Week 1

Introduction to OS

Oana Balmau

January 5, 2023

Welcome to OS!

See office hours schedule and zoom links in Syllabus. TBD

- Instructor: Oana Balmau
- TAs:
  - Alice Chang — le.chang2@mail.mcgill.ca
  - Clare Jang — junyoung.jang@mail.mcgill.ca
  - Aayush Kapur — aayush.kapur@mail.mcgill.ca
  - Murray Kornelsen — murray.kornelsen@mcgill.ca
  - Mohammad Rahman — mohammad.rahman4@mail.mcgill.ca
  - Sebastian Rolon — sebastian.rolon@mail.mcgill.ca
  - Jiaxuan Chen — jiaxuan.chen2@mail.mcgill.ca
  - Emmanuel Wilson — emmanuel.wilson2@mail.mcgill.ca
- Graders:
  - Zhongjie Wu — zhongjie.wu@mail.mcgill.ca
  - Shakiba Bolbolian Khah — shakiba.bolboliankhah@mail.mcgill.ca

About Me

- Prof at McGill since Jan ’21
  - Research area: Systems
  - OS, Storage, Data-Intensive Applications
- PhD at EPFL + University of Sydney
- BSc, MSc in Computer Science EPFL
  - Masters Thesis at Brown University

TA Introductions

About You ☺

- Cross listed Course: **384 students**, up by 80 students since last year.

- ~132 Electrical & Computer Engineering students
- ~253 Computer Science students

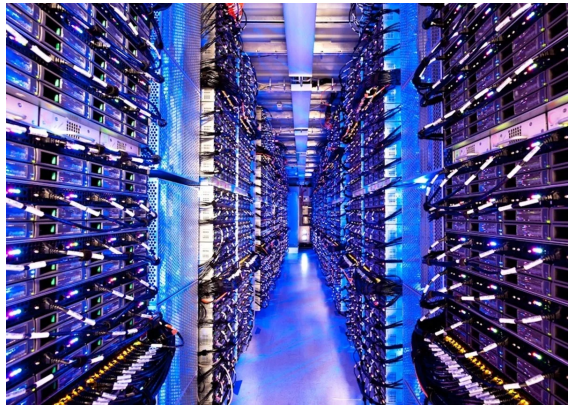
About the course

Operating Systems

Why should you care?

Why should you care?

What do these have in common?



What do these devices have in common?

They have an operating system (OS)

Every program you will ever write will run on an OS

Its performance and execution will depend on the OS

Develop systems-thinking

- OS is one of the oldest disciplines in CS
- With a lot of influence on other systems disciplines.

OS concepts are the foundation for:

- Distributed systems (e.g., blockchain), Cloud infrastructure, Internet of things, Database infrastructure

Overall goal of COMP-310/ECSE-427

- Learn **principles** of Operating Systems

Method

- Lectures
- Exercises
- Programming assignments
- Labs

Everything is recorded and posted on MyCourses ☺

Grading

- Project, consisting of 3 C programming assignments
  - OS Shell 10%
  - Scheduling 20%
  - Memory management 20%
- Take-home graded exercises – 10%
- Written final in exam session – 40%
  - If your final exam grade is higher than your take-home exercises grade, then the exercises will not count.
  - i.e., the final counts for 50%.

Topic	Monday	Tuesday	Wednesday	Thursday	Friday
Week 1 **Introduction**	jan 2	jan 3	jan 4 – first day of class 😊	jan 5 **Course logistics and Intro to OS**	jan 6
Week 2 **Process Management**	jan 9 **Workflow: working with mimi and GitLab, Git basics**	jan 10 **Intro to Process Management (1/2)** **Optional reading: OSTEP Chapters 3 – 7**	jan 11	jan 12 **Intro to Process Management (2/2)**	jan 13
Week 3 **Process Management**	jan 16 **C Review: C Basics**	jan 17 **Synchronization Primitives (1/2)** **Optional reading: OSTEP Chapters 25 – 32** **add/drop deadline**	jan 18 OS Shell Assignment Released	jan 19 **Synchronization Primitives (2/2)** OS Shell Assignment Overview – with Jiaxuan	jan 20
Week 4 **Process Management**	jan 23 **C Tools: GDB basics**	jan 24 **Multi-process Structuring (1/2)** **Team registration deadline**	jan 25	jan 26 **Multi-process Structuring (2/2)**	jan 27
Week 5 **Process Management**	jan 30 **C Review: Pointers & Memory Allocation I**	jan 31 **Multithreading (1/2)** Practice Exercises Sheet: Process Management	feb 1	feb 2 **Multithreading (2/2)**	feb 3
Week 6 **Memory Management**	feb 6 **C Review: C files**	feb 7 **Virtual Memory (1/2)** **Optional reading: OSTEP Chapters 12 – 18**	feb 8 Scheduling Assignment Released	feb 9 **Virtual Memory (2/2)** Scheduling Assignment Overview – with Jiaxuan	feb 10
Week 7 **Memory Management**	feb 13 **OS Shell Assignment Due** **C Review: Working with pthreads I**	feb 14 **Demand Paging (1/3)** **Optional reading: OSTEP Chapters 19 – 22**	feb 15	feb 16 **Demand Paging (2/3)**	feb 17
Week 8 **Memory Management**	feb 20 **C Review: Working with pthreads II**	feb 21 **Demand Paging (3/3)** **Optional reading: OSTEP Chapters 19 – 22** Practice Exercises Sheet: Memory Management	feb 22	feb 23 Mid-semester Q&A – not recorded • Graded Exercises Sheet Released • Grades released for OS Shell Assignment	feb 24
Week 9 **Reading week**	feb 27 **No class**	feb 28 **No class**	mar 1 **No class**	mar 2 **No class**	mar 3 **No class**
Week 10 **File Systems**	mar 6 No lab. Work on Assignment 2 **Scheduling Assignment Due**	mar 7 **Intro to File Systems (1/2)** **Recorded lecture. Do not come to class.** **Optional reading: OSTEP Chapters 36, 37, 39**	mar 8 **Memory Management Assignment Released**	mar 9 **Intro to File Systems (2/2)** Memory Management Assignment Overview – with Jiaxuan	mar 10
Week 11 **File Systems**	mar 13 **Graded Exercises Due** **C Review: Complex structs**	mar 14 **Basic File System Implementation (1/2)** **Optional reading: OSTEP Chapters 40, 41, 45**	mar 15	mar 16 **Basic File System Implementation (2/2)** • Grades released for Scheduling Assignment	mar 17
Week 12 **File Systems**	mar 20 **C Review: Pointers & Memory Allocation II**	mar 21 **Advanced File System Implementation (1/2)**	mar 22	mar 23 **Advanced File System Implementation (2/2)**	mar 24
Week 13 **File Systems**	mar 27 **C Review: Advanced debugging**	mar 28 **Handling Crashes & Performance (1/2)** **Optional reading: OSTEP Chapters 38, 43**	mar 29	mar 30 **Handling Crashes & Performance (2/2)** • Grades released for Exercises Sheet • Practice Exercises Sheet: File Systems	mar 31
Week 14 **Advanced Topics**	apr 3 No lab. Work on Assignment 3 **Memory Management Assignment Due**	apr 4 **Advanced topics: Virtualization**	apr 5	apr 6 **Advanced topics: Operating Systems Research** **(Invited Speaker: TBD)**	apr 7
Week 15 **Wrap-up**	apr 10 No Lab. Prepare for end-of-semester.	apr 11 **End-of-semester Q&A– not recorded**	apr 12	apr 13 **End-of-semester Q&A – not recorded. Last class!** • Grades released for Memory Management Assignment	apr 14

Tentative class schedule

link

Lectures

- Slides + Recordings
  - Best effort recordings
- Slides available on Monday mornings on MyCourses
- Recordings available by Friday on MyCourses

Exercises

- Sprinkled through the lectures.
- 3 ungraded exercise sheets for practice.
- 1 graded take-home exercise sheet (details later).

Programming Assignments

- 3 assignments in C
- Assignments build upon each other
  - OS Shell
  - Scheduling
  - Memory Management
- Create a simple OS simulation (run in user-mode)

Assignments Logistics

- Must run on DISCS server
  - Details on how to run tests on DISCS server will follow.

- Must be solved in C
  - If you need a C refresher, attend the C labs.

- For local development, use the SOCS mimi servers

```
ssh <SOCSusername>@mimi.cs.mcgill.ca
```

CS Accounts

If you don’t have a CS account, make one today

- Most likely, you do not have an account if ECSE student
- Make an account here:

https://www.cs.mcgill.ca/docs/

Assignment Grading

- Based on Unit tests and code quality

Unit Tests

- We will provide you with all the unit tests and expected output for each assignment.
- Unit tests will be automatically executed, daily, on the DISCS server.
- You will get points for each correct expected output.
  - Formatting (spaces, capitalization, new lines, etc.) **will not** be taken into account for expected outputs.

Code quality

- TAs will check for **hardcoded results.**
  - And will remove all the points for hardcoded test outputs
- TAs can remove points for **coding style.**
  - Make sure to respect the course programming style expectations.
  - The guide will be posted on MyCourses soon.

