

INF113: Introduction

Kirill Simonov

20.08.2025



Teaching Team

- **Lecturer:**

Kirill Simonov

kirill.simonov@uib.no

Høyteksenteret, Datablokken, 306P1

- **Teaching Assistant:**

Martin Feussner

martin.feussner@uib.no

- **Group Leaders:**

Sara Stavang Dahle

Fillip Sebastian Daae Lampe

Martin Styve Pedersen

Classes

- **Lectures:**

Wednesday, 10:15–12:00, Allégaten 66, Auditorium A

Friday, 8:15–10:00, Realfagbygget, Auditorium 1

This Friday only at Høyteksenteret, Stort Auditorium

- **Group sessions:**

1 - Monday 08:15-10:00, TM51 Room A

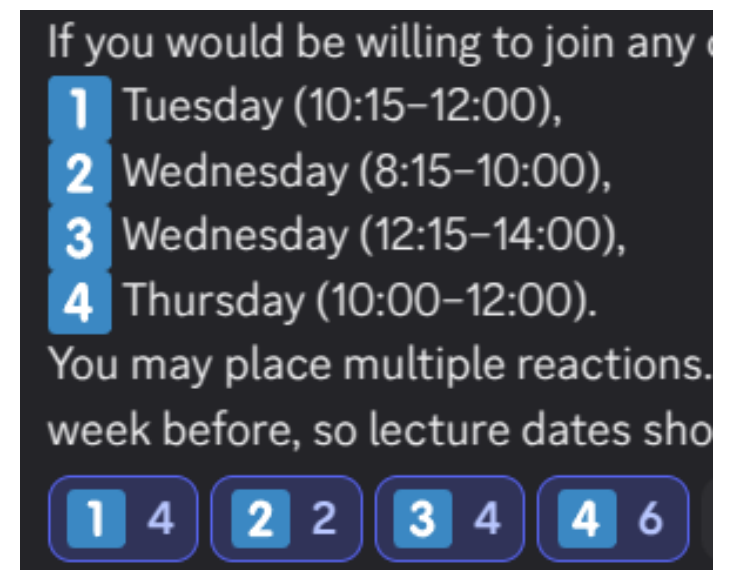
2 - Tuesday 08:15-10:00, TM51 Room A

4 - Thursday 08:15-10:00, TM51 Room D

5 - Friday 10:15-12:00, TM51 Room C

6 - Friday 12:15-14:00, Biologen Blokk B, K3+K4

On Discord, there is a poll for alternative times for group sessions—cast your vote!



Grading

10%	Assignment 1	12.09–26.09
10%	Assignment 2	10.10–24.10
10%	Assignment 3	31.10–14.11
70%	Final exam	21.11, 15:00

- Deliver via Mitt/CodeGrade
- Coding/experimentation tasks

You must pass all assignments and the exam!

- Digital exam with Inspira
- No support materials

The Book

- From Week 3 and onwards, we will follow the OSTEP book

Operating Systems: Three Easy Pieces

by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau
University of Wisconsin-Madison

- Available online: <https://pages.cs.wisc.edu/~remzi/OSTEP/>
- Also has code and simulators in the companion Git repo—run them!

Materials

- Course page on Mitt UiB: <https://mitt.uib.no/courses/53792>
 - **Homepage:** course info, week-by-week plan
 - **Modules:** complete set of materials for each week
Slides, book chapters, homework, additional resources
 - **Assignments:** Mandatory assignments will appear there
 - **People:** Sign up for a group for attending group sessions
- Discord server: <https://discord.gg/2DWNAd3MzC>
 - Announcements
 - General questions and discussions
 - Technical help
 - Questions about lecture material



Homework

- Every week there will be a small ungraded homework—look into **Modules** on Mitt
- The homework is covered on the group session next week
- Come to the session to resolve issues, discuss results, or just to make sure that you complete the homework by the session's end
- With homeworks, you learn by doing;
Mandatory assignments and the exam will be way easier
- This week: Set up a virtual machine with Ubuntu
Necessary for homeworks and assignments later on
- Next week: Practice programming in C
Compiling and running programs on your VM

LLMs

- It is **not allowed** to submit LLM-produced code or text in the assignments—be it from chatbots or IDE extensions like Copilot
- Naturally, such tools are not available on the exam either
- Upon submitting an assignment, you will be asked to declare any use of LLM/AI
- Normal UiB/faculty policy for academic integrity applies

Motivation: Overuse of LLMs is a fundamental problem for learning:

- Coursework is a sequence of relatively simple tasks
- Most individual tasks are likely to be also easy for an LLM
- This does not mean that completing them is meaningless: your goal is **gaining skill by solving**, not the tasks themselves

⇒ You should also not delegate too much to LLMs even on ungraded tasks!

LLMs: Example 1

Complete the following exercise:

"Start by entering a blank C program in the file:

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
int main(int argc, char *argv[]) {
```

```
    //TODO your code here
```

```
    return 0;
```

```
}
```

You are now ready to actually write the program. The program should read two integers from the standard input, and print their product to the standard output."

  < 2/2 >

Here's a completed version of the program that reads two integers from standard input and prints their product:

LLMs: Example 1

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    int a, b;

    // Read two integers from standard input
    if (scanf("%d %d", &a, &b) != 2) {
        fprintf(stderr, "Error: Failed to read two integers.\n");
        return 1;
    }

    // Print their product
    printf("%d\n", a * b);

    return 0;
}
```

Copy Edit

✓ This program:

- Uses `scanf` to read two integers from input.
- Validates that exactly two integers were read.
- Prints their product with `printf`.



