security

- How to leverage LLMs (e.g., ChatGPT) to better discover unknown bugs?
- Can we *automatically* `detect` and `fix` bugs?
- How to compartmentalize large-scale software to prevent bugs from spreading?

What is the Linux Kernel?

One of operating system kernels

- e.g., Windows, FreeBSD, OSX, etc.

What does an OS do for you?

- **Abstract** the hardware for convenience and portability
- **Multiplex** the hardware among multiple applications
- **Isolate** applications to contain bugs
- Allow **sharing** among applications

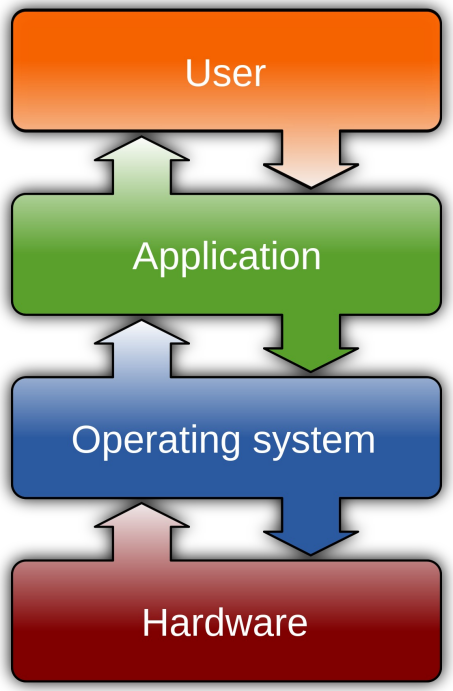
View: layered organization

User: applications (e.g., `vim` and `gcc`)

Kernel: file system, process, etc.

Hardware: CPU, mem, disk, etc.

→ Interface between layers



View: core services

- Process
- Memory
- File contents
- Directories and file names
- Security
- Many others: users, IPC, network, time, terminals, etc.

→ Abstraction for applications

Example: system calls

Interface: applications talk to an OS via system calls

Abstraction: process and file descriptor

```
fd = open("out", 1);
write(fd, "hello\n", 6);
pid = fork();
```

Why is Linux kernel interesting?

OS design deals with conflicting goals and trade-offs

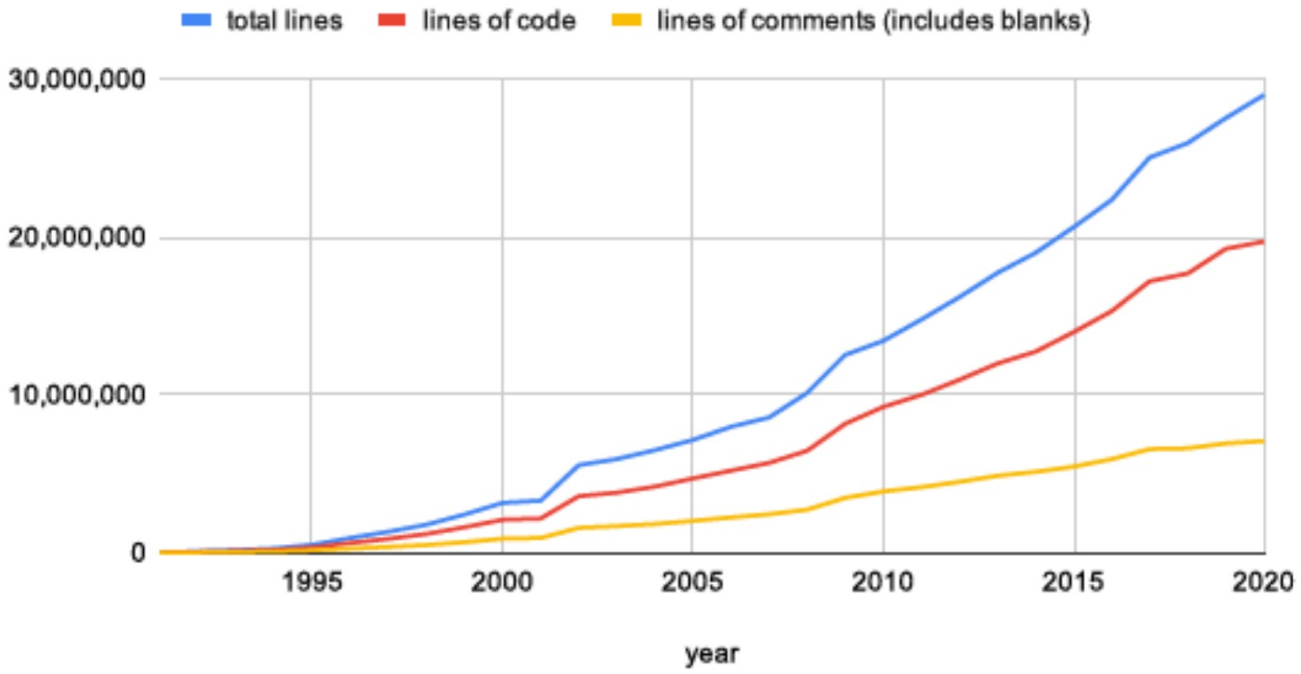
- Efficient yet portable
- Powerful yet simple
- Isolated yet interactable
- General yet performant