Topic	Monday	Tuesday	Wednesday	Thursday	Friday
Week 1 **Introduction**	jan 2	jan 3	jan 4 – first day of class 😊	jan 5 **Course logistics and Intro to OS**	jan 6
Week 2 **Process Management**	jan 9 **Workflow: working with mimi and GitLab, Git basics**	jan 10 **Intro to Process Management (1/2)** Optional reading: OSTEP **Chapters 3 – 7**	jan 11	jan 12 **Intro to Process Management (2/2)**	jan 13
Week 3 **Process Management**	jan 16 **C Review: C Basics**	jan 17 **Synchronization Primitives (1/2)** Optional reading: OSTEP **Chapters 25 – 32** **add/drop deadline**	jan 18 OS Shell Assignment Released	jan 19 **Synchronization Primitives (2/2)** OS Shell Assignment Overview – with Jiaxuan	jan 20
Week 4 **Process Management**	jan 23 **C Tools: GDB basics**	jan 24 **Multi-process Structuring (1/2)** **Team registration deadline**	jan 25	jan 26 **Multi-process Structuring (2/2)**	jan 27
Week 5 **Process Management**	jan 30 **C Review: Pointers & Memory Allocation I**	jan 31 **Multithreading (1/2)** Practice Exercises Sheet: Process Management	feb 1	feb 2 **Multithreading (2/2)**	feb 3
Week 6 **Memory Management**	feb 6 **C Review: C files**	feb 7 **Virtual Memory (1/2)** Optional reading: OSTEP **Chapters 12 – 18**	feb 8 Scheduling Assignment Released	feb 9 **Virtual Memory (2/2)** Scheduling Assignment Overview – with Jiaxuan	feb 10
Week 7 **Memory Management**	feb 13 **OS Shell Assignment Due** **C Review: Working with pthreads I**	feb 14 **Demand Paging (1/3)** Optional reading: OSTEP **Chapters 19 – 22**	feb 15	feb 16 **Demand Paging (2/3)**	feb 17
Week 8 **Memory Management**	feb 20 **C Review: Working with pthreads II**	feb 21 **Demand Paging (3/3)** Optional reading: OSTEP **Chapters 19 – 22** Practice Exercises Sheet: Memory Management	feb 22	feb 23 Mid-semester Q&A – not recorded • Graded Exercises Sheet Released • Grades released for OS Shell Assignment	feb 24
Week 9 **Reading week**	feb 27 **No class**	feb 28 **No class**	mar 1 **No class**	mar 2 **No class**	mar 3 **No class**
Week 10 **File Systems**	mar 6 No lab. Work on Assignment 2 **Scheduling Assignment Due**	mar 7 **Intro to File Systems (1/2)** **Recorded lecture. Do not come to class.** Optional reading: OSTEP **Chapters 36, 37, 39**	mar 8 **Memory Management Assignment Released**	mar 9 **Intro to File Systems (2/2)** Memory Management Assignment Overview – with Jiaxuan	mar 10
Week 11 **File Systems**	mar 13 **Graded Exercises Due** **C Review: Complex structs**	mar 14 **Basic File System Implementation (1/2)** Optional reading: OSTEP **Chapters 40, 41, 45**	mar 15	mar 16 **Basic File System Implementation (2/2)** • Grades released for Scheduling Assignment	mar 17
Week 12 **File Systems**	mar 20 **C Review: Pointers & Memory Allocation II**	mar 21 **Advanced File System Implementation (1/2)**	mar 22	mar 23 **Advanced File System Implementation (2/2)**	mar 24
Week 13 **File Systems**	mar 27 **C Review: Advanced debugging**	mar 28 **Handling Crashes & Performance (1/2)** Optional reading: OSTEP **Chapters 38, 43**	mar 29	mar 30 **Handling Crashes & Performance (2/2)** • Grades released for Exercises Sheet • Practice Exercises Sheet: File Systems	mar 31
Week 14 **Advanced Topics**	apr 3 No lab. Work on Assignment 3 **Memory Management Assignment Due**	apr 4 **Advanced topics: Virtualization**	apr 5	apr 6 **Advanced topics: Operating Systems Research** (Invited Speaker: TBD)	apr 7
Week 15 **Wrap-up**	apr 10 No Lab. Prepare for end-of-semester.	apr 11 **End-of-semester Q&A– not recorded**	apr 12	apr 13 **End-of-semester Q&A – not recorded. Last class!** • Grades released for Memory Management Assignment	apr 14

Use this time to
Get familiar with git, linux environment, and grading infrastructure.

2+/3+ Weeks to solve each assignment; plenty of time
→ No extensions

Starter Code

- We provide starter code for assignments
- Starter code isn’t perfect
  - But it provides basic functionality we are looking for in class

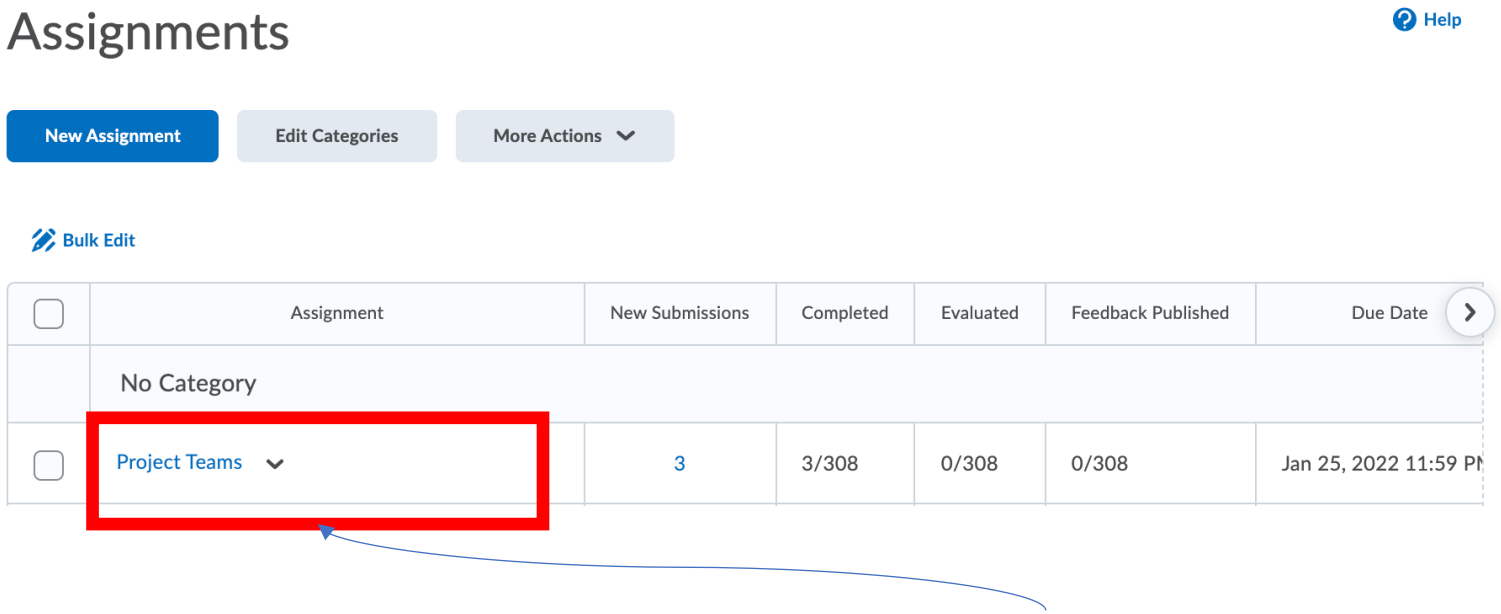
- Feel free to use our starter code
- or code your OS from scratch ☺

Teams for Assignments

- Assignments can be done individually
  - Workload and timeframe are ok to solve assignments on your own.

- Or in **teams of 2**
  - If you decide to team-up,
  - you commit to your teammate for **all 3 assignments**
  - **both teammates get the same grade**

How to form teams on MyCourses



- Upload a .txt file with the names & McGill IDs of the 2 teammates, and Gitlab link
  - Only upload one file per team.
- If you want to work alone, **you will need to register a team of 1,** with your name & McGill ID, and Gitlab link

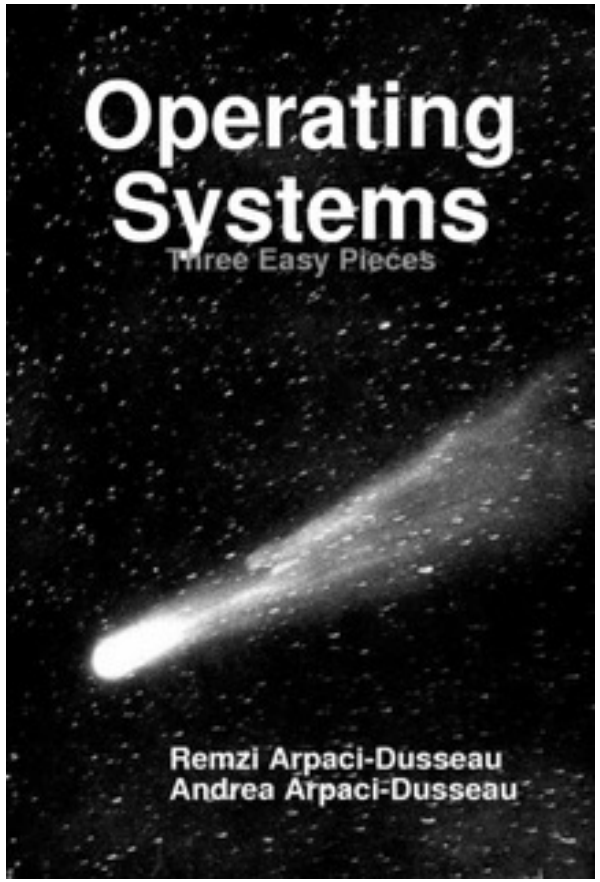
Topic	Monday	Tuesday	We
Week 1 **Introductio**	jan 2	an 3	jan 4 – fi class 😊
Week 2 **Process Manageme t**	jan 9 Workflow: working with mimi and GitLab, Git basics	an 10 **ntro to Process Management (1/2)** Optional reading: OSTEP Chapters 3 – 7	jan 11
Week 3 **Process Manageme t**	jan 16 C Review: C Basics	an 17 **ynchronization Primitives (1/2)** Optional reading: OSTEP Chapters 25 – 32 **dd/drop deadline**	jan 18 OS Shell Released
Week 4 **Process Manageme t**	jan 23 C Tools: GDB basics	an 24 **Multi-process Structuring (1/2)** **eam registration deadline**	jan 25
Week 5 **Process Manageme t**	jan 30 C Review: Pointers & Memory Allocation I	an 31 **Multithreading (1/2)** ractice Exercises Sheet: Process Management	feb 1
Week 6 **Memory Manageme t**	feb 6 C Review: C files	eb 7 **irtual Memory (1/2)** Optional reading: OSTEP Chapters 12 – 18	feb 8 Schedul Assignm
Week 7 **Memory Manageme t**	feb 13 **OS Shell Assignment Due** C Review: Working with pthreads I	eb 14 **emand Paging (1/3)** Optional reading: OSTEP Chapters 19 – 22	feb 15
Week 8 **Memory Manageme t**	feb 20 C Review: Working with pthreads II	eb 21 **emand Paging (3/3)** Optional reading: OSTEP Chapters 19 – 22 ractice Exercises Sheet: Memory Management	feb 22
Week 9 **Reading we k**	feb 27 **No class**	eb 28 **o class**	mar 1 **No class**
Week 10 **File System**	mar 6 No lab. Work on Assignment 2 **Scheduling Assignment Due**	ar 7 **ntro to File Systems (1/2)** ecorded lecture. Do not come to class. Optional reading: OSTEP Chapters 36, 37, 39	mar 8 Memory Manage Assignm
Week 11 **File System**	mar 13 **Graded Exercises Due** C Review: Complex structs	ar 14 **asic File System Implementation (1/2)** Optional reading: OSTEP Chapters 40, 41, 45	mar 15
Week 12 **File System**	mar 20 C Review: Pointers & Memory Allocation II	ar 21 **dvanced File System Implementation (1/2)**	mar 22
Week 13 **File System**	mar 27 C Review: Advanced debugging	ar 28 **andling Crashes & Performance (1/2)** Optional reading: OSTEP Chapters 38, 43	mar 29
Week 14 **Advanced Topics**	apr 3 No lab. Work on Assignment 3 **Memory Management Assignment Due**	pr 4 **dvanced topics: Virtualization**	apr 5
Week 15 **Wrap-up**	apr 10 No Lab. Prepare for end-of-semester	pr 11 **nd-of-semester Q&A– not recorded**	apr 12