LLMs: Example 1



Complete the following exercise:

"Start by entering a blank C program in the file:

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

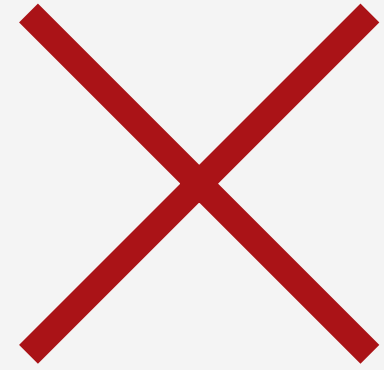
```
int main(int argc, char *argv[]) {
```

```
    //TODO your code here
```

```
    return 0;
```

```
}
```

You are now ready to actually write the program. The program should read two integers from the standard input, and print their product to the standard output."



📄 ✎ < 2/2 >

Here's a completed version of the program that reads two integers from standard input and prints their product:

LLMs: Example 2

How to read an integer with scanf in C?

In C, you can read an integer from the user using `scanf` like this:

c

 Copy  Edit

```
#include <stdio.h>

int main() {
    int number;
    printf("Enter an integer: ");
    scanf("%d", &number); // %d tells scanf to expect an int
    printf("You entered: %d\n", number);
    return 0;
}
```

LLMs: Example 2

Key Points:

1. Format Specifier:

- `%d` → for `int`
- `%ld` → for `long`
- `%lld` → for `long long`
- `%u` → for `unsigned int`

2. Address-of Operator (&):

You must pass the **address** of the variable to `scanf`, because it needs to modify the variable directly.

Example: `&number` instead of just `number`.

3. Return Value:

`scanf` returns the number of successfully read values. You can check it for input validation:

```
c                                                                    Copy Edit
if (scanf("%d", &number) == 1) {
    printf("Valid integer: %d\n", number);
} else {
    printf("Invalid input!\n");
}
```

👉 Do you want me to also show you how to **safely handle invalid input** (like when the user types letters instead of numbers), or just the basic usage?

LLMs: Example 2

How to read an integer with scanf in C?

In C, you can read an integer from the user using `scanf` like this:

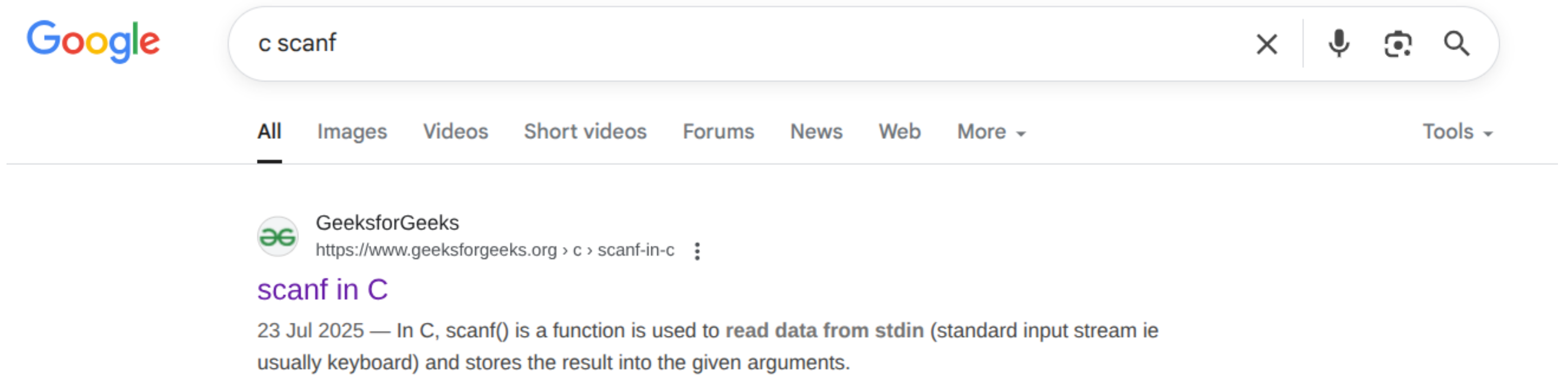
c

Copy Edit

```
#include <stdio.h>

int main() {
    int number;
    printf("Enter an integer: ");
    scanf("%d", &number); // %d tells scanf to expect an int
    printf("You entered: %d\n", number);
    return 0;
}
```

LLMs: Example 2



LLMs: Example 2

Syntax

The syntax of `scanf()` in C is similar to the syntax of `printf()`.

```
scanf("format", address_of_args... );
```



Parameters:

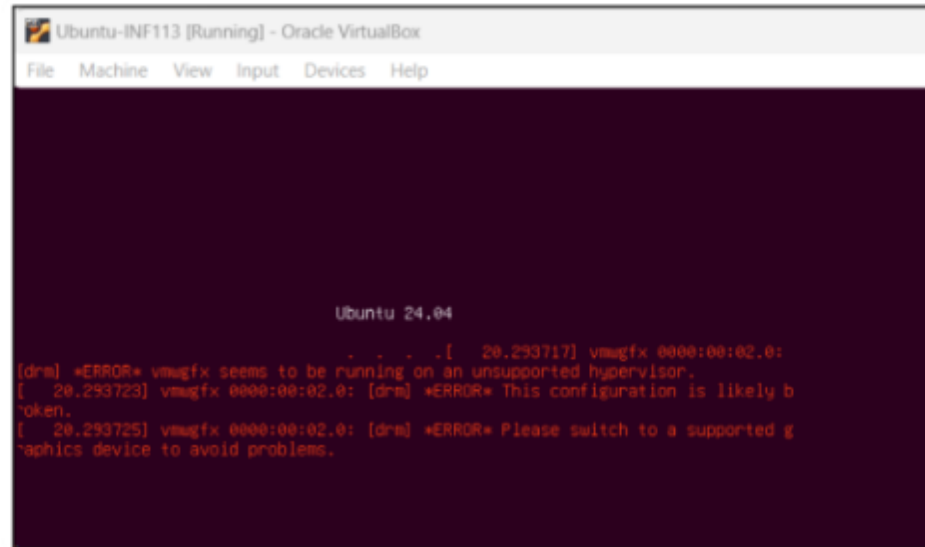
- **format:** It is the format string that contains the format specifiers(s).
- **address_of_args:** Address of the variables where we want to store the input.