Open problems: multi-core (scalability), performance, heterogeneous devices, security, …

How does a state-of-the-art OS deal with above issues?

- Hack the Linux kernel!

Why is Linux kernel interesting?



Extremely large software project

- more than 30 ~~25~~ million lines of code
- 7,500 ~~4,600~~ lines of code are added every day!

https://www.linuxfoundation.org/resources/publications/linux-kernel-history-report-2020

Why is Linux kernel interesting?

Very fast development cycles

- release about every 70 days
- 13,000 patches / release

One of the most well-written/designed/maintained C code projects

More here:

- Linux Foundation Annual Report 2021

Linux is eating the World

71% of smartphones and tablets run Linux (Android), as of Dec/2023

- iOS: 28%

98% of top 1 million web servers run Linux

99% of super computers run Linux

SpaceX: From Earth to orbit with Linux and SpaceX

Ref: Use share of operating systems

It is good for your job search

Contributions from unpaid developers had been in slow decline

- 14.6% (2012) → 13.6% (2013) → 11.8% (2014) → 7.7% (2015)

Why?

- “There are many possible reasons for this decline, but, arguably, the most plausible of those is quite simple: **Kernel developers are in short supply**, so **anybody who demonstrates an ability to get code into the mainline tends not to have trouble finding job offers**.”

https://www.linuxfoundation.org/resources/publications/linux-kernel-history-report-2020

Who should take this course?

Anyone wants to work on the above problems

Anyone cares about what's going on under the hood

Anyone has to build high-performance systems

Anyone needs to diagnose bugs or security problems

…

About this course

CS 594: Linux Kernel Programming (special topics)

- A little different from a traditional OS course
- Avoid teaching some important but not commonly used OS concepts
  - E.g., boot-loading, enable paging, page table creation, …
- How to more confidently program inside the Linux kernel

About this course

CS 594: Linux Kernel Programming (special topics)

- A little different from a traditional OS course

Goals

- Understand core subsystems of the Linux kernel in depth
- Design, implement, and modify Linux kernel code and modules for these subsystems
- Test, debug and evaluate the performance of systems software in kernel or user space, using debugging, monitoring and tracing tools

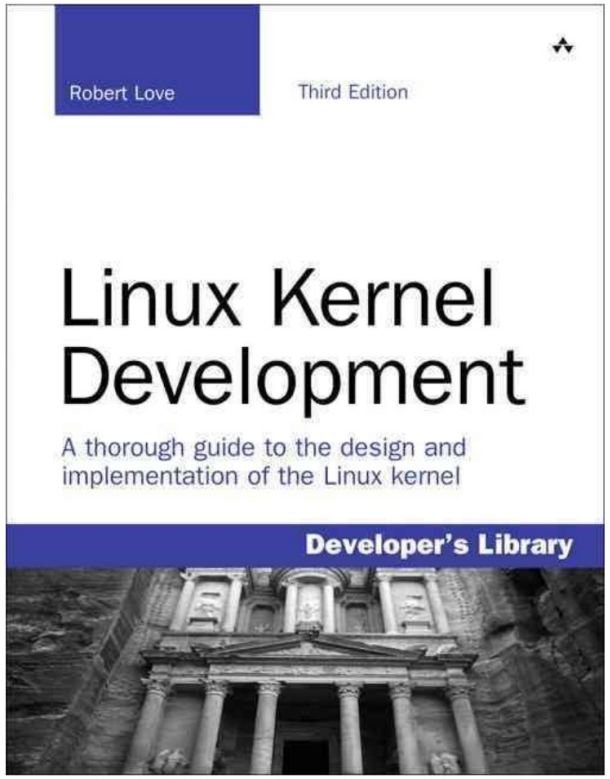
Prerequisite

Required skills:

- C programming and Linux command line (required)
- Basic knowledge of computer architecture, operating systems, algorithms, and data structures (highly recommended)

Textbook

Robert Love, Linux Kernel Development, Addison-Wesley

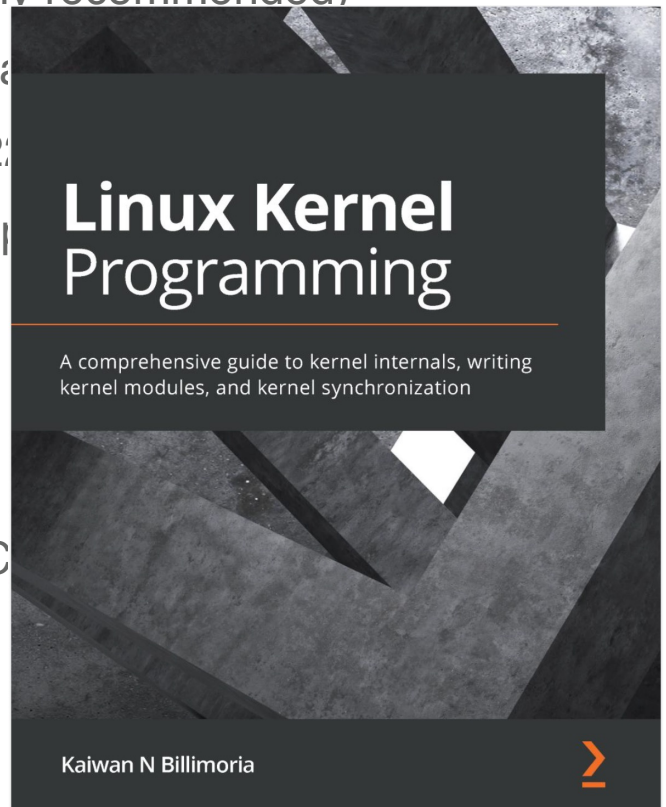


Other useful resources

- Billimoria, Kaiwan N. Linux Kernel Programming: A comprehensive guide to kernel internals, writing kernel modules, and kernel synchronization. Packt Publishing Ltd, 2021. (highly recommended)
- Billimoria, Kaiwan N. Linux Kernel Debugging: Leverage proven tools and advanced techniques to effectively debug Linux kernels and kernel modules. Packt Publishing Ltd, 2022.
- Bovet, Daniel P., and Marco Cesati. Understanding the Linux Kernel: from I/O ports to process management (3rd Edition). O'Reilly Media, Inc., 2005.
- Corbet, Jonathan, Alessandro Rubini, and Greg Kroah-Hartman. Linux Device Drivers (3rd Edition). O'Reilly Media, Inc., 2005.
- Love, Robert. Linux System Programming: Talking Directly to the Kernel and C Library (2nd Edition). O'Reilly Media, Inc., 2013.

Other useful resources

- Billimoria, Kaiwan N. Linux Kernel Programming: A comprehensive guide to kernel internals, writing kernel modules, and kernel synchronization. Packt Publishing Ltd, 2021. (highly recommended)
- Billimoria, Kaiwan N. Linux Kernel Debugging: Leverage proven tools and adva
effectively debug Linux kernels and kernel modules. Packt Publishing Ltd, 202
- Bovet, Daniel P., and Marco Cesati. Understanding the Linux Kernel: from I/O p
management (3rd Edition). O'Reilly Media, Inc., 2005.
- Corbet, Jonathan, Alessandro Rubini, and Greg Kroah-Hartman. Linux Device
O'Reilly Media, Inc., 2005.
- Love, Robert. Linux System Programming: Talking Directly to the Kernel and C
Edition). O'Reilly Media, Inc., 2013.