Labs

- In-person sessions on Mondays.
- Recordings + slides posted on MyCourses by Friday.
- Goal: Refresh your C knowledge
  - Meant as a support for students who are a bit rusty in C.

- Labs are fully led by TAs.

Recommended Book



A *free* online book: http://pages.cs.wisc.edu/~remzi/OSTEP/

Prerequisites

- **ECSE-324** or
- **COMP-273**

Late policy

- No extensions
- See syllabus for exceptional situations (e.g., medical emergencies)

Ed – Main place to ask course-related questions

- Do not send course-related questions by email
- Ed Sign-up link: https://edstem.org/us/join/AP6Xuu
  - Sign up now, please
  - You need to login using McGill email to be able to sign up.
- Ed discussion link: https://edstem.org/us/courses/32167/discussion/

Email Policy

- Do not email course-related questions
  - Use Ed
- For issues with **grading**, email **TA who graded**
  - not the instructor.
- For **personal and medical** issues, feel free to **send email to instructor**.

Questions?

Introducing the OS

- What does the OS do?
- Where does the OS live?
- OS interfaces
- OS control flow
- OS structure

What does an OS do?

A Bit of History

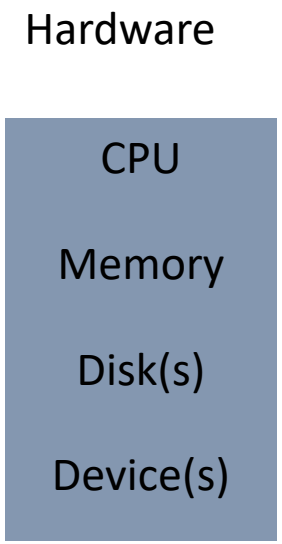
- Early days
  - Users program raw machine
- First “abstraction”
  - Libraries for scientific functions (sin, cos, …)
  - Libraries for doing I/O
- I/O libraries are the first pieces of an OS

What does the OS do?

- Abstraction: makes hardware easier to use

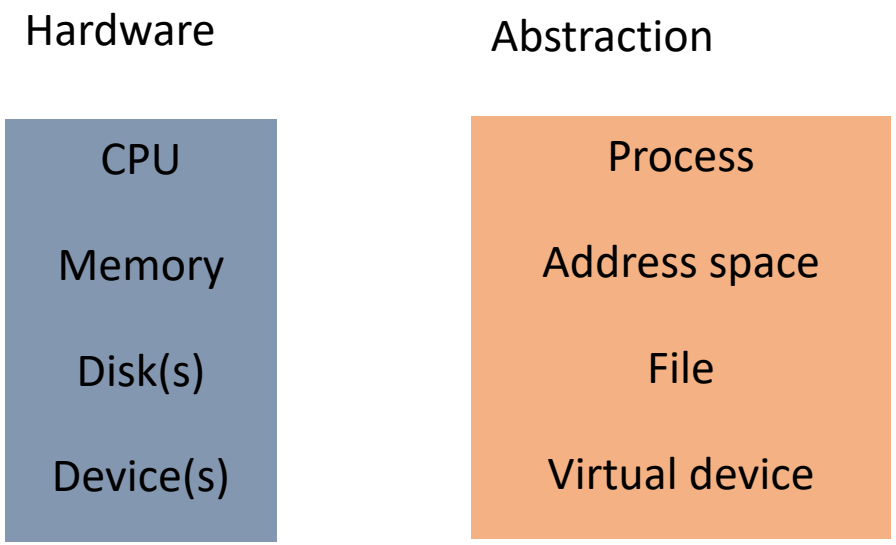
What does the OS do?

- Abstraction: makes hardware easier to use



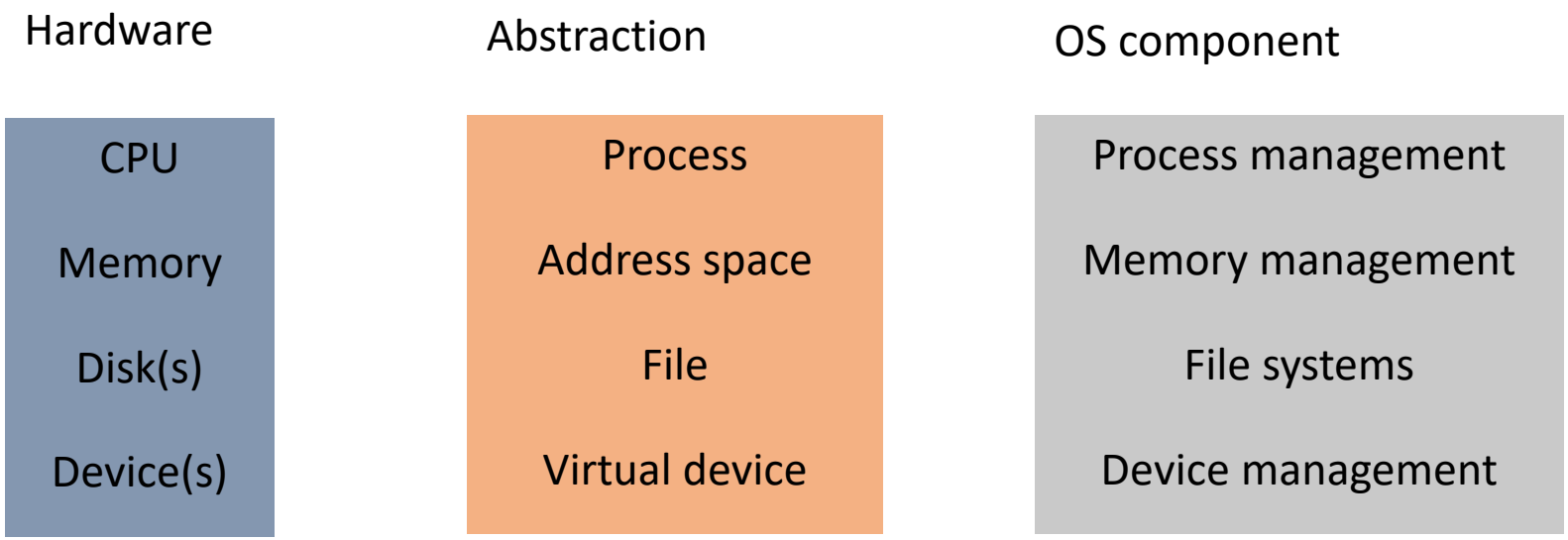
What does the OS do?

- Abstraction: makes hardware easier to use



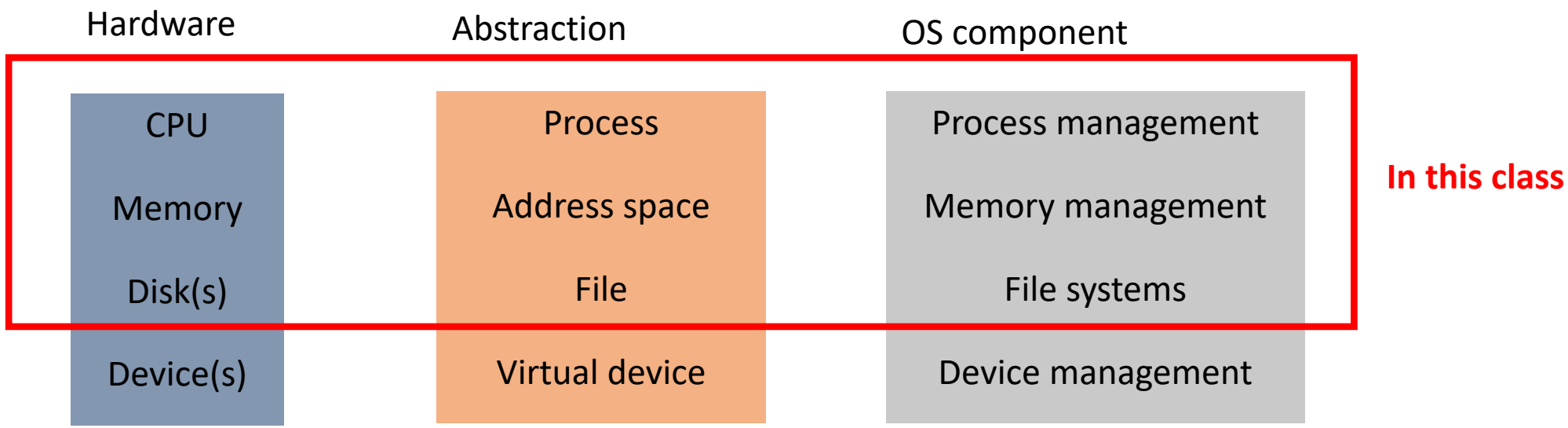
What does the OS do?

- Abstraction: makes hardware easier to use



What does the OS do?