Return Value:

- **>0:** The number of values converted and assigned successfully.
- **0:** No value was assigned.
- **<0:** Read error encountered or end-of-file (EOF) reached before any assignment was made.

We use & operator to find the address of the variables by appending it before the variable name and format specifier to recognize which type of data to be store.

LLMs: Example 3



Why do I get the following error upon installing Ubuntu on a VirtualBox machine?

The error you are seeing:

typo!

vbnet

Copy Edit

```
[drm] *ERROR* vmwgfx seems to be running on an unsupported hypervisor.  
[drm] *ERROR* This configuration is likely broken.  
[drm] *ERROR* Please switch to a supported graphics device to avoid problems.
```

happens because Ubuntu is trying to load the VMware graphics driver (`vmwgfx`) instead of the VirtualBox graphics driver (`vboxvideo`).

LLMs: Example 3

It does not solve the problem exactly though...

✓ Fix:

1. Shut down the VM in VirtualBox.
2. Go to **Settings** → **Display** → **Screen**.
3. Under **Graphics Controller**, select:
 - `VBoxSVGA` (recommended for modern Linux guests).
 - Or `VMSVGA` *with* Guest Additions installed.
- ⚠ Avoid `VBoxVGA` unless you need legacy compatibility.
4. Make sure **Enable 3D Acceleration** is **unchecked** (at least for the installation phase).
5. Start the VM again.
6. Once Ubuntu boots, install **VirtualBox Guest Additions**:

VBoxVGA
actually
solves
the issue

```
bash
```

Copy Edit

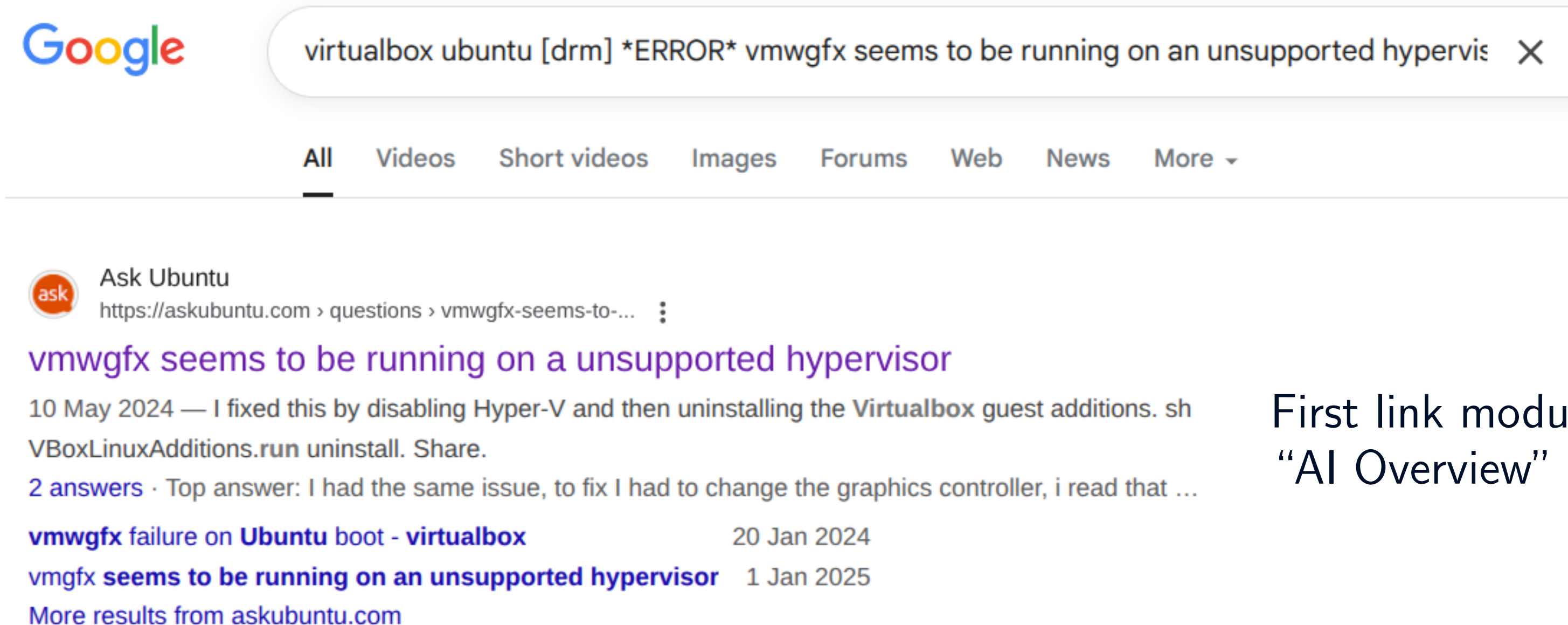
```
sudo apt update
```

```
sudo apt install virtualbox-guest-x11 virtualbox-guest-utils virtualbox-guest-dkms
```

LLMs: Example 3

Better options:

- Google the error message and find an actual person with the same problem



The screenshot shows a Google search interface. The search bar contains the text "virtualbox ubuntu [drm] *ERROR* vmwgfx seems to be running on an unsupported hypervis". Below the search bar, the "All" tab is selected. The first search result is from "Ask Ubuntu" with the URL "https://askubuntu.com > questions > vmwgfx-seems-to-...". The title of the result is "vmwgfx seems to be running on a unsupported hypervisor" in purple. The snippet below the title reads: "10 May 2024 — I fixed this by disabling Hyper-V and then uninstalling the Virtualbox guest additions. sh VBoxLinuxAdditions.run uninstall. Share." Below the snippet, it says "2 answers · Top answer: I had the same issue, to fix I had to change the graphics controller, i read that ...". At the bottom of the result, there are two links: "vmwgfx failure on Ubuntu boot - virtualbox" dated "20 Jan 2024" and "vmgfx seems to be running on an unsupported hypervisor" dated "1 Jan 2025". A link "More results from askubuntu.com" is also present.