Communication

Lecture

- Location: TBH 180E
- Time: TR 9:30am – 10:45am
- Recorded video will be uploaded to Blackboard
  - Strongly encourage you to join in person

Office Hours

- Thursday 3:30 PM - 4:30 PM, SEO 1331
- Or by appointment: https://calendly.com/xgwang9/15-min-office-hour

Communication

Blackboard

- https://uic.blackboard.com/ultra/courses/_266043_1/cl/outline
- Syllabus, slides, recorded videos, etc.
- Exercise, project submission
- Grades posted

Grading policy

Quizzes and assignments [45%]

- Including paper reading

Midterm exam [15%]

- No final exam

Final research project [30%]

Class participation and discussion [10%]

Policy for Missed or Late Work

- Late submission (0, 24 hours] will be accepted with a 15% penalty;
- Late submission (24-48 hours] will be accepted with a 30% penalty;
- Late submission beyond 48 hours will not be accepted.

A: >=90, B: [80,90), C: [70, 80), D: [60, 70), F: <60

If you clearly put significant effort into homework, class discussion, the paper(s) to read, and your project, you'll get an A.

About projects (subject to change)

Tentative topics (maybe only choose 3):

- Adding new system calls
- Kernel module – data structure handling
- Handling page faults from the user-space with `userfaultfd`
- An OS virtualization lab – virtualize a simple random number generator (RNG)

Final research project

- TBD (semester long research project, group of ~2)
- Target high (aim to make a paper submission to SOSP/OSDI/ATC/APSys)

About paper reading

Paper reading with a report

- help you understand a topic better
  - E.g., FlexSC OSDI'10, system call patch processing

Paper reading at the end of the semester (~ 3 weeks)

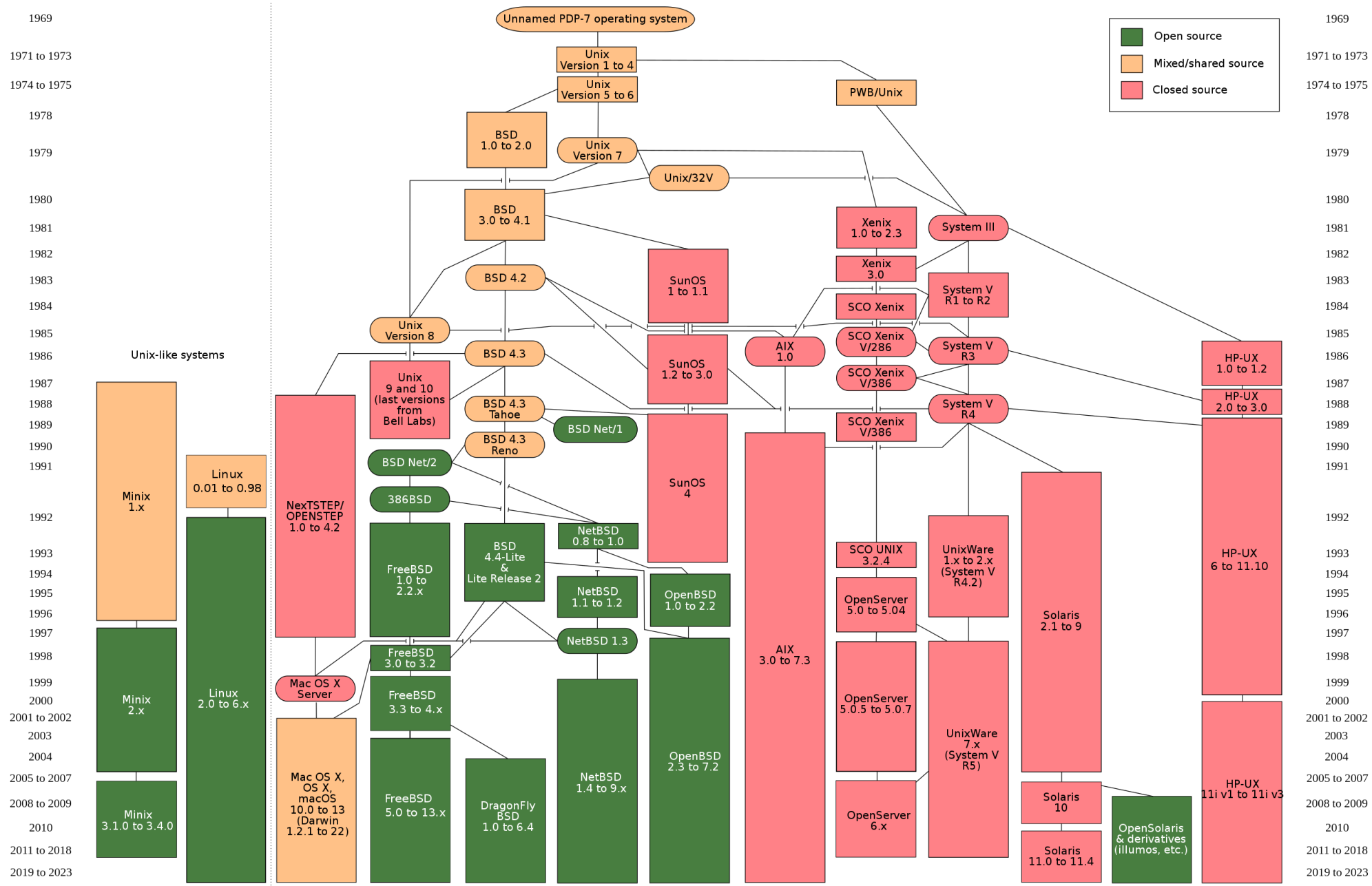
- everyone read and present 1~2 paper from OSDI / SOSP / ATC / EuroSys / ASPLOS 2023 / 2024

