CS 5264/4224; ECE 5414/4414
(Advanced) Linux Kernel Programming
Lecture 1

Introduction to Linux Kernel

January 21st, 2025

Huaicheng Li

Slides courtesy of Prof. John Kubiatowicz, CS 162 @ UCB

Getting to Know Each Other!

- Who am I?
  - PhD (UChicago, ‘15-’20)
  - Postdoc (CMU, ‘20-’22)
  - Assistant Professor (VT, Fall ‘22)
  - https://people.cs.vt.edu/huaicheng
  - Office: 4109 Gilbert Place (GP)
  - huaicheng@cs.vt.edu; huaicheng@vt.edu
- What do I do?
  - Research
  - Teaching: mentoring and classes
- Interests: Operating Systems, Storage, Memory, Architecture

What is the Linux Kernel

- One of the operating system kernels
  - e.g., Windows, FreeBSD, MacOS, 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, gcc)
- Kernel: file systems, process, etc.
- Hardware: CPU, memory, network, disk, GPU, etc.

Providing interface between layers ....

View: Core Services

- Processes
- Memory management
- Files (systems)
- Security
- Networking
- among many others: users, IPC, time, various drivers, etc.

Providing abstractions for applications

Example: System Calls

- Interface: applications talk to an OS via system calls
- Abstraction: process and file descriptors

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

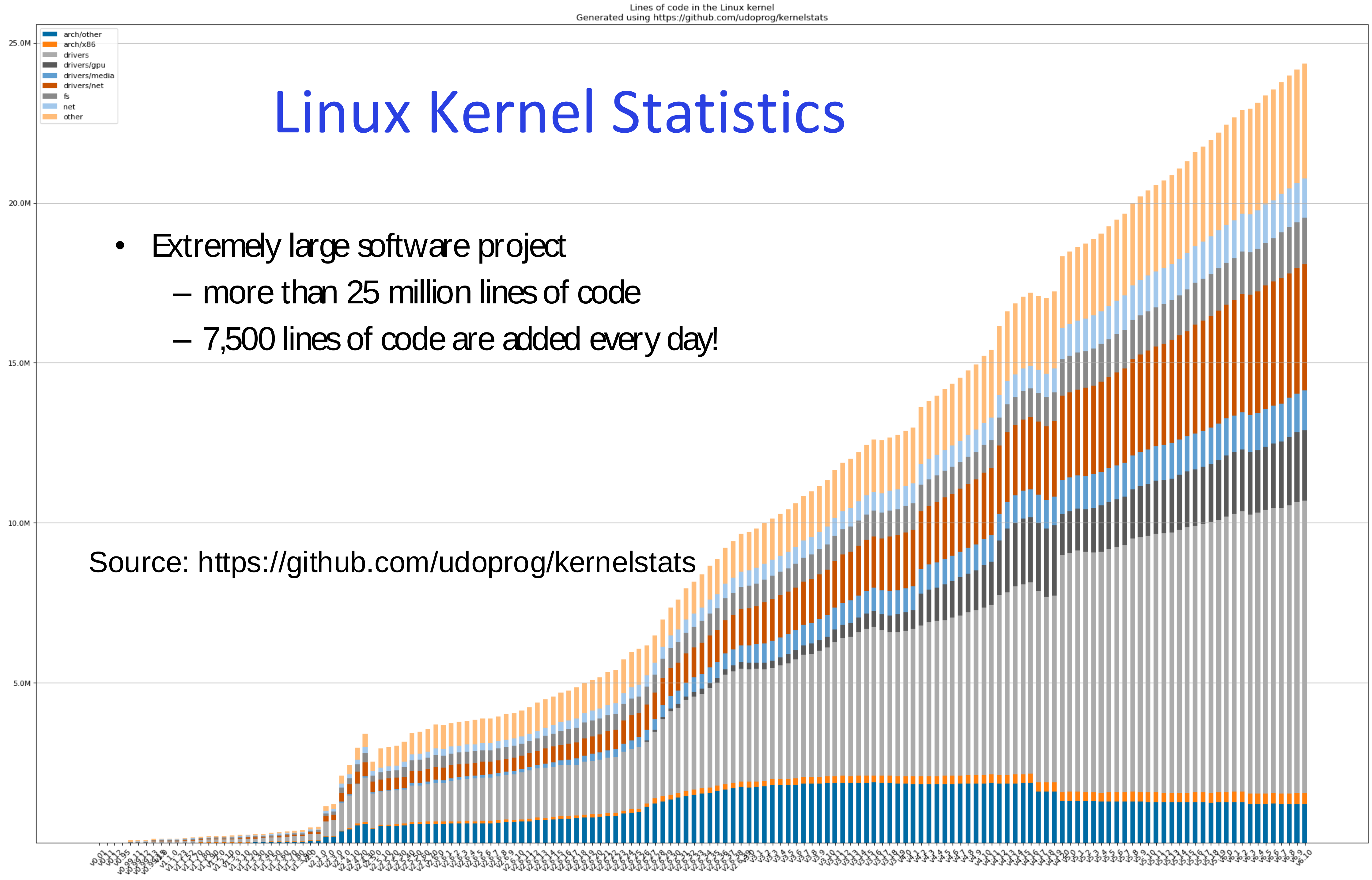
Why is Linux Kernel Interesting?

- 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 and security
- How does a state-of-the-art OS deal with above issues?
  - Hack the Linux kernel!

Linux Kernel Statistics

- Extremely large software project
  - more than 25 million lines of code
  - 7,500 lines of code are added every day!

Source: https://github.com/udoprog/kernelstats



Why is Linux Kernel Interesting?

- Very fast development cycles
  - release about every 70 days
  - 13,000 patches / release
  - 273 companies / release (or 1,600 developers / release)
- One of the most well-written/designed/maintained C code
- More here
  - Linux Foundation Kernel Report 2017
  - Linux Foundation Annual Report 2021

Linux Rules the World

- 85.1% of smartphones and tables run Linux (Android)
  - iOS: 14.9%
- 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: Usage share of OS

Useful for 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**."

Source: Linux Foundation Kernel Report 2017

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

Goals of This Course (LKP)

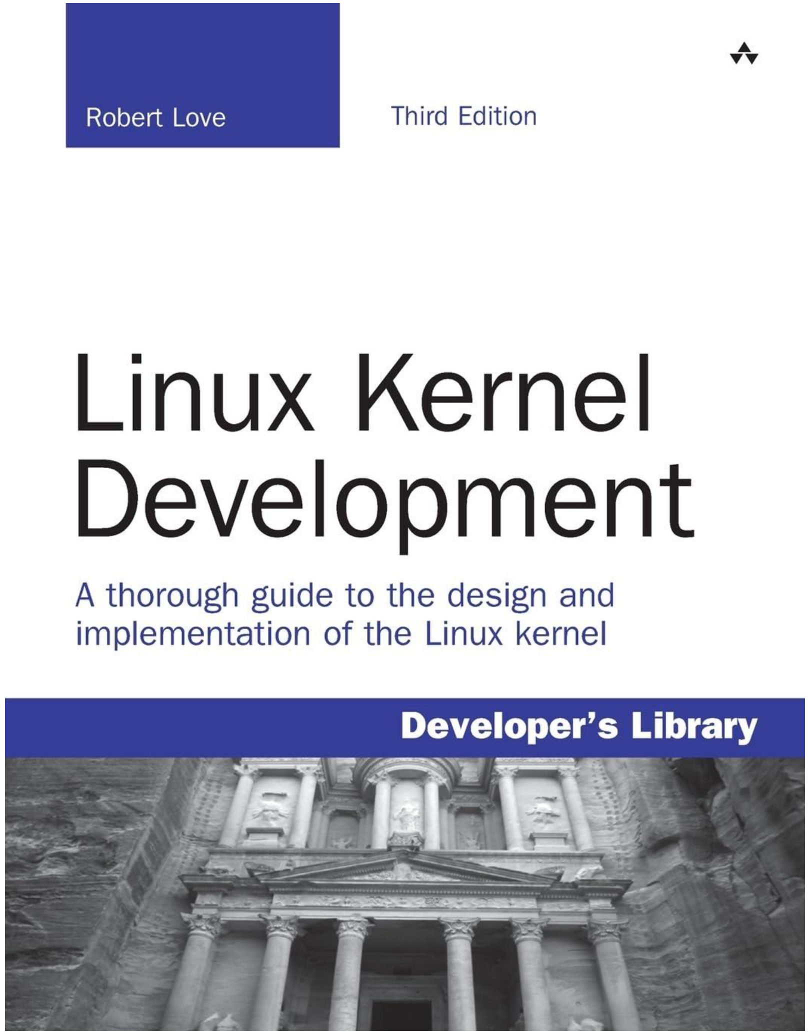
- **Understand** core subsystems of the Linux kernel in depth
- **Design, implement, and modify** Linux kernel code and modules for these subsystems
- **Test, debug, and evaluate** the performance of systems software in kernel or user space, using debugging, monitoring and tracing tools