- Abstraction: makes hardware easier to use



Topic	Monday	Tuesday	Wednesday	Thursday	Friday
Week 1 **Introduction**	jan 2	jan 3	jan 4 – first day of class 😊	jan 5 **Course logistics and Intro to OS**	jan 6
Week 2 **Process Management**	jan 9 **Workflow: working with mimi and GitLab, Git basics**	jan 10 **Intro to Process Management (1/2)** Optional reading: OSTEP Chapters 3 – 7	jan 11	jan 12 **Intro to Process Management (2/2)**	jan 13
Week 3 **Process Management**	jan 16 **C Review: C Basics**	jan 17 **Synchronization Primitives (1/2)** Optional reading: OSTEP Chapters 25 – 32 add/drop deadline	jan 18 OS Shell Assignment Released	jan 19 **Synchronization Primitives (2/2)** OS Shell Assignment Overview – with Jiaxuan	jan 20
Week 4 **Process Management**	jan 23 **C Tools: GDB basics**	jan 24 **Multi-process Structuring (1/2)** **Team registration deadline**	jan 25	jan 26 **Multi-process Structuring (2/2)**	jan 27
Week 5 **Process Management**	jan 30 **C Review: Pointers & Memory Allocation I**	jan 31 **Multithreading (1/2)** Practice Exercises Sheet: Process Management	feb 1	feb 2 **Multithreading (2/2)**	feb 3
Week 6 **Memory Management**	feb 6 **C Review: C files**	feb 7 **Virtual Memory (1/2)** Optional reading: OSTEP Chapters 12 – 18	feb 8 Scheduling Assignment Released	feb 9 **Virtual Memory (2/2)** Scheduling Assignment Overview – with Jiaxuan	feb 10
Week 7 **Memory Management**	feb 13 **OS Shell Assignment Due** **C Review: Working with pthreads I**	feb 14 **Demand Paging (1/3)** Optional reading: OSTEP Chapters 19 – 22	feb 15	feb 16 **Demand Paging (2/3)**	feb 17
Week 8 **Memory Management**	feb 20 **C Review: Working with pthreads II**	feb 21 **Demand Paging (3/3)** Optional reading: OSTEP Chapters 19 – 22 Practice Exercises Sheet: Memory Management	feb 22	feb 23 Mid-semester Q&A – not recorded • Graded Exercises Sheet Released • Grades released for OS Shell Assignment	feb 24
Week 9 **Reading week**	feb 27 **No class**	feb 28 **No class**	mar 1 **No class**	mar 2 **No class**	mar 3 **No class**
Week 10 **File Systems**	mar 6 No lab. Work on Assignment 2 **Scheduling Assignment Due**	mar 7 **Intro to File Systems (1/2)** Recorded lecture. Do not come to class. Optional reading: OSTEP Chapters 36, 37, 39	mar 8 **Memory Management Assignment Released**	mar 9 **Intro to File Systems (2/2)** Memory Management Assignment Overview – with Jiaxuan	mar 10
Week 11 **File Systems**	mar 13 **Graded Exercises Due** **C Review: Complex structs**	mar 14 **Basic File System Implementation (1/2)** Optional reading: OSTEP Chapters 40, 41, 45	mar 15	mar 16 **Basic File System Implementation (2/2)** • Grades released for Scheduling Assignment	mar 17
Week 12 **File Systems**	mar 20 **C Review: Pointers & Memory Allocation II**	mar 21 **Advanced File System Implementation (1/2)**	mar 22	mar 23 **Advanced File System Implementation (2/2)**	mar 24
Week 13 **File Systems**	mar 27 **C Review: Advanced debugging**	mar 28 **Handling Crashes & Performance (1/2)** Optional reading: OSTEP Chapters 38, 43	mar 29	mar 30 **Handling Crashes & Performance (2/2)** • Grades released for Exercises Sheet • Practice Exercises Sheet: File Systems	mar 31
Week 14 **Advanced Topics**	apr 3 No lab. Work on Assignment 3 **Memory Management Assignment Due**	apr 4 **Advanced topics: Virtualization**	apr 5	apr 6 **Advanced topics: Operating Systems Research** (Invited Speaker: TBD)	apr 7
Week 15 **Wrap-up**	apr 10 No Lab. Prepare for end-of-semester.	apr 11 **End-of-semester Q&A– not recorded**	apr 12	apr 13 **End-of-semester Q&A – not recorded. Last class!** • Grades released for Memory Management Assignment	apr 14

Tentative class schedule

link

Simple Example

Simple Example

- Write a Photoshop application

- Easier to deal with files containing photos
- Than to deal with data locations on disk

- OS provides **file abstraction**
- Finds data locations on disk given file name

Another Simple Example

- Write a web server

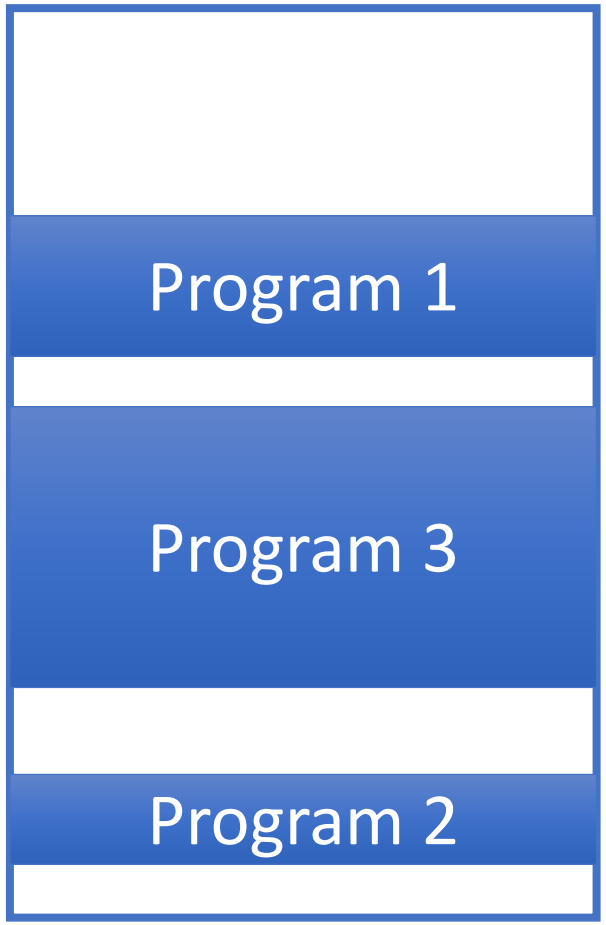
- Easier to deal with sending/receiving packets
- Than with NIC device registers

- OS provides **packet abstraction**
- Does the NIC device register manipulation

A Bit More History

- At some point, multiprogramming
- More than one program runs at the same time

Multiprogramming



Memory

Multiprogramming

- Need to protect programs from each other
- Need to protect OS from programs

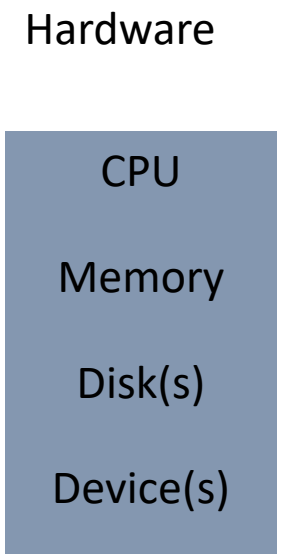
- Need to allocate/free memory

What does the OS do?

- Resource management: allocates hardware resources between programs

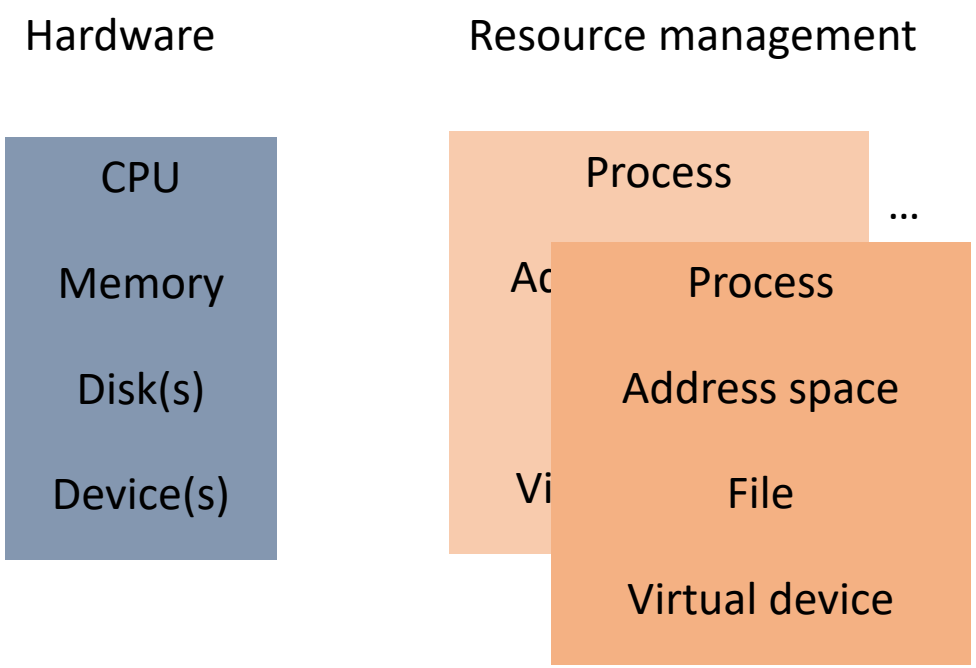
What does the OS do?