First link modulo
“AI Overview”

LLMs: Example 3

Better options:

- Google the error message and find an actual person with the same problem

vmwgfx seems to be running on a unsupported hypervisor

Asked 1 year, 3 months ago Modified 7 months ago Viewed 63k times

12 So I want to install Ubuntu 24.04 in a virtual machine with VirtualBox 7.0 so I downloaded the iso from Ubuntu's website and put it in virtual machine, but when I boot and select "Install or Try Ubuntu" it shows a very primitive loading screen (instead of the fancy Ubuntu loading screen) with the text "Ubuntu 24.04" and in a couple of seconds it spits out this errors, and I can't install Ubuntu, the only way to have not this happen is to change from VMSVGA TO VBOXSVGA or VBOXVGA but those are crummy and flicker.

```
Ubuntu 24.04
. . . .[ 19.989993] vmwgfx 0000:00:02.0:
[drm] *ERROR* vmwgfx seems to be running on an unsupported hypervisor.
[ 19.990003] vmwgfx 0000:00:02.0: [drm] *ERROR* This configuration is likely b
roken.
[ 19.990008] vmwgfx 0000:00:02.0: [drm] *ERROR* Please switch to a supported g
raphics device to avoid problems.
```

I didn't have this happen to me in the past, and also that I used also VirtualBox 7, why is this happening

2 Answers

Sorted by: Highest score (default)

- 15 I had the same issue, to fix I had to change the graphics controller, i read that apparently vmwgfx happens to be the graphic drivers for VMware
- In settings > screen > set the 'graphics controller' to VBoxVGA or VBoxSGVA > make sure 'enable 3D acceleration' is unticked.
- Should now be able to boot up your system, make sure VirtualBox is up to date

Share Improve this answer Follow

answered Jun 13, 2024 at 6:18
 TrollLitsoMan
151 ● 3

LLMs: Example 3

Better options:

- Google the error message and find an actual person with the same problem
- Ask for help
 - On a group session
 - On Discord, #technical-help for this kind of issues

What is an OS?

Try for yourself!

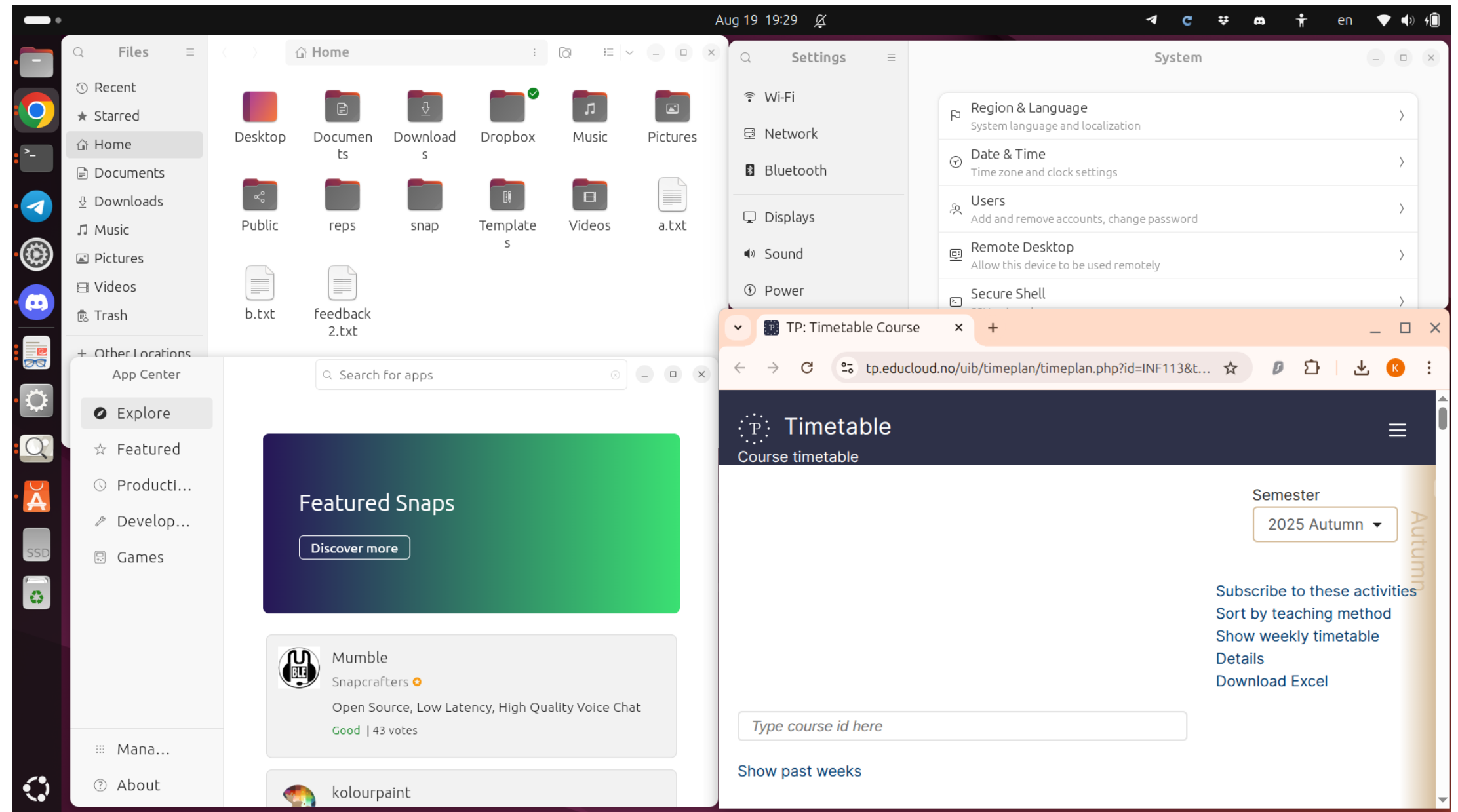
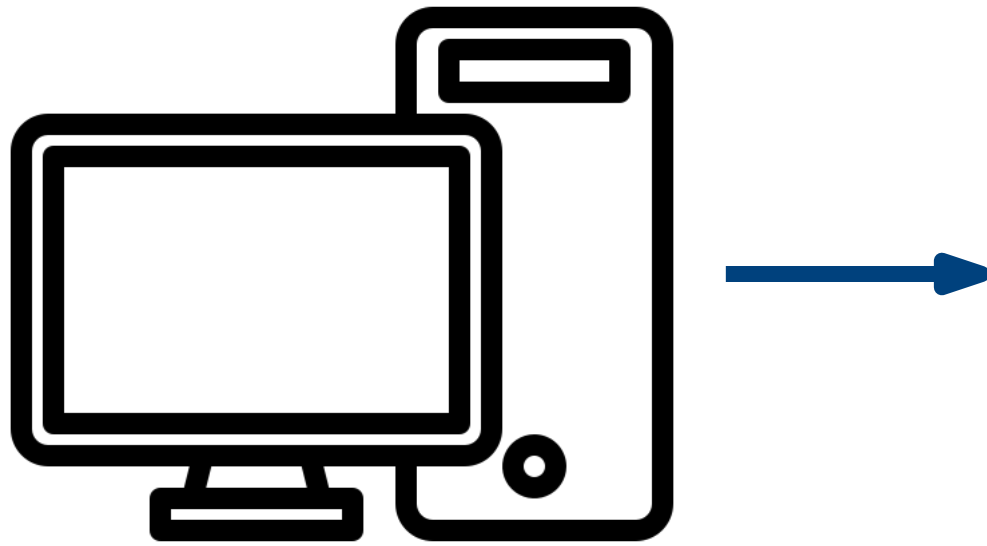
<https://plbrault.itch.io/youre-the-os>