Prerequisite

- Undergraduate and graduate students
  - C programming (strict)
  - Linux command line (strict)
  - Computer architecture and operating system (recommended)
- Undergraduate students
  - ECE 3574 (Applied Software Design) or CS 3114 (Data Structures)
  - CS 3214 (Computer Systems)

Textbooks

- Robert Love, Linux Kernel Development (3rd edition), Addison-Wesley



Robert Love

Third Edition

Linux Kernel Development

A thorough guide to the design and implementation of the Linux kernel

Developer's Library

Many Other Useful Resources

- Understanding the Linux Kernel, O'Reilly Media
- Professional Linux Kernel Architecture, Wrox
- Linux Device Drivers, O’Reilly Media
- Understanding Linux Network Internals, O’Reilly Media
- Operating Systems: Three Easy Pieces
- Intel 64 and IA-32 Architectures Software Developer Manuals

Logistics

- Lectures: TR 3:30-4:45pm, WMS 120
  - Regular lectures + Paper discussion + Guest lectures (TBD)
  - Instructor office hour: Fridays 11-12am, GP 4109, or by appointment
  - No recordings
  - Attendance is mandatory
  - Ask questions
- TA: Ezekiel Cochran, ecochran@vt.edu
  - Office hours: TBA (ex, projects, lectures)
- Course
  - Website: https://people.cs.vt.edu/huaicheng/lkp-sp25/
    - schedule, homework/project instructions, pointers to materials, etc.
- Canvas:
  - Will publish it soon, mainly used for hosting quiz, exercises, notes, slides, projects, etc.
- Ed Discussion: https://edstem.org/us/join/eVsySF
  - Announcements, Q/As, etc, social, ...

Grading

- Participation (5%)
- Exercise (6%)
  - 2% x 3 exercises
- Paper reading (15%)
  - 3% x 5 papers
- Projects (64%)
  - 2 small projects: 4% + 10%
  - 1 medium project: 20%
  - 1 final project: 30%
- Final exam (10%)
- Bonus (5%)

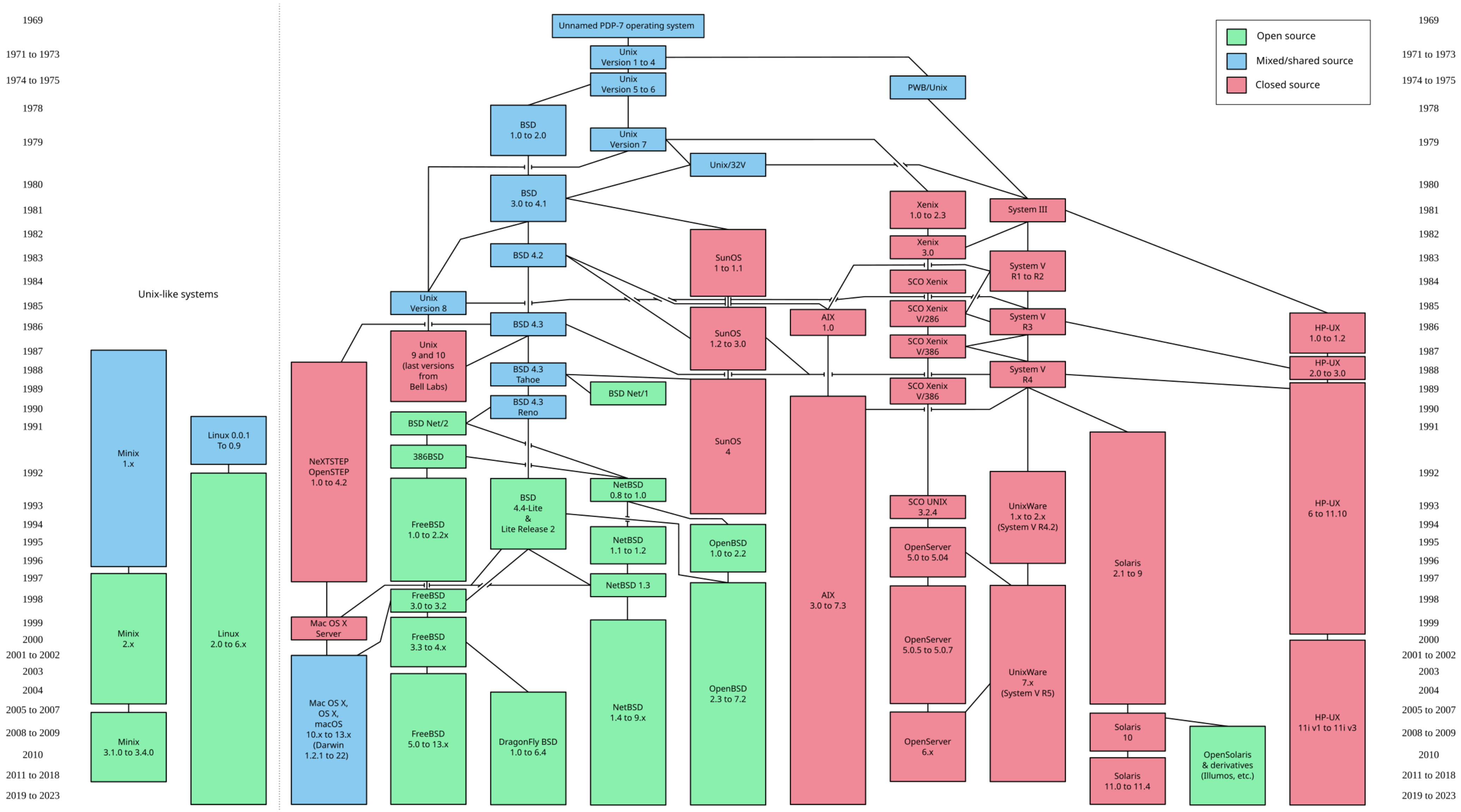
Projects

- Small projects
  - p1: Add new system calls
  - p2: Kernel module – data structure handling
- Medium project
  - p3: TBD (kernel programming project), e.g., mm or fs
- Final project
  - p4 for 4xxx: TBD (kernel programming project)
  - p4 for 5xxx: TBD (semester long research project)

Today’s Agenda

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

History of UNIX



Source: https://en.wikipedia.org/wiki/History_of_Unix

The Birth 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: Linus Torvalds
- 1992: GPL License, first Linux distributions
- 1994: v1.0 - Single CPU for i386, then ported to Alpha, Sparc, MIPS
- 1996: v2.0 - Symmetric multiprocessing (SMP) support
- 1999: v2.2 - Big Kernel Lock removed
- 2001: v2.4 - USB, RAID, Bluetooth, etc.
- 2003: v2.6 - Physical Address Expansion (PAE), new architectures, etc.
- 2011: v3.0 - Incremental release of v2.6
- 2015: v4.0
- 2022: v6.0
- 2024: v6.13 (released a few days ago)