Acknowledgement

Part of the teaching materials are from an ECE-4414/5414G(CS-4224/5264G) course from Virginia Tech provided by Dr. Changwoo Min and Dr. Pierre Olivier.

Today's agenda

- The history of Linux
- Linux open-source model and community
- High level overview of the Linux kernel



History of UNIX: https://en.wikipedia.org/wiki/History_of_Unix

UIC COMPUTER SCIENCE

Beginning of Linux

```
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu)
for 386(486) AT clones. This has been brewing since april, and is starting to get ready.
I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat
(same physical layout of the file-system (due to practical reasons) among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that
I'll get something practical within a few months, and Id like to know what features most
people would want. Any suggestions are welcome, but I won't promise I'll implement them 🙂

Linus (torvalds@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT protable
(uses 386 task switching etc), and it probably never will support anything other than
AT-harddisks, as that's all I have :-(.
```

Linux history

1991: First apparition, author Linux Torvalds

1992: GPL License, first Linux distribution

1994: v1.0 – Single CPU for i386, then ported to Alpha, MIPS

1996: v2.0 – Symmetric multiprocessing (SMP) support

1999: v2.2 – Big kernel lock removed

2003: v2.6 – Physical address expansion (PAE), new architectures, …

2011: v3.0 – Incremental release of v2.6

2015: v4.0 – Livepatch

→ today's latest version: https://kernel.org/

Linux open-source model

Linux is licensed under GPLv2

"*You may copy, distribute and modify the software as long as you track changes/dates in source files. Any modifications to or software including (via compiler) GPL-licensed code must also be made available under the GPL along with build & install instructions*."

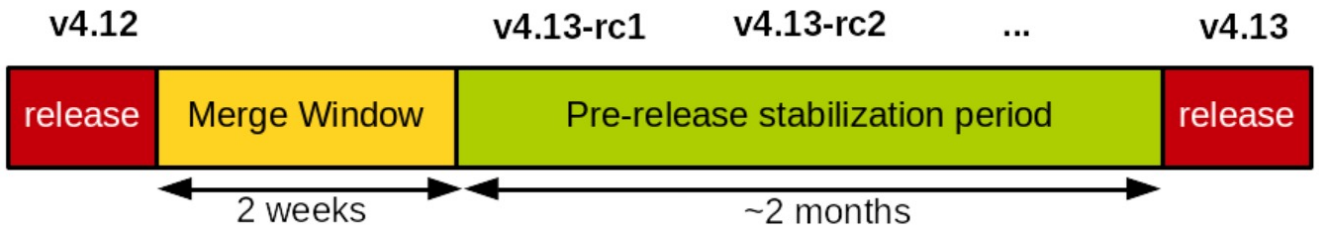
Benefit of open-source model

Given enough eyeballs, all bus are shallow

Given a large enough tests, almost every problem will be characterized quickly and the fix obvious to someone.