Do you sometimes get frustrated at how slow your computer's operating system can be? Now is your chance to prove that you could do a better job!

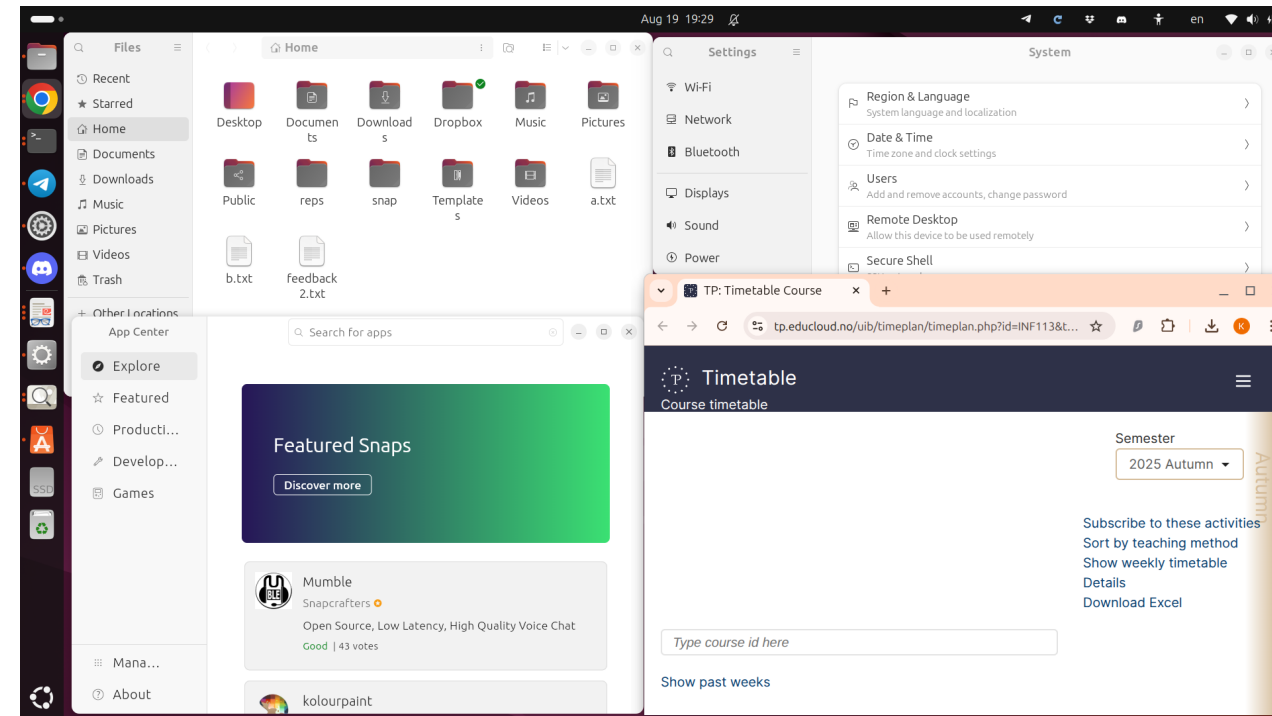
What is an OS?