Linux Open Source Model

- Linux is licensed under GPLv2
- Source code is freely available at https://www.kernel.org/
- Ref: tdlr Legal, GPLv2

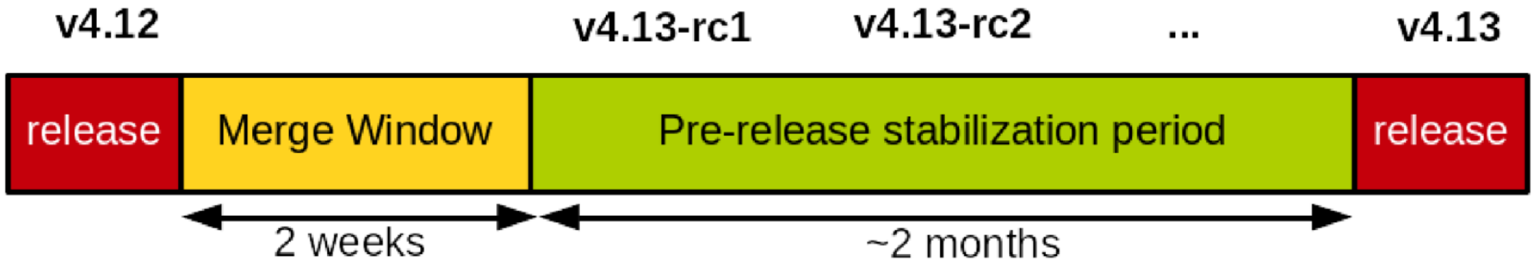
“You may copy, distribute and modify the software as long as you track changes/dates in source files. Any modifications to or software including (via compiler) GPL-licensed code must also be made available under the GPL along with build & install instructions.”

Benefit of Open Source Model

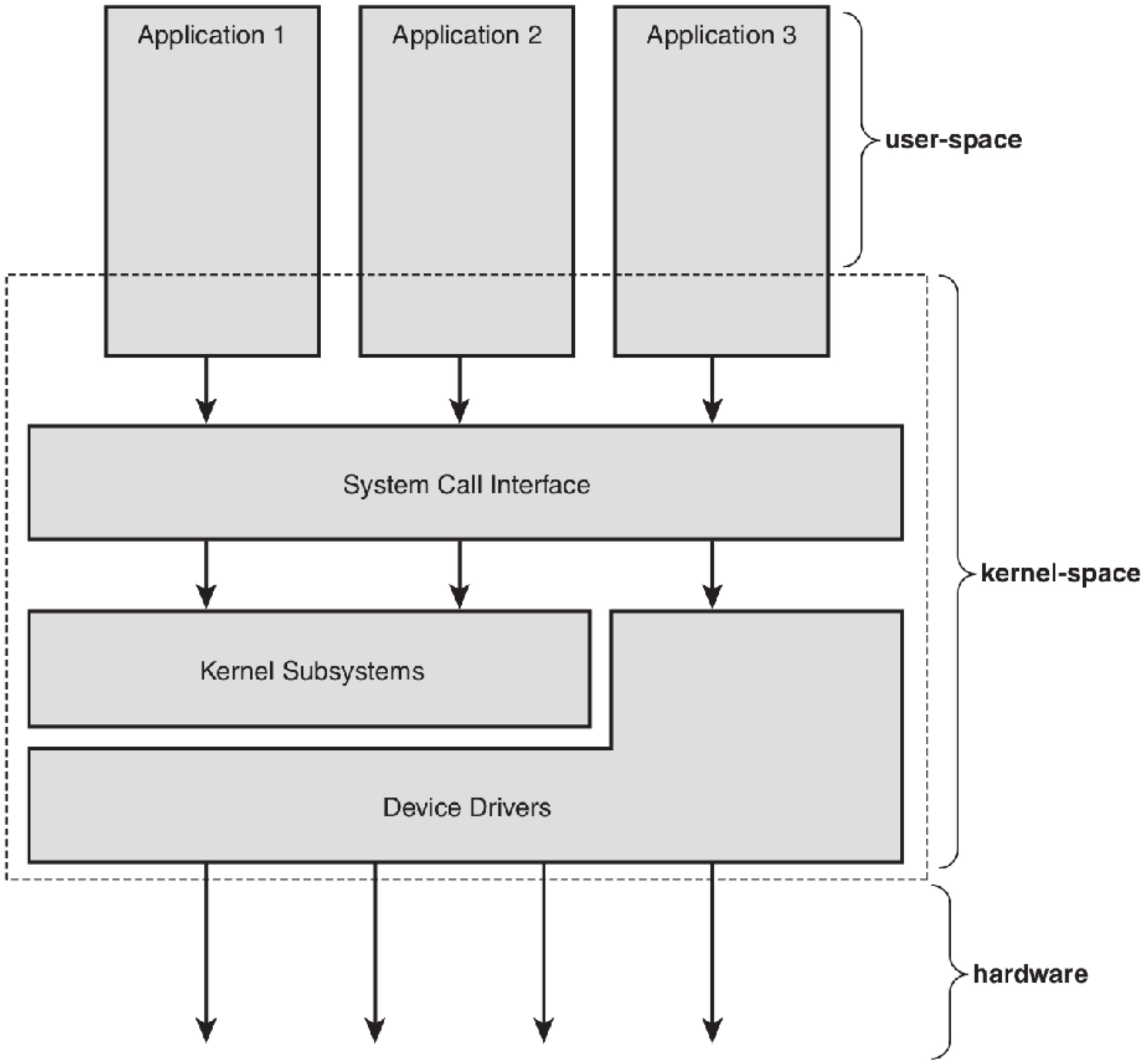
- “Given enough eyeballs, all bugs are shallow”
- “Given a large enough beta-test and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone.”
- Linus's Law
  - The Cathedral & the Bazaar by Eric S. Raymond
  - Security, stability, quality, speed of innovation, education, research, etc

Linux Kernel Release Cycles

- (major).(minor).(stable) → E.g., 5.19.3
- Prepatch or "RC" kernel release → for 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., 5.15.62