Kernel release cycle

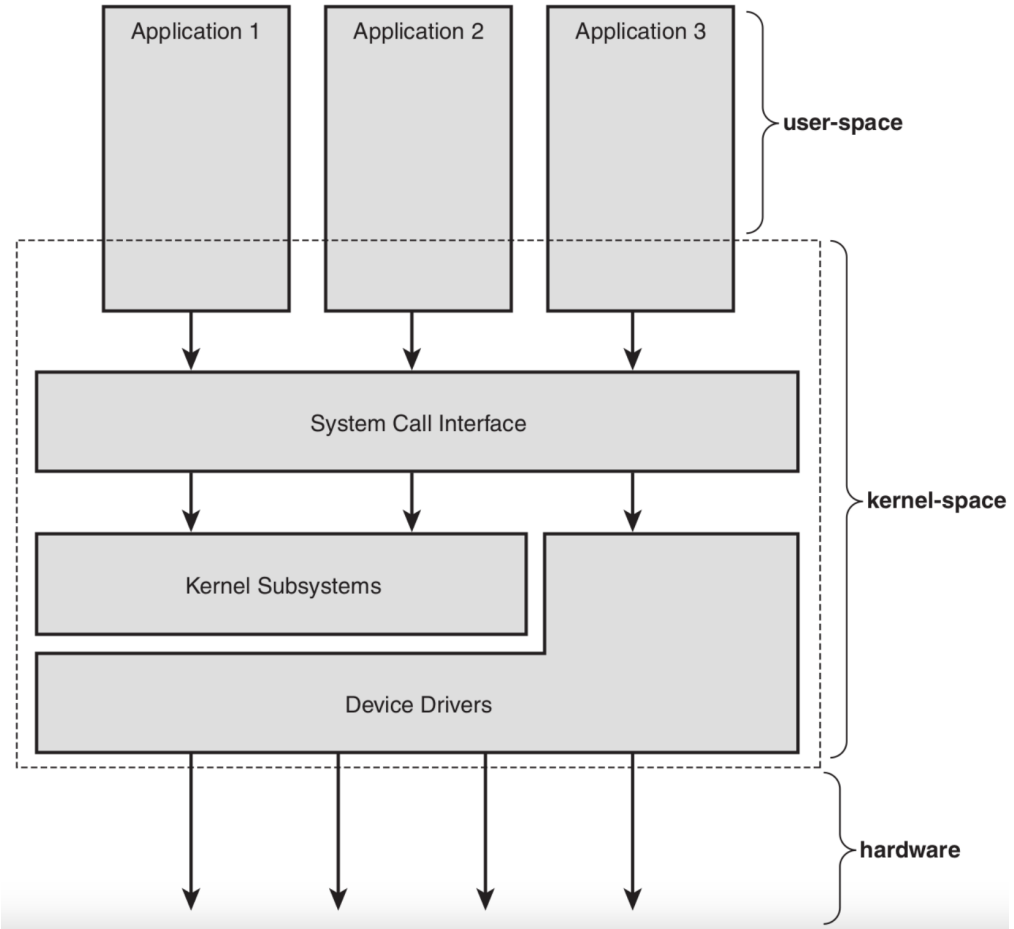
(major).(minor).(stable) → E.g., 6.1.71



- "RC" (Release Candidate) → testing before the mainline release
- Mainline release → maintained by Linus with all new features
- Stable release → additional bug fixes after the mainline kernel release
- Long term support (LTS) for a subset of releases → e.g., 6.1, 5.15, …