- Commonly, by OS we mean *the whole thing*



User Interfaces

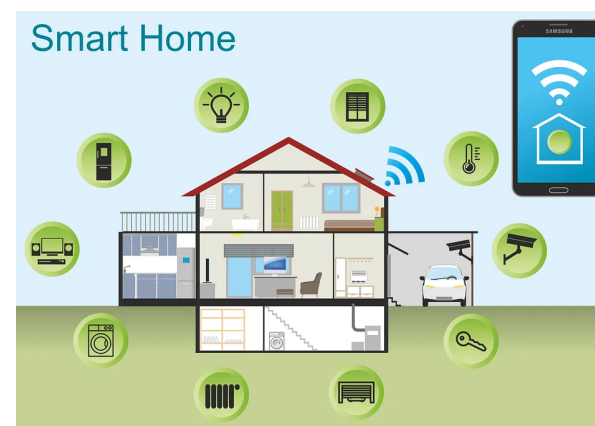
- Graphical User Interface (GUI)



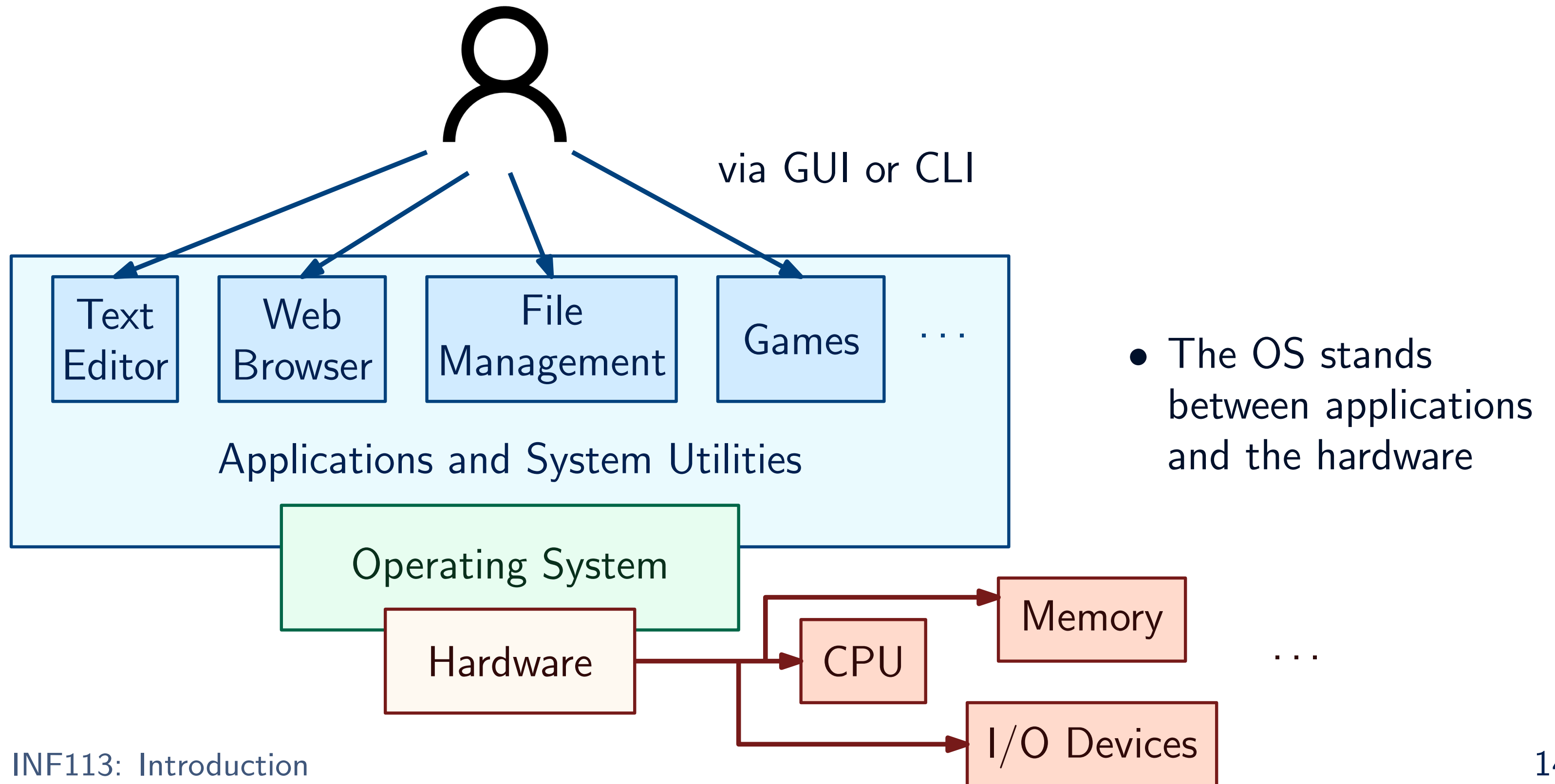
- Command-line Interface (CLI)

```
seemann@seemann-work-pc: ~  
seemann@seemann-work-pc:~$ cd reps/inf113-exercises/  
exercises/ .git/  
seemann@seemann-work-pc:~$ cd reps/inf113-exercises/  
exercises/ .git/  
seemann@seemann-work-pc:~$ cd reps/inf113-exercises/exercises/week02/  
seemann@seemann-work-pc:~/reps/inf113-exercises/exercises/week02$ ls  
1.in  atof.c  Makefile  mult.c  replace.c  replace.h  
seemann@seemann-work-pc:~/reps/inf113-exercises/exercises/week02$ cp 1.in 2.in  
seemann@seemann-work-pc:~/reps/inf113-exercises/exercises/week02$ rm 2.in  
seemann@seemann-work-pc:~/reps/inf113-exercises/exercises/week02$ as  
as                                aseqnet                aspell-import  
asciitopgm                       aspell                 asy  
aseqdump                         aspell-autobuildhash  
seemann@seemann-work-pc:~/reps/inf113-exercises/exercises/week02$ g++ mult.c -o  
mult -Wall  
seemann@seemann-work-pc:~/reps/inf113-exercises/exercises/week02$ ./mult  
1 2  
2  
seemann@seemann-work-pc:~/reps/inf113-exercises/exercises/week02$ cd ~  
seemann@seemann-work-pc:~$
```