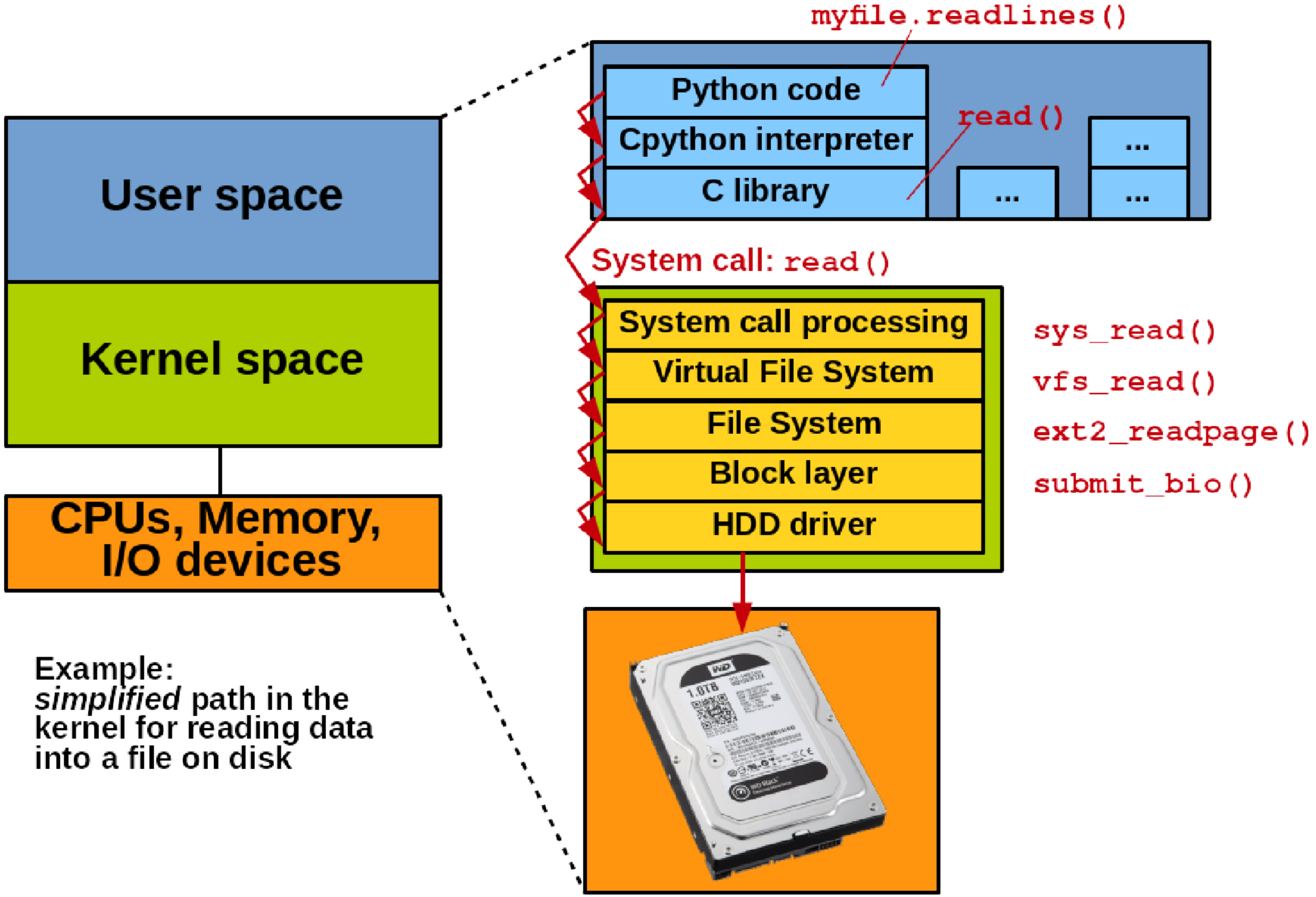


Overview of an OS



User Space vs. Kernel Space

- A CPU is executing in either of user space or in kernel space
- Only the kernel is allowed to perform privileged operations such as controlling CPU and IO devices
  - – E.g., protection ring in x86 architecture
  - – ring 3: user-space application
  - – ring 0: operating system kernel
- An user-space application talks to the kernel space through system call interface
  - – open(), read(), write(), close()



User space

Kernel space

CPUs, Memory, I/O devices

`myfile.readlines()`

Python code

Cpython interpreter

C library

`read()`

...

...

...

System call: `read()`

System call processing

Virtual File System

File System

Block layer

HDD driver

`sys_read()`

`vfs_read()`

`ext2_readpage()`

`submit_bio()`

Example:
simplified path in the
kernel for reading data
into a file on disk

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: interactions are complex leads to bugs, no isolation within kernel