https://kernel.org/

Overview of operating systems



User space v.s. kernel space

A CPU is executing in either of **user space** or **kernel space**

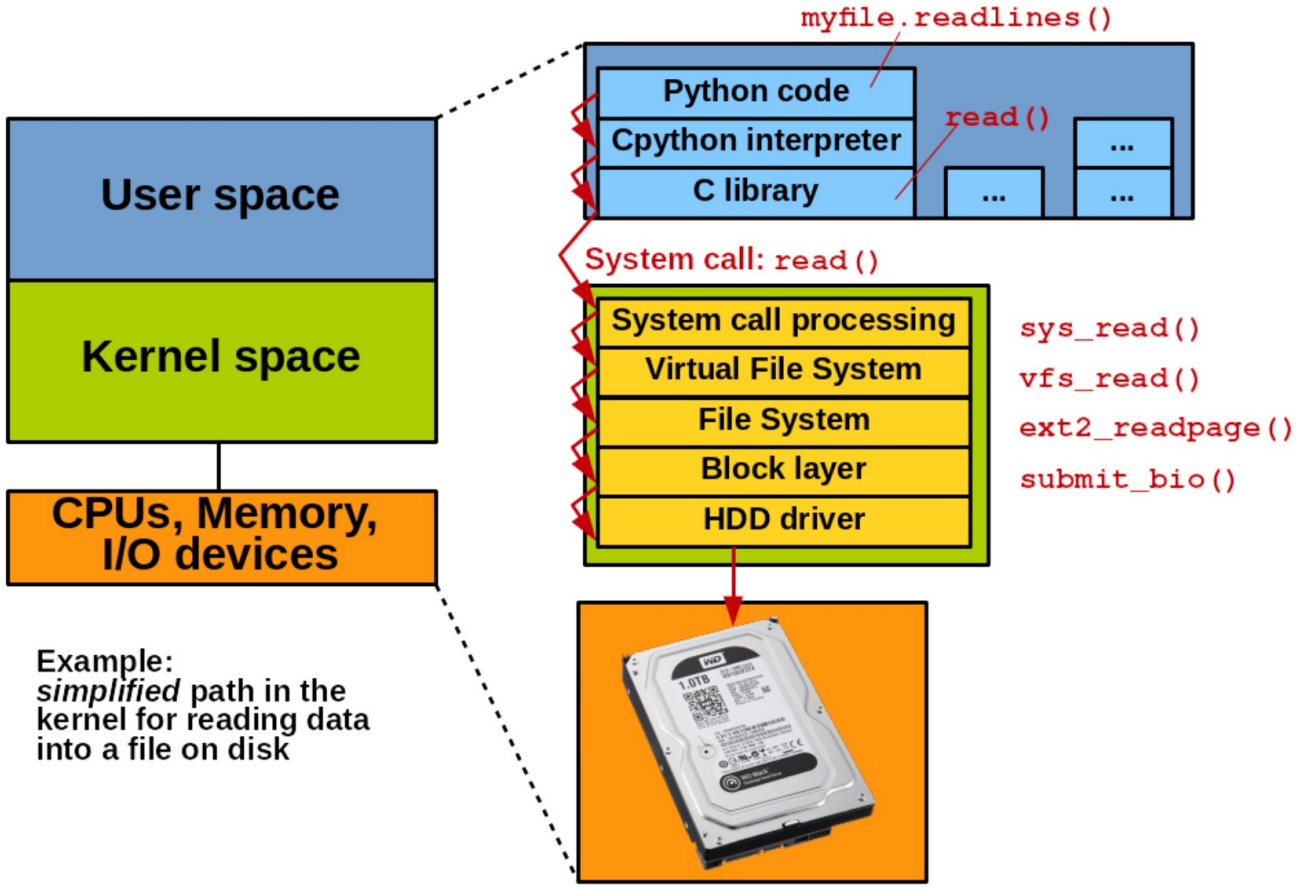
Only the kernel is allowed to perform **privileged operations** such as controlling CPU and IO devices

- E.g., protection rings in x86 architecture
- ring 3: user-space application; ring 0: OS kernel

A user-space application talks to the kernel through the **system call** interface