- Resource management: allocates hardware resources between programs



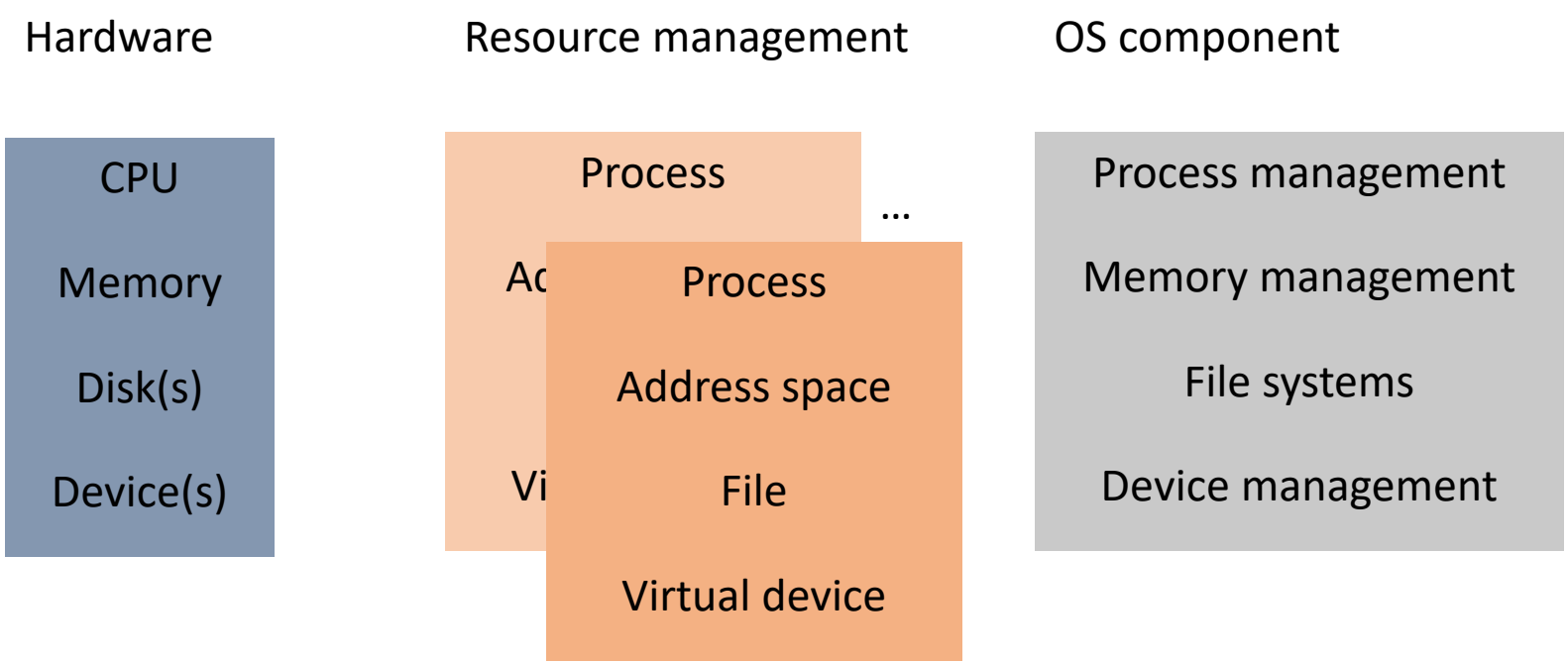
What does the OS do?

- Resource management: allocates hardware resources between programs



What does the OS do?

- Resource management: allocates hardware resources between programs



A Simple Example

- Many users want to compute

- OS allocates CPU to different users

Another Simple Example

- Many users want to use memory

- OS allocates memory between users

A Final Example

- Many files need to be stored on disk

- OS allocates disk space to files

What does the OS do?

- Abstraction: makes hardware easier to use
- Resource management: allocates hardware resources between programs

- **OS does *both* at the same time**

What Is and What Is Not in the OS

- Web browser?
- Graphics library?
- Device driver?
- Printer server?

What Is and What Is Not in the OS

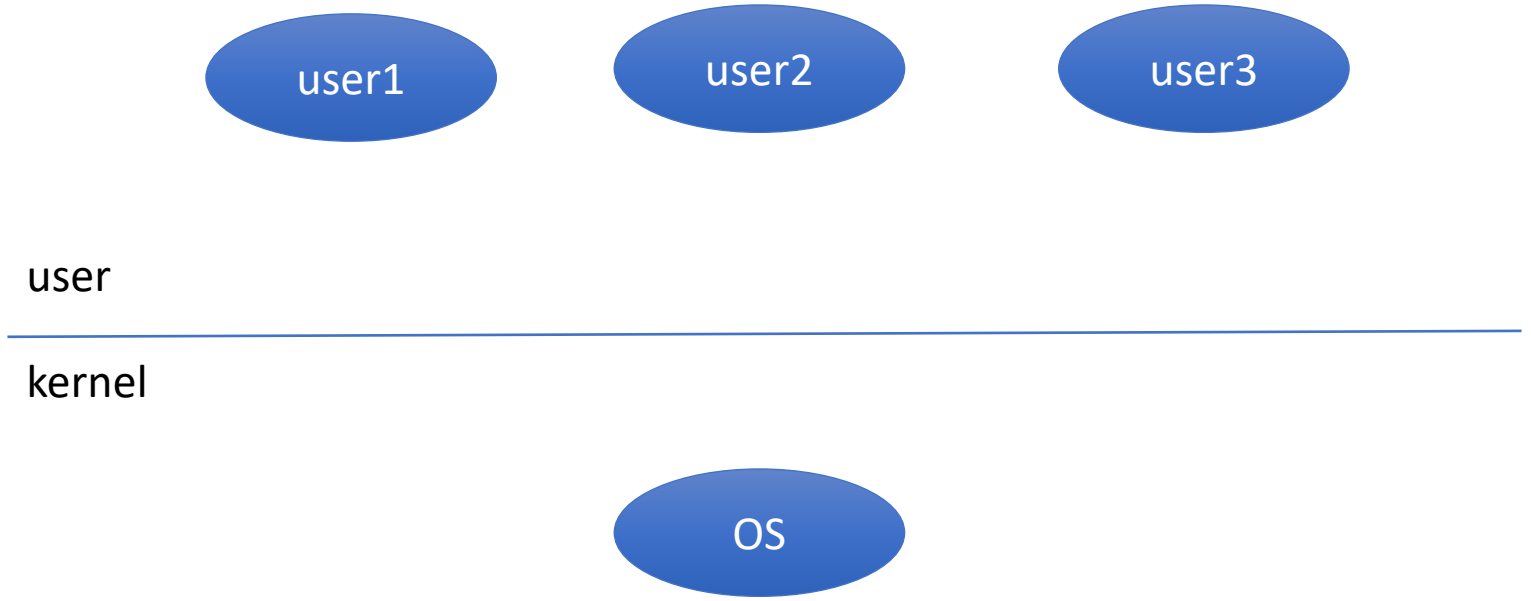
- Web browser: only abstraction
  - Not considered part of the OS
- Graphics library: only abstraction
  - Not considered part of the OS
- Device driver: both
  - Part of the OS
- Printer server: both
  - Part of the OS

Where does the OS live?

A Bit of Computer Architecture: CPU: Dual-Mode Operation

- Kernel mode vs. user mode
- Mode bit provided by hardware

User/OS Separation



Kernel Mode

- Privileged instructions:
  - Set mode bit
  - …
- Direct access to all of memory
- Direct access to devices

User Mode

- No privileged instructions:
  - Set mode bit
  - …
- No direct access to all of memory
- No direct access to devices

In General

- OS runs in kernel mode
- Applications run in user mode

- This allows OS
  - To protect itself
  - To manage applications/devices

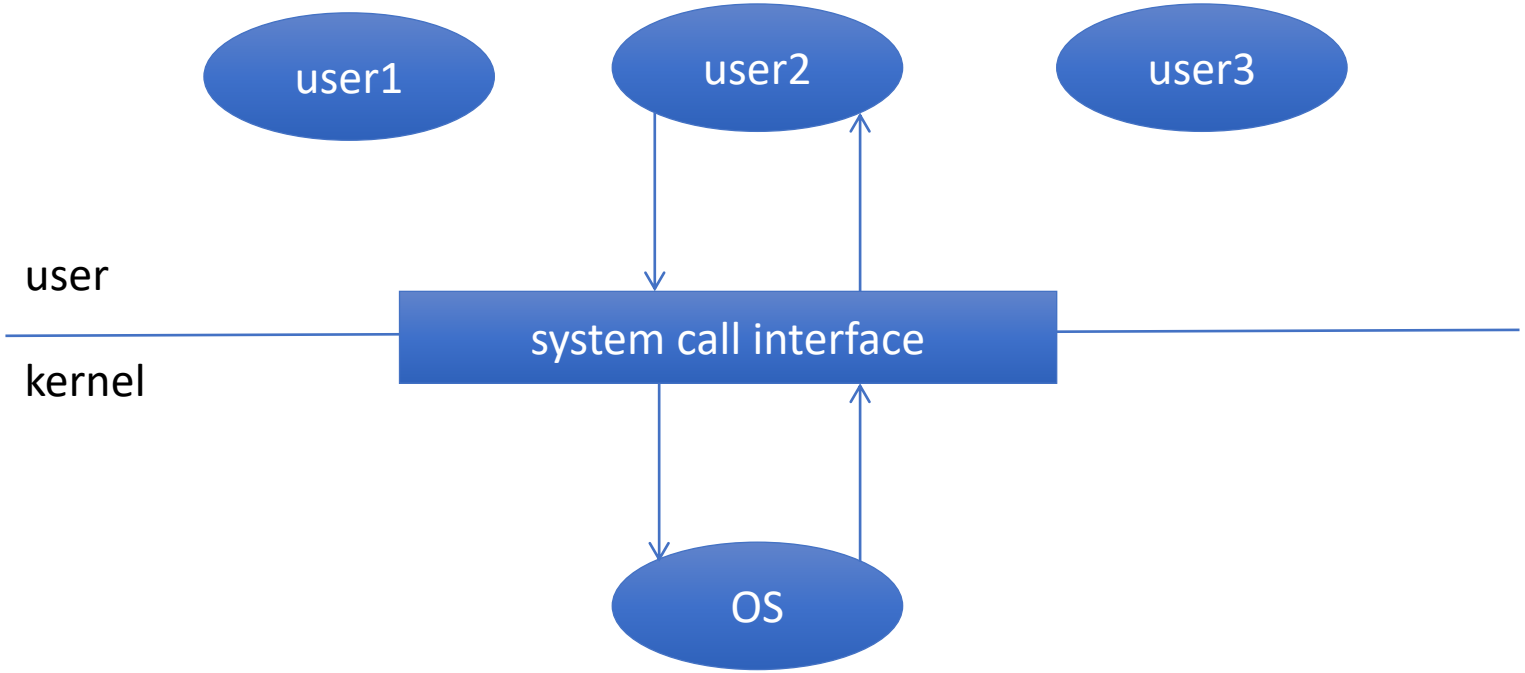
From Kernel to User Mode

- By the OS setting the mode bit to user
- Usually as a by-product of an instruction

From User to Kernel Mode

- By a device generating an interrupt
- By a program executing a trap or system call

System Calls:
Across User/Kernel Boundary



System Calls

- Are the *only* interface from program to OS
- Narrow interface essential for integrity of OS

System Calls in Linux?