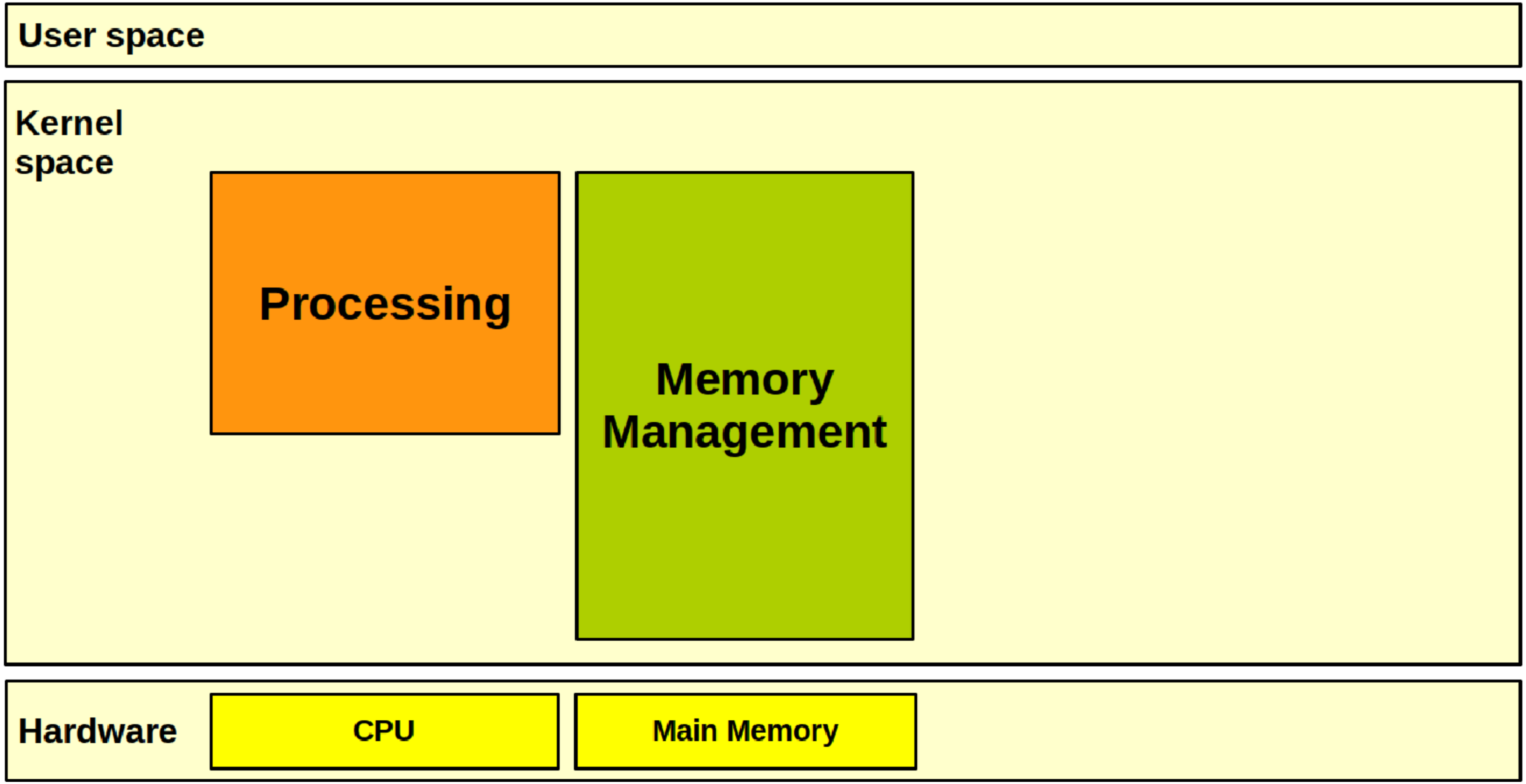
Alternative: Microkernel Design

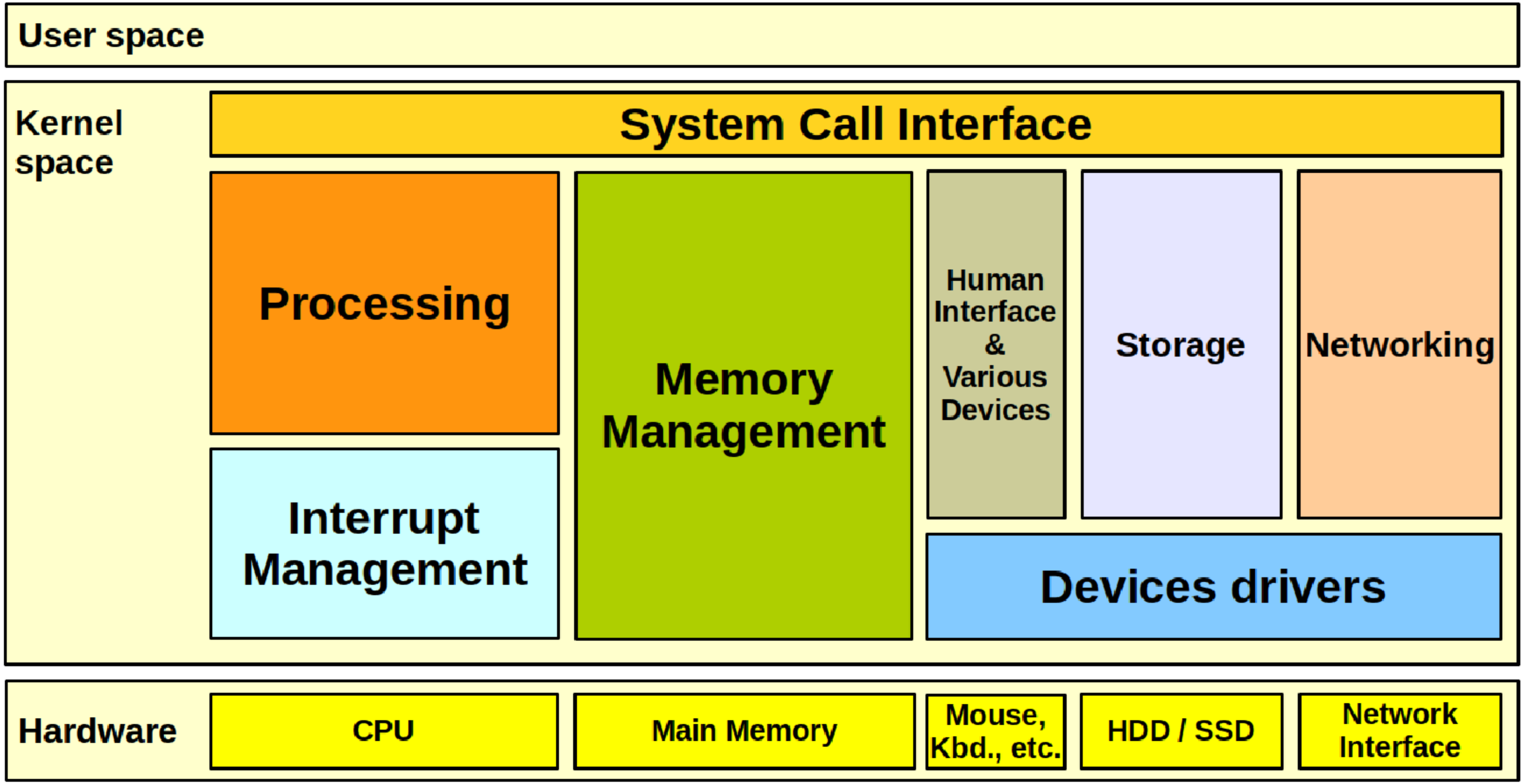
- Many OS services run as ordinary user programs
  - e.g., file system in a file server
- Kernel implements minimal mechanism to run services in user space
  - IPC, virtual memory, threads
- Kernel interface != system call interface
  - applications talk to servers via IPCs
- Good: more isolation
- Bad: IPCs may be slow

Debate

- Tanenbaum-Torvalds debate
- Most real-world kernels are mixed: Linux, OS X, Windows
  - e.g., X Window Systems

Kernel and Course Map





User space

Kernel space

System Call Interface

Processing

Interrupt Management

Memory Management

Human Interface & Various Devices

Storage

Networking

Devices drivers

Hardware

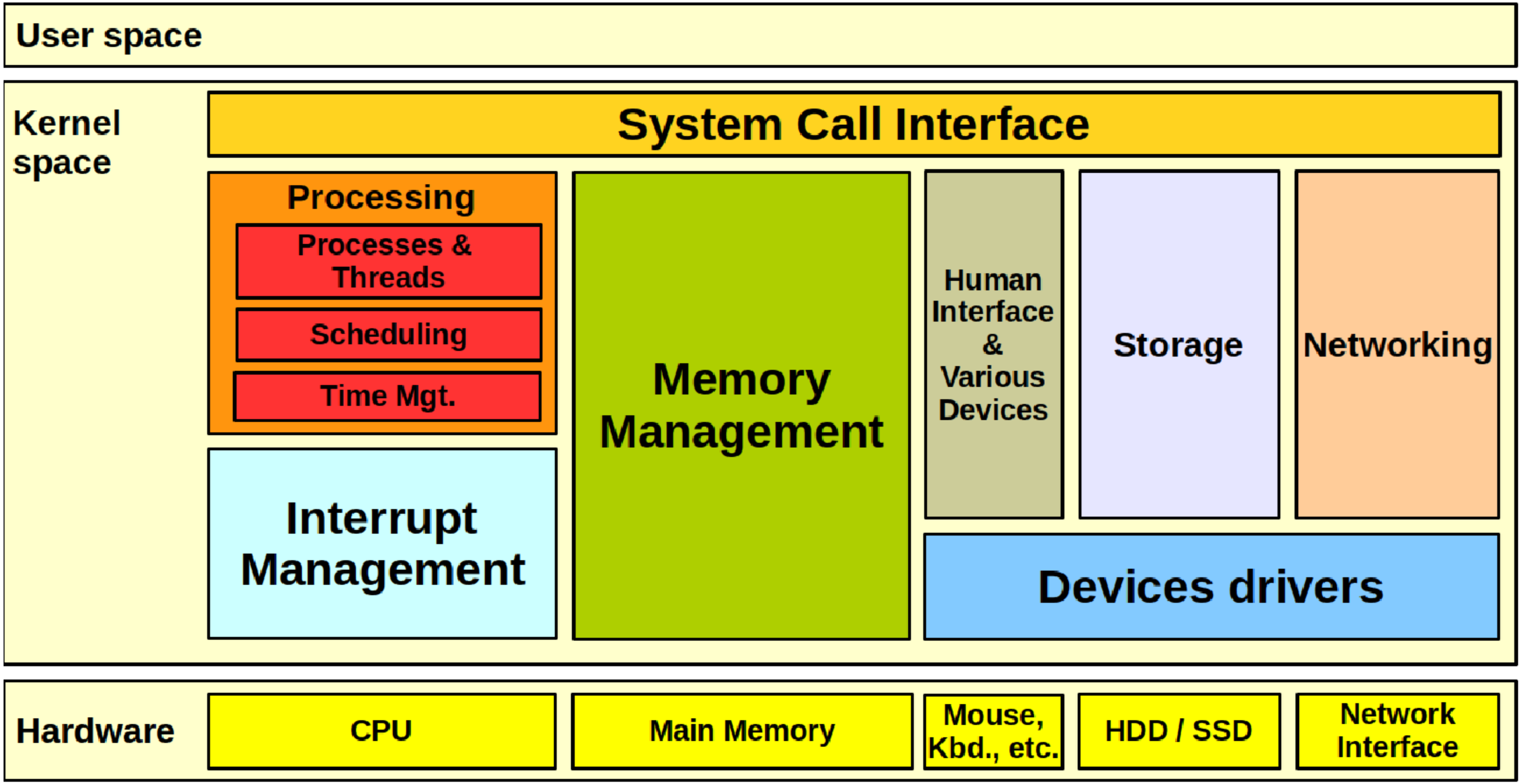
CPU

Main Memory

Mouse, Kbd., etc.