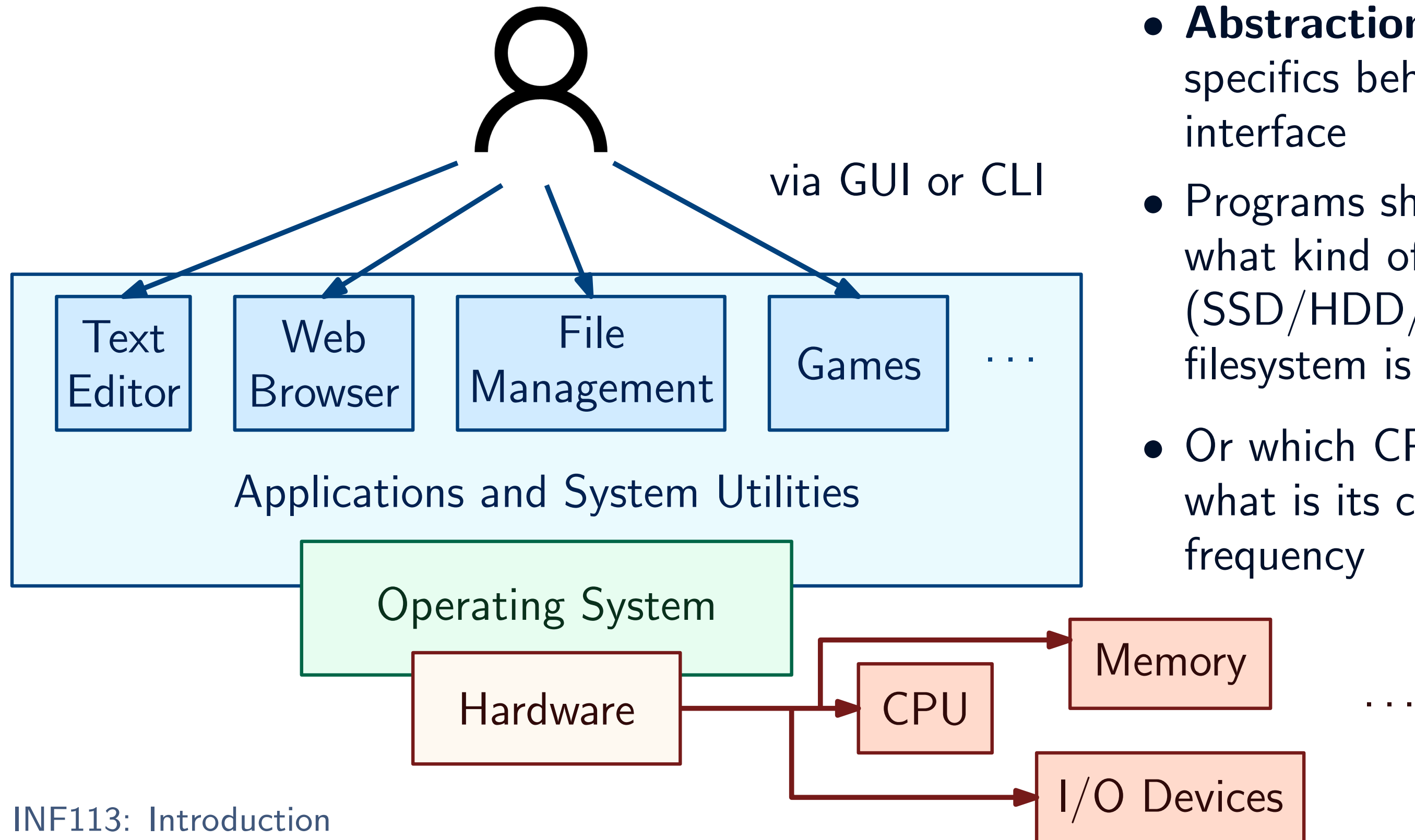
- Special interfaces



What should OS do?