System call number	System call name
0	restart_syscall
1	exit
2	fork
3	read
4	write
5	open
6	close
7	waitpid
8	creat
9	link
10	unlink
…	

System Calls in Linux?

System call number	System call name
…	
350	name_to_handle_at
351	open_by_handle_at
352	clock_adjtime
353	syncfs
354	sendmmsg
355	setns
356	process_vm_readv
357	process_vm_writev

System Call Implementation

- Architecture-specific

System Call Identification

- Unique system call number

System call number	System call name
0	restart_syscall
1	exit
2	fork
3	read
4	write
5	open
6	close
…	

To Perform a Given System Call

- Architecture-specific, example for x86
- Put system call number in register %eax
- Execute system call instruction

System Call Parameter Passing

- Again, architecture-specific

- Put in designated registers
- Put on the stack
- Put in table and have register point to it

In Linux/x86

- System call number in %eax register
- Parameters in registers
- If more parameters, register used as pointer

Question

- Ever called the OS?

Question

- Ever called the OS?
  - Yes, of course, e.g., any file system operation.
- Ever written a system call instruction?

Question

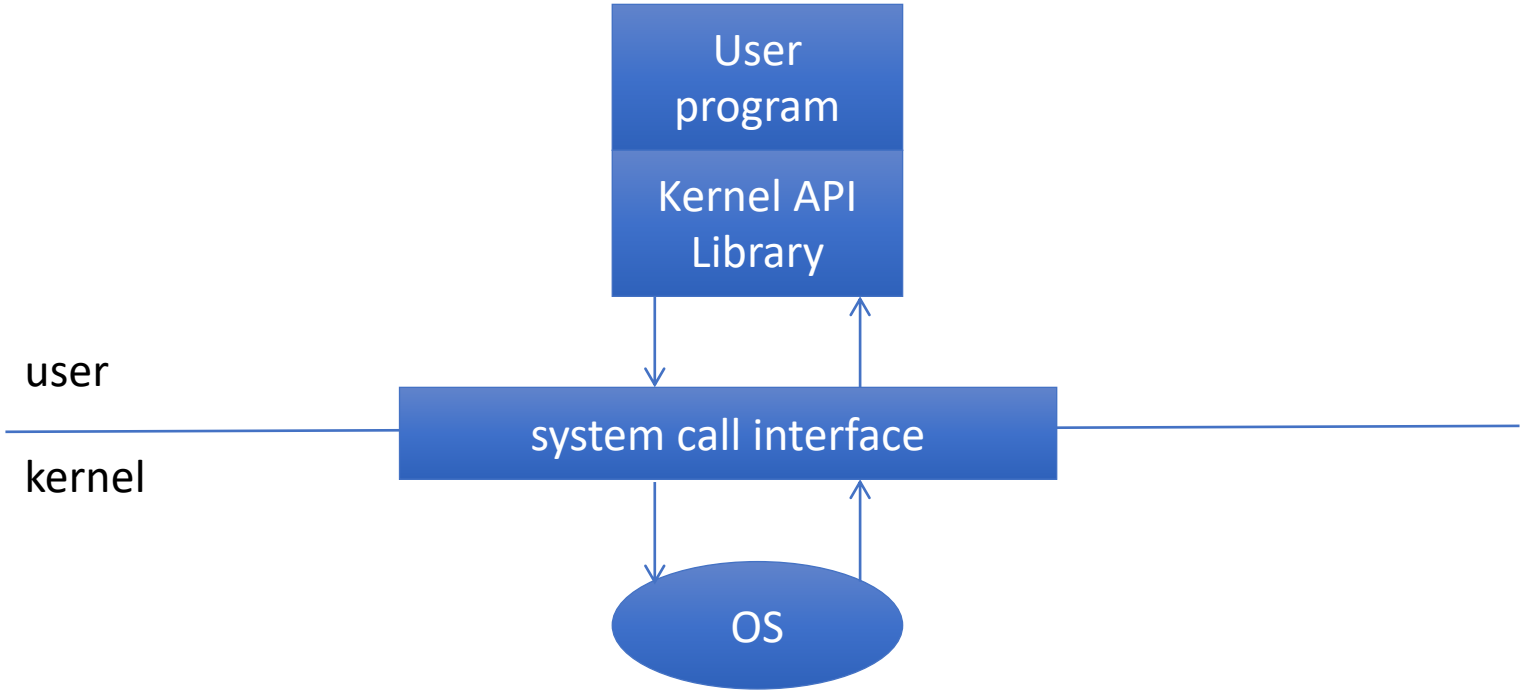
- Ever called the OS?
  - Yes, of course, e.g., any file system operation.
- Ever written a system call instruction?
  - I doubt it
- How so?

Answer: Kernel API

- A set of function calls that wrap system calls
- Easier to use
- More portable

- Example: Linux Kernel API

Kernel API


User program
Kernel API Library
user
kernel
system call interface
OS

Linux Kernel API

- Process management
  - fork(), exec(), wait(), …
- Memory management
  - mmap(), munmap(), sbrk(), …
- File system
  - open(), close(), read(), write(), …
- Device management
  - ioctl(), read(), write(), …
- Other examples
  - getpid(), alarm(), sleep(), chmod(), …

What Do Wrapper Functions Do?

- At the time of the call
  - Put arguments in registers
  - Put system call number in register %eax
  - Execute system call instruction
- At the time of the return
  - Take return value out of register
  - Return

Kernel API

```
main() {
        …
        write(…)
        …
}

write(…) {
        …
        execute system call instruction
        …
}
```

Question

- Ever called the OS?
  - Yes, of course, e.g., any file system operation.
- Ever written a system call instruction?
  - I doubt it
- Have you ever had to invoke the kernel API?

Question

- Ever called the OS?
  - Yes, of course, e.g., any file system operation.
- Ever written a system call instruction?
  - I doubt it
- Have you ever had to invoke the kernel API?
  - Maybe, maybe not

Answer: The Language Library

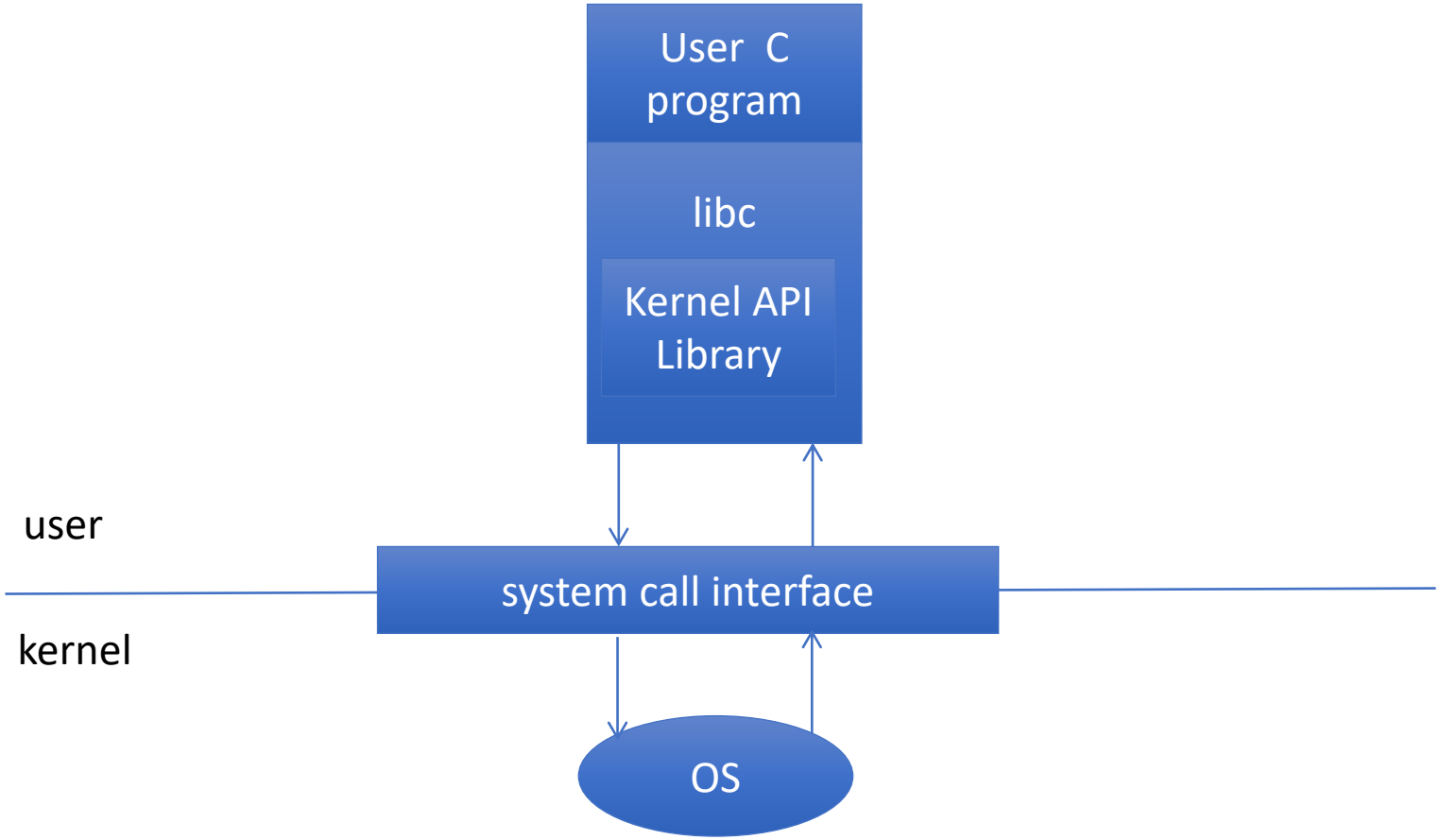
- A language-specific library
- Wraps the kernel API

- Classic example: the standard C library libc

libc

- printf, sprintf, fprintf, …
- getchar, putchar, …

libc


User C program
libc
Kernel API Library
user
kernel
system call interface
OS

libc

```
#include <stdio.h>
main() {
        …
        printf(…)
        …
}

printf(…) {
        …
        write(…)
        …
}

write(…) {
        …
        execute system call instruction
        …
}
```

Please Note!