HDD / SSD

Network Interface



User space

Kernel space

System Call Interface

Processing

Processes & Threads

Scheduling

Time Mgt.

Interrupt Management

Memory Management

Human Interface & Various Devices

Storage

Networking

Devices drivers

Hardware

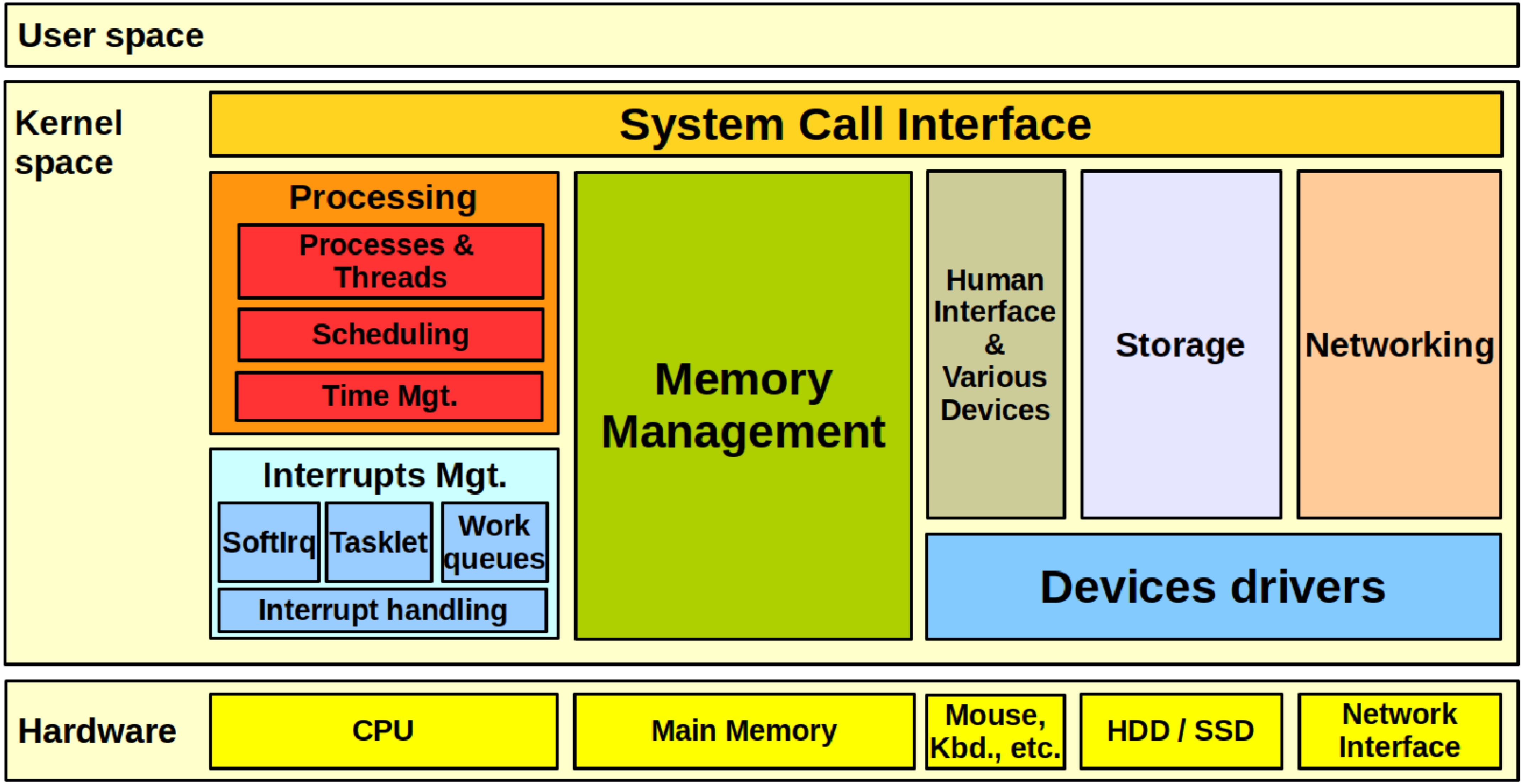
CPU

Main Memory

Mouse, Kbd., etc.

HDD / SSD

Network Interface



User space

Kernel space

System Call Interface

Processing

Processes & Threads

Scheduling

Time Mgt.

Interrupts Mgt.

SoftIrq

Tasklet

Work queues

Interrupt handling

Memory Management

Human Interface & Various Devices

Storage

Networking

Devices drivers

Hardware

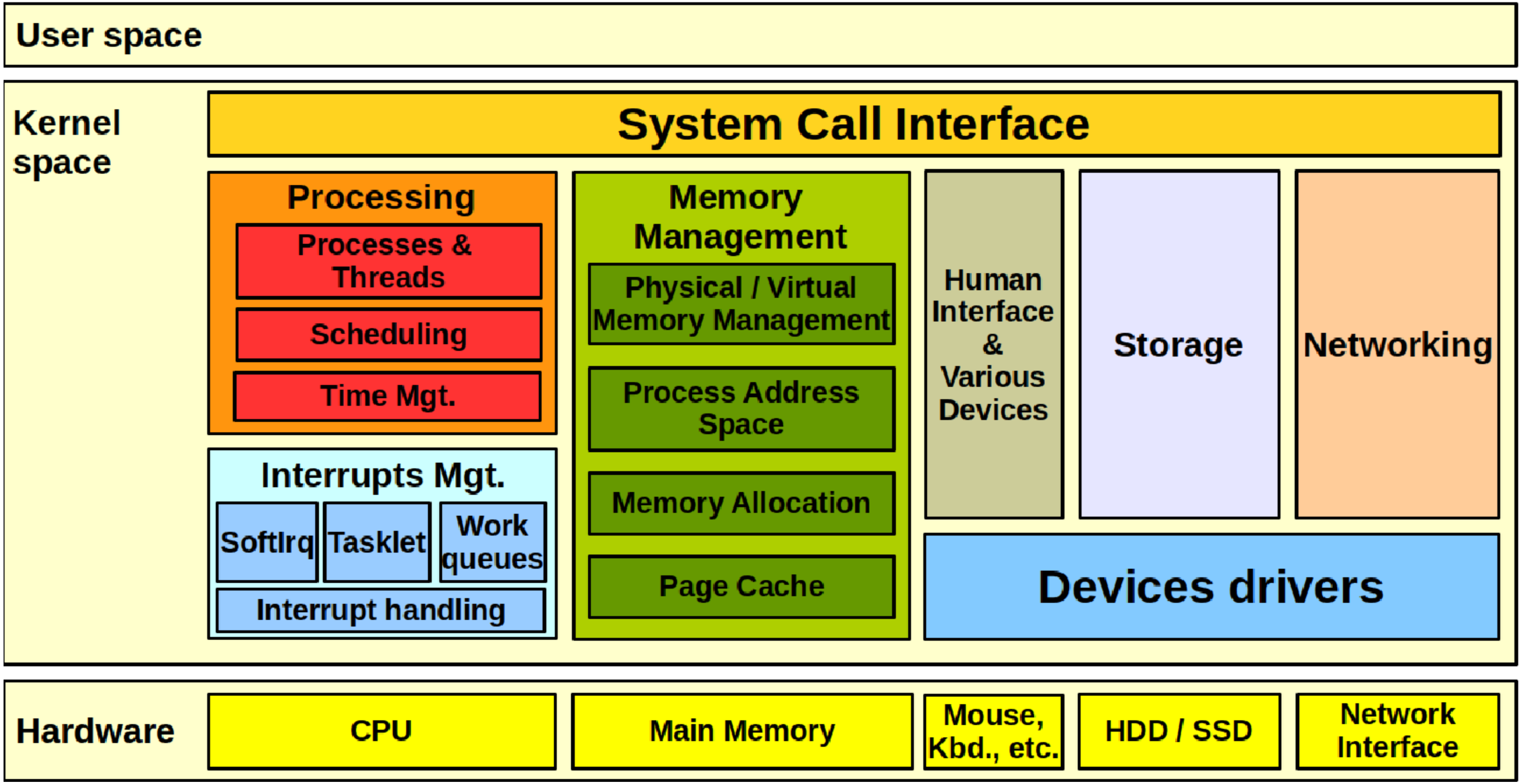
CPU

Main Memory

Mouse, Kbd., etc.