- E.g., `open()`, `read()`, `write()`, `close()`

User space v.s. kernel space



Linux is a monolithic kernel

A traditional design: all of the OS runs in kernel, privileged mode

- share the same address space

Kernel interface ~= system call interface

Good: easy for subsystems to cooperate

- one cache shared by file system and virtual memory

Bad: leads to bugs, no isolation within kernel

Alternative: micro-kernel design

Many OS services run as ordinary user programs

- e.g., file system in a file server

Kernel implements minimal mechanism to run services in user space

- IPC, virtual memory, threads

Kernel interface != system call interface

- applications talk to servers via IPCs

Good: more isolation

Bad: IPCs may be slow

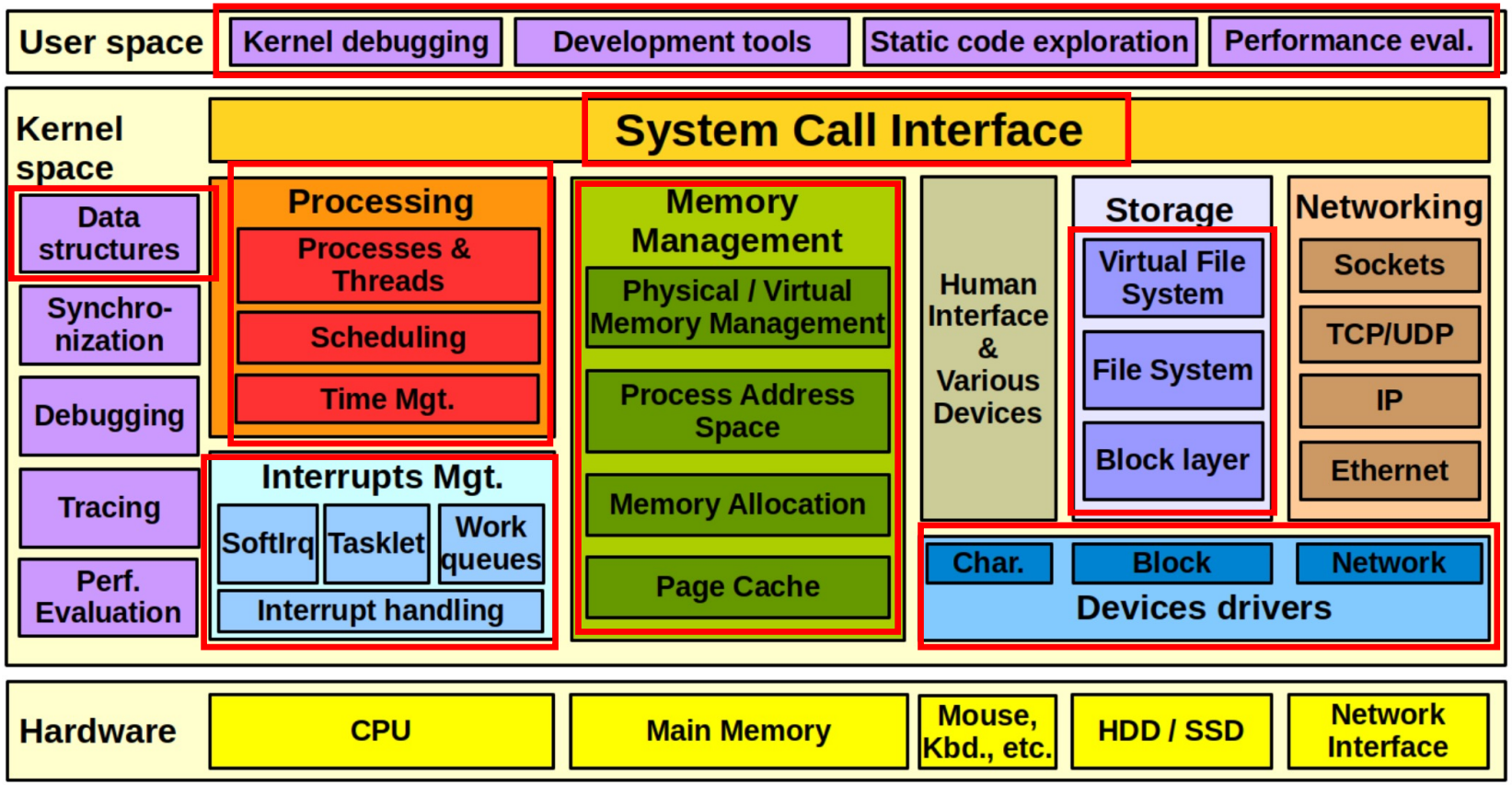
Debate

Tanenbaum-Torvalds debate

Most real-world kernels are mixed: Linux, OS X, Windows

- e.g., X Window system

Kernel & course map



Set up course environment

VirtualBox to run Linux VM

- Recommended setting
  - disk >= 64GB, RAM >= 4GB, # CPU >= 2
- Add port forwarding rule
  - protocol: TCP, host IP: 127.0.0.1
  - host port: 2222, guest port: 22

Cloudlab

- https://cloudlab.us/

Set up course environment

Ubuntu 22.04 server for Linux distribution

- Add your account as a `sudo` user

```
sudo usermod -aG sudo xiaoguang
```

- Tricks to avoid typing password every time

```
sudo visudo
xiaoguang ALL=(ALL) NOPASSWD:ALL
```

Linux kernel: v6.1 released on Dec 12th, 2022

Next steps

Finish to set up course environment

- Bring your laptop

Take the Readiness Exercise (hw1)

- Due Friday!

If you are not familiar with Linux commands, learn followings:

- `vim`, `ssh`, `scp`, `tmux`, and more

Download the latest Linux kernel source inside your Linux VM

- `$ git clone https://github.com/torvalds/linux.git`

Next lecture

Build and explore Linux kernel

Survey