- libc makes system call look like function call
- It is *not* a function call
- It is a user – kernel transition
  - From one program (user) to another (kernel)
  - Much more expensive

Traps

- Trap is generated by CPU as a result of error
  - Divide by zero
  - Execute privileged instruction in user mode
  - Illegal access to memory
  - …
- Works like an “involuntary” system call
  - Sets mode to kernel mode
  - Transfers control to kernel

Interrupts

- Generated by a device that needs attention
  - Packet arrived from the network
  - Disk I/O completed
  - …

OS Control Flow

OS Control Flow: Event-Driven Program

- Nothing to do

Do nothing

OS Control Flow: Event-Driven Program

- Nothing to do

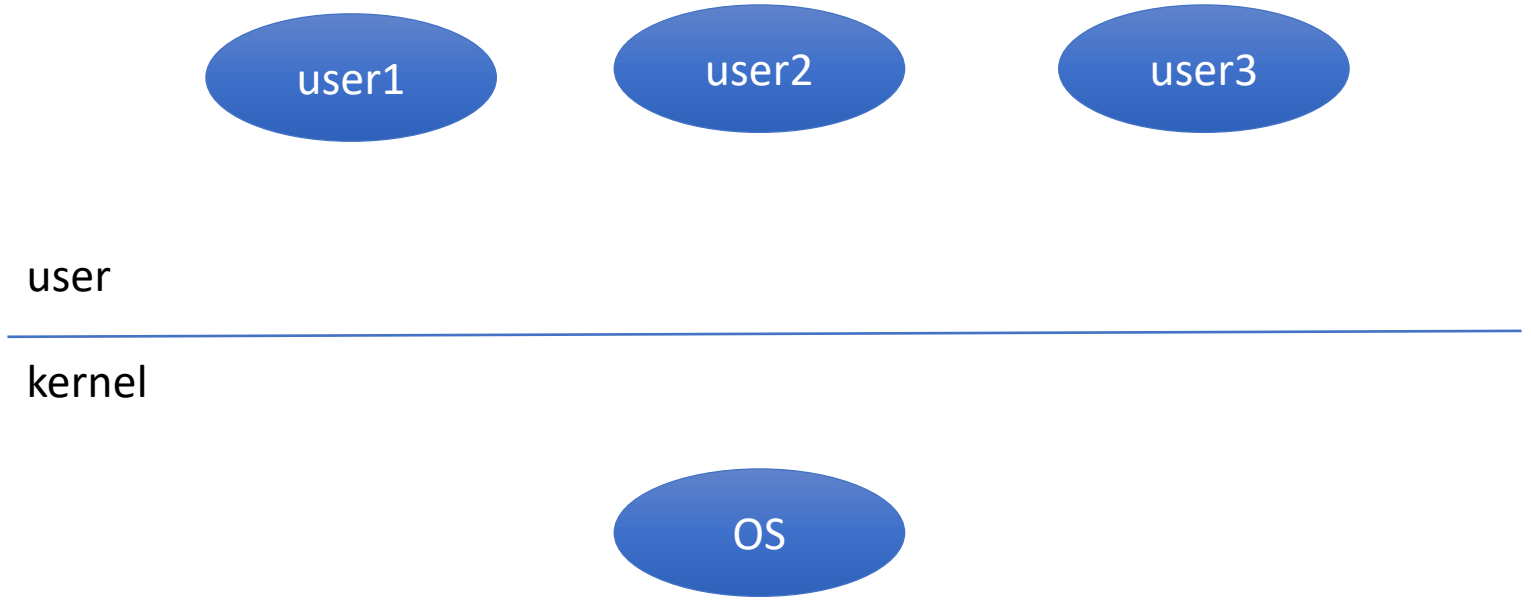
} Do nothing

- Interrupt (from device)
- Trap (from process)
- System call (from process}

} Start running

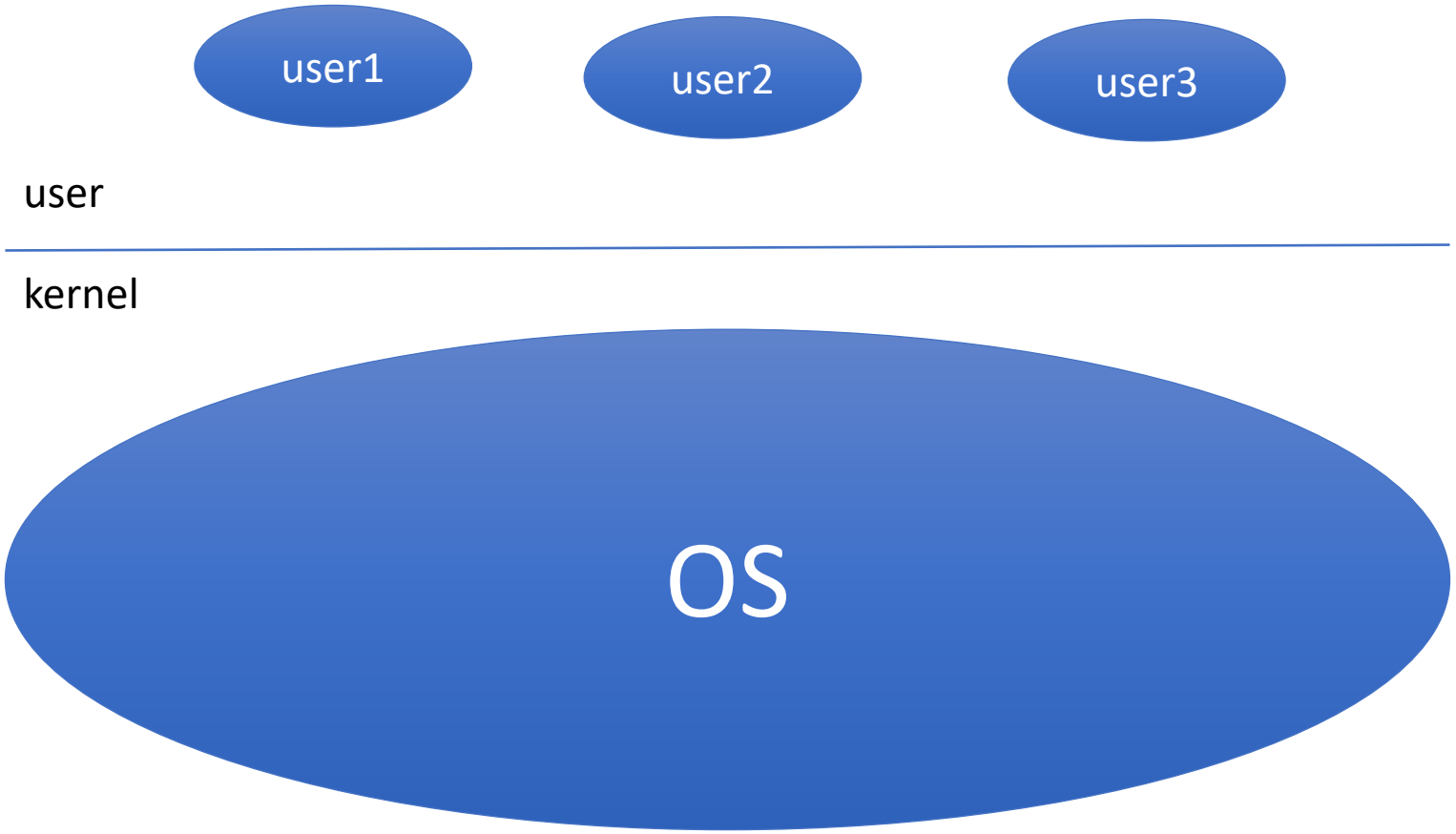
OS Structure

User/OS Separation



This approach is called the “monolithic OS”