HDD / SSD

Network Interface



User space

Kernel space

System Call Interface

Processing

Processes & Threads

Scheduling

Time Mgt.

Interrupts Mgt.

SoftIrq

Tasklet

Work queues

Interrupt handling

Memory Management

Physical / Virtual Memory Management

Process Address Space

Memory Allocation

Page Cache

Human Interface & Various Devices

Storage

Networking

Devices drivers

Hardware

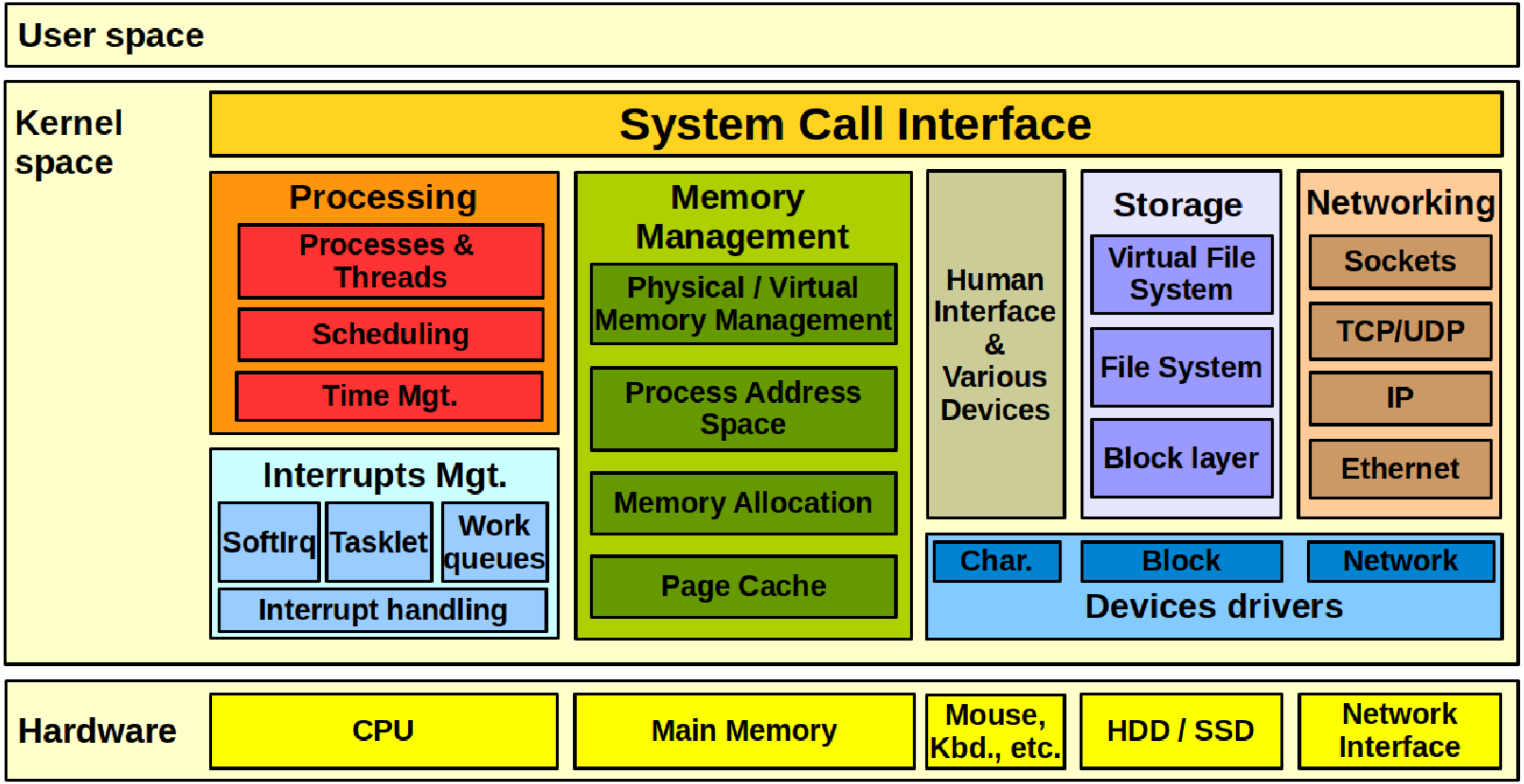
CPU

Main Memory

Mouse, Kbd., etc.

HDD / SSD

Network Interface



User space

Kernel space

System Call Interface

Processing

Processes & Threads

Scheduling

Time Mgt.

Interrupts Mgt.

SoftIrq

Tasklet

Work queues

Interrupt handling

Memory Management

Physical / Virtual Memory Management

Process Address Space

Memory Allocation

Page Cache

Human Interface & Various Devices

Storage

Virtual File System

File System

Block layer

Networking

Sockets

TCP/UDP

IP

Ethernet

Char.

Block

Network

Devices drivers

Hardware

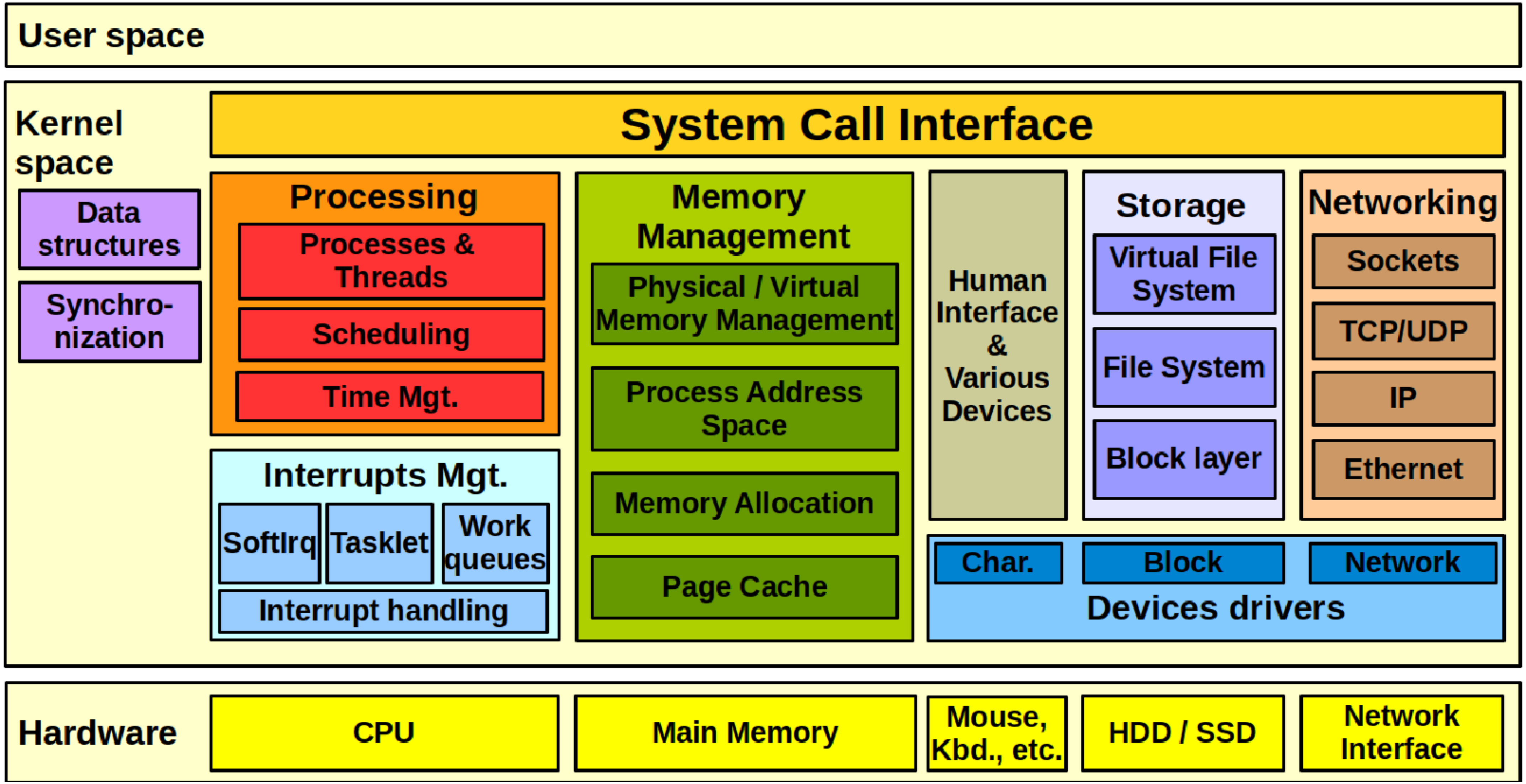
CPU

Main Memory

Mouse, Kbd., etc.

HDD / SSD

Network Interface



User space

Kernel space

System Call Interface

Data structures

Synchro-nization

Processing

Processes & Threads

Scheduling

Time Mgt.

Interrupts Mgt.

SoftIrq

Tasklet

Work queues

Interrupt handling

Memory Management

Physical / Virtual Memory Management

Process Address Space

Memory Allocation

Page Cache

Human Interface & Various Devices

Storage

Virtual File System

File System

Block layer

Networking

Sockets

TCP/UDP

IP

Ethernet

Char.

Block

Network

Devices drivers

Hardware

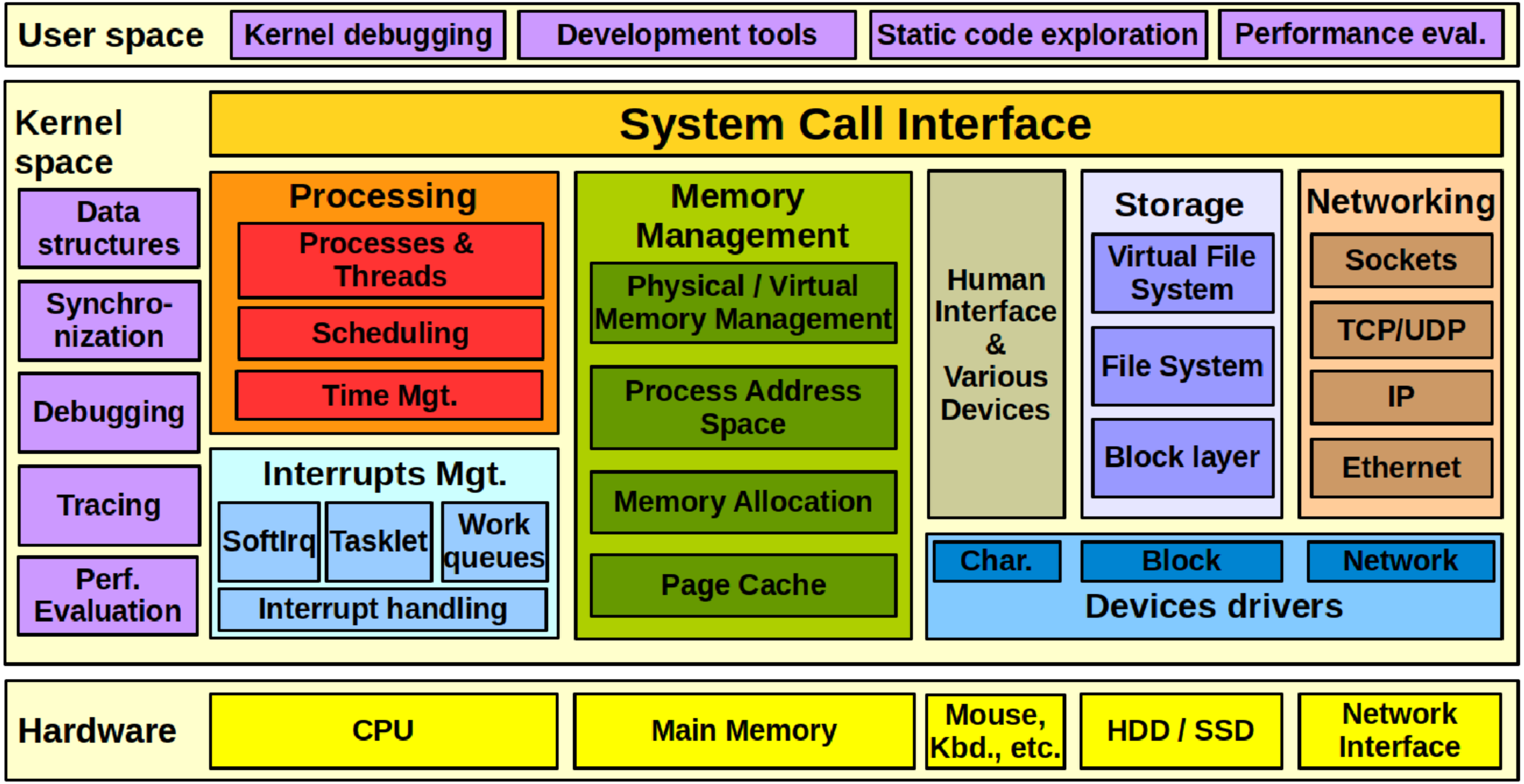
CPU

Main Memory

Mouse, Kbd., etc.

HDD / SSD

Network Interface



User space

Kernel debugging

Development tools

Static code exploration

Performance eval.

Kernel space

System Call Interface

Data structures

Synchro-nization

Debugging

Tracing

Perf. Evaluation

Processing

Processes & Threads

Scheduling

Time Mgt.

Interrupts Mgt.

SoftIrq

Tasklet

Work queues

Interrupt handling

Memory Management

Physical / Virtual Memory Management

Process Address Space

Memory Allocation

Page Cache

Human Interface & Various Devices

Storage

Virtual File System

File System

Block layer

Networking

Sockets

TCP/UDP

IP

Ethernet

Char.

Block

Network

Devices drivers

Hardware

CPU

Main Memory

Mouse, Kbd., etc.

HDD / SSD

Network Interface

- Schedule: https://people.cs.vt.edu/huaicheng/lkp-sp25/schedule/

Setup Dev Environment

- VirtualBox / VMWare Workstation / QEMU to run Linux VM
  - Recommended settings
    - disk >= 64GB, DRAM >= 4GB, #CPU >=4
  - Porting forwarding:
    - protocol: TCP, host IP: 127.0.0.1
    - host port: 2222, guest port: 22
  - VM/host file transmission: scp, folder sharing, etc.
- Guest OS
  - Ubuntu server 24.04, or any other Linux distros
    - openssh-server
    - ssh –p2222 $username@localhost

Next Step

- Demos on setting up the virtual machine env
- Bring your laptop
- Ex0 will be released on Wednesday
- Productive tools:
  - vim, ssh, scp, tmux, git, and more
  - Check the missing semester of your cs education,
- Linux source code:
  - git clone https://github.com/torvalds/linux.git
- Next Lecture,
  - building and exploring Linux kernel source code