It looks more like this



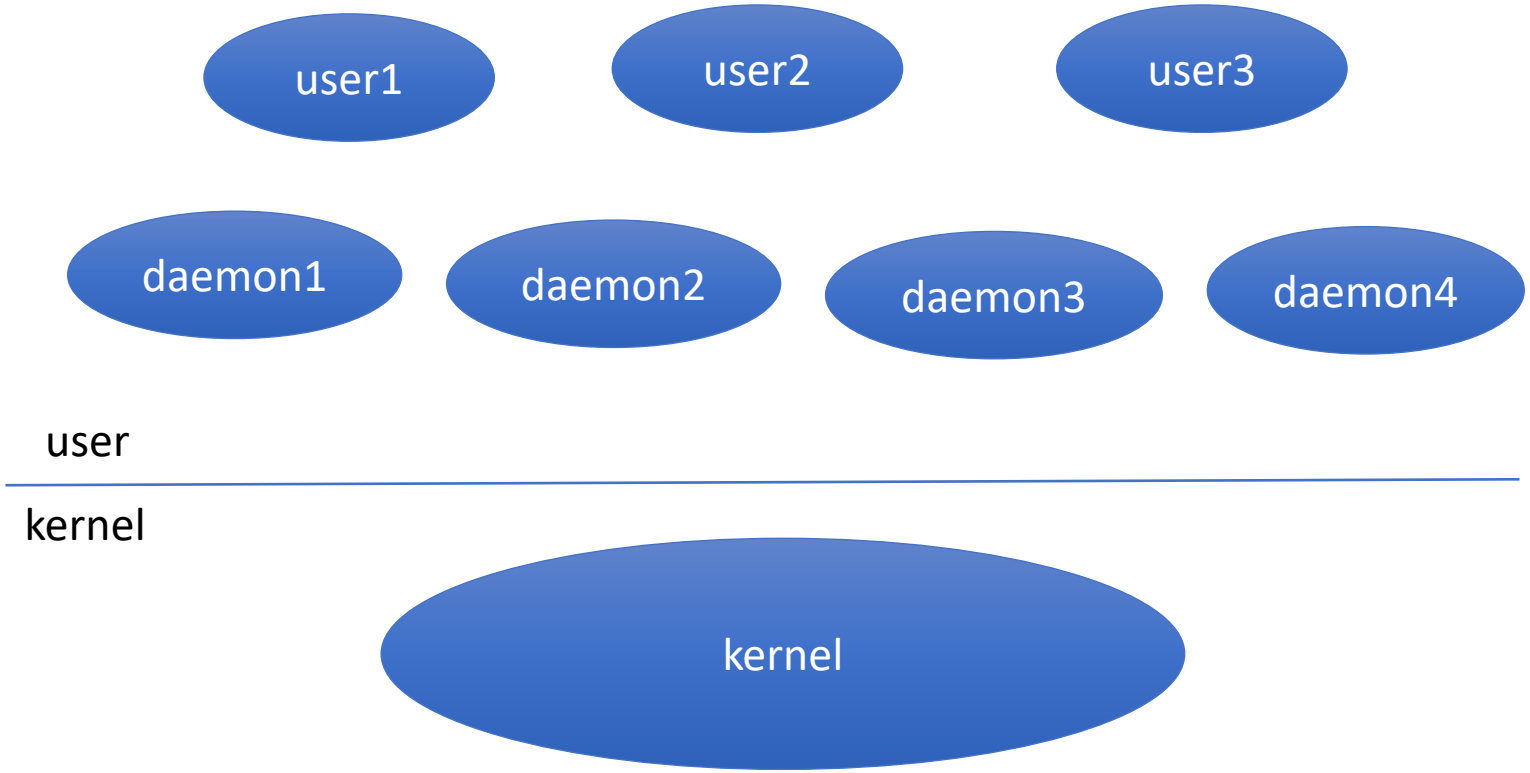
Downside of Monolithic OS

- The OS is a huge piece of software
  - Millions of lines of code and growing
- Something goes wrong in kernel mode
  - Most likely, machine will halt or crash
- Incentive to move stuff out of kernel mode

No need for entire OS in kernel mode

- Some pieces can be in user mode
  - No need for privileged access
  - No need for speed
- Example: daemons
  - System log
  - Printer daemon
  - Etc.

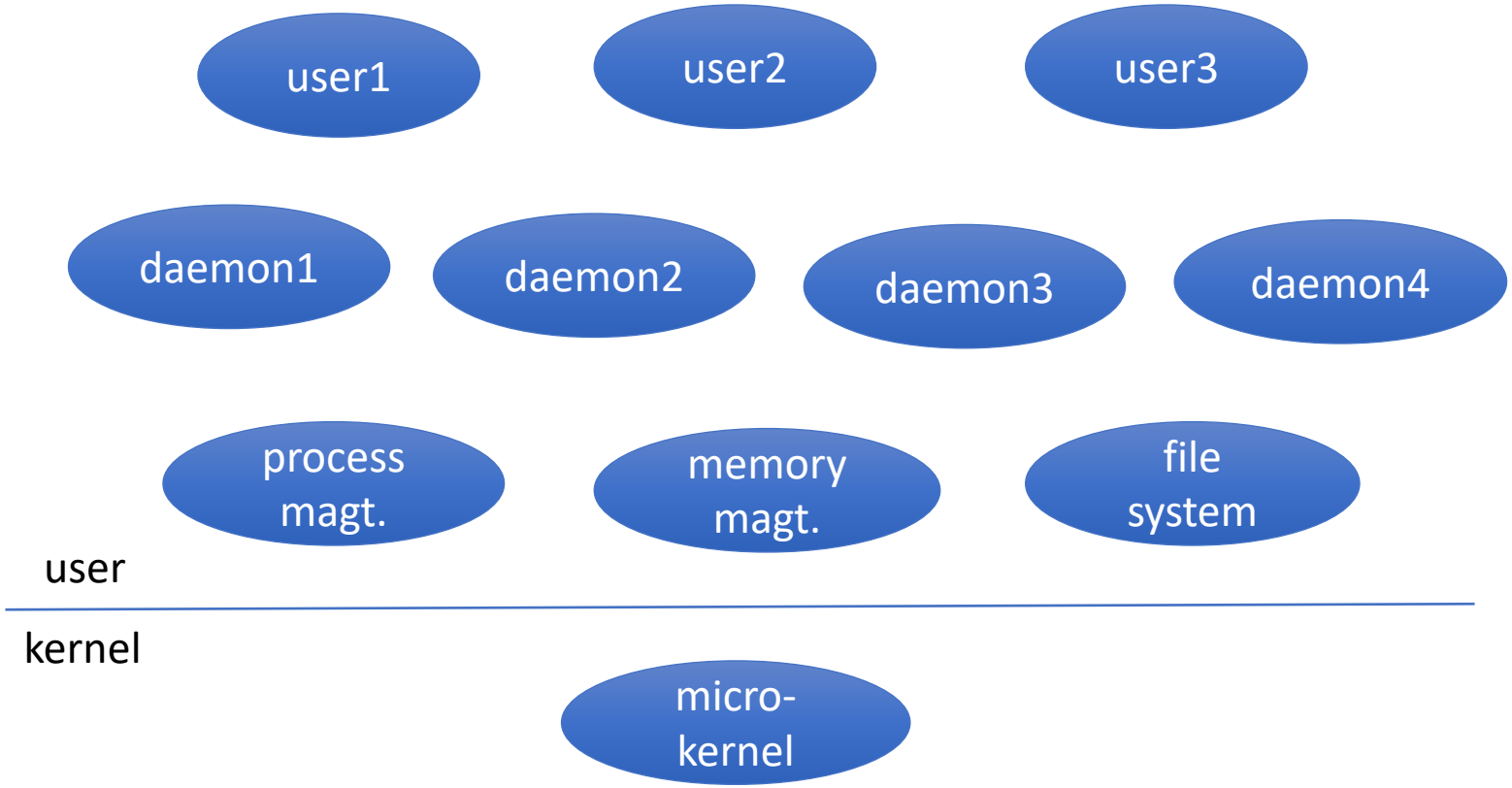
User/OS Separation: Systems Programs



The Ultimate Minimum: Microkernel

- Absolute minimum in kernel mode
  - Interprocess communication primitives
- All the rest in user mode

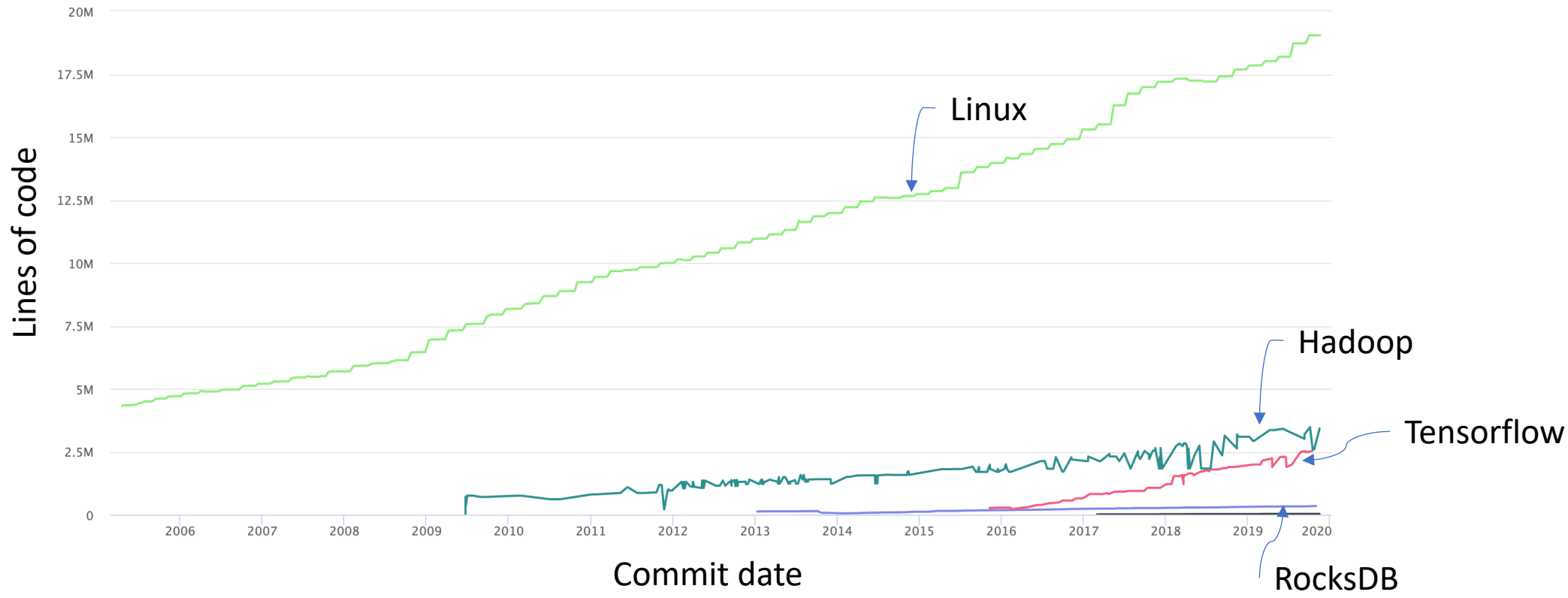
Microkernel



In Practice

- Microkernels have failed commercially
  - Except for niches like embedded computing
- The “systems programs” model has won out

The Price: Lines of Code in Linux Kernel



Source: https://panthema.net/2019/1122-Lines-of-Code-Plotted-over-Time/

Summary – Key Concepts

- What does the OS do?
  - Abstraction, resource management
- Where does the OS live?
  - User mode / Kernel mode
- OS interfaces
  - System call interface, Kernel API, Language library
- OS structure
  - Monolithic, microkernel

Further Reading

Operating Systems: Three Easy Pieces by R. & A. Arpaci-Dusseau

Chapter 2 https://pages.cs.wisc.edu/~remzi/OSTEP/

Credits:

Some slides adapted from the OS courses of Profs. Remzi and Andrea Arpaci-Dusseau (University of Wisconsin-Madison), Prof. Willy Zwaenepoel (University of Sydney), Prof. Youjip Won (Hanyang University), and Prof. Natacha Crooks (UC Berkeley)