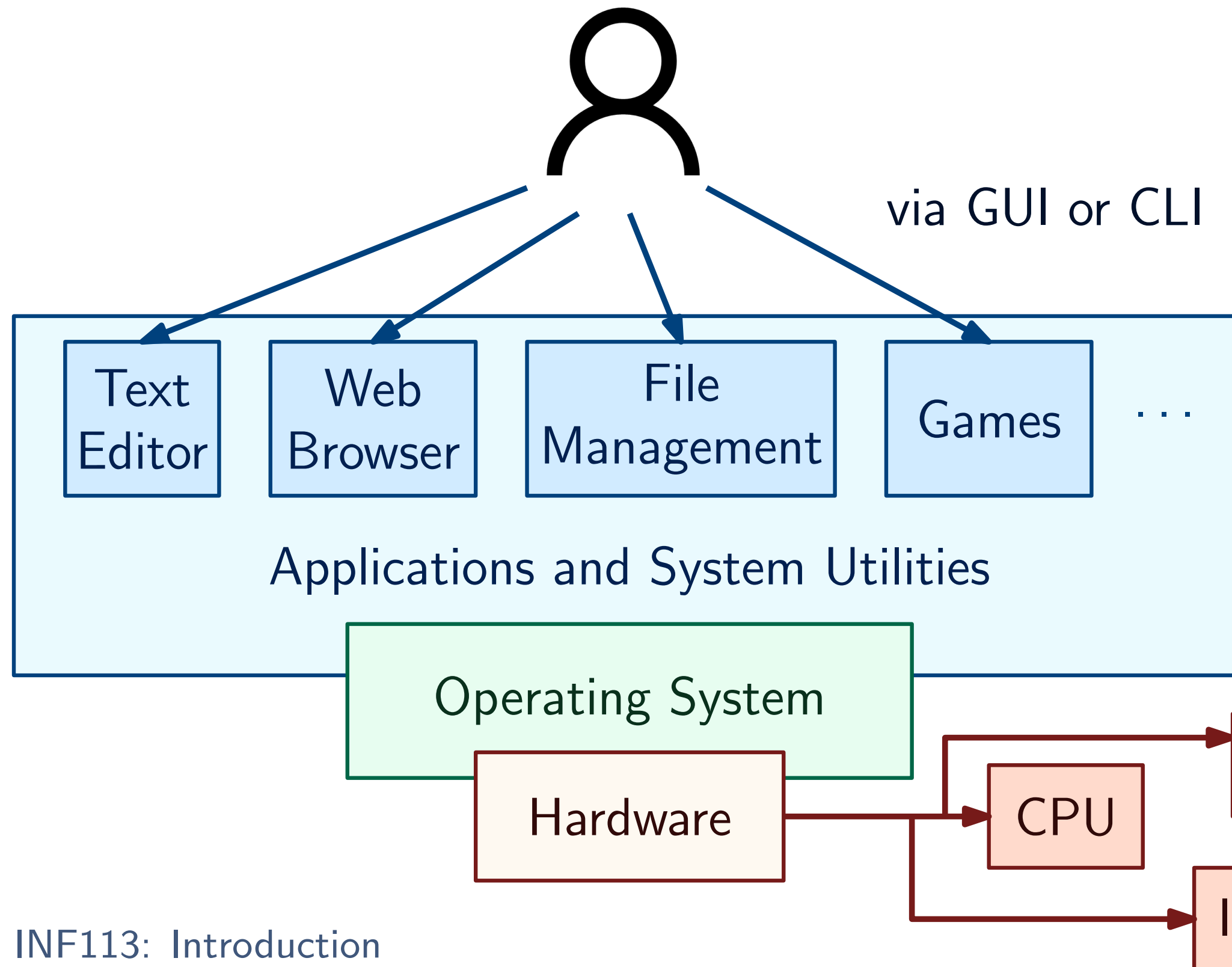


What should OS do?



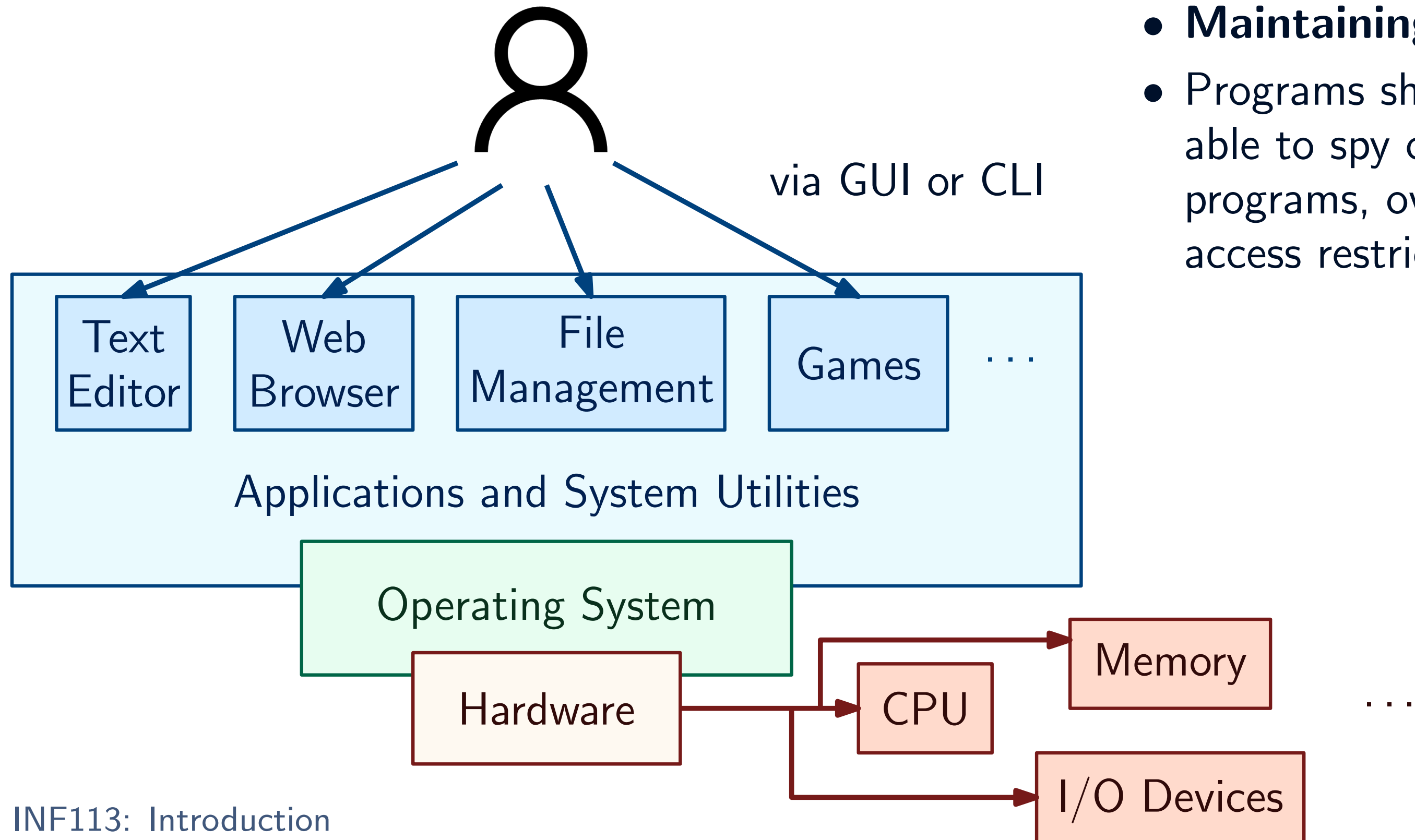
- **Abstraction:** hide device specifics behind a general interface
- Programs should not care what kind of a disk (SSD/HDD/...) or a filesystem is there
- Or which CPU model and what is its current frequency

What should OS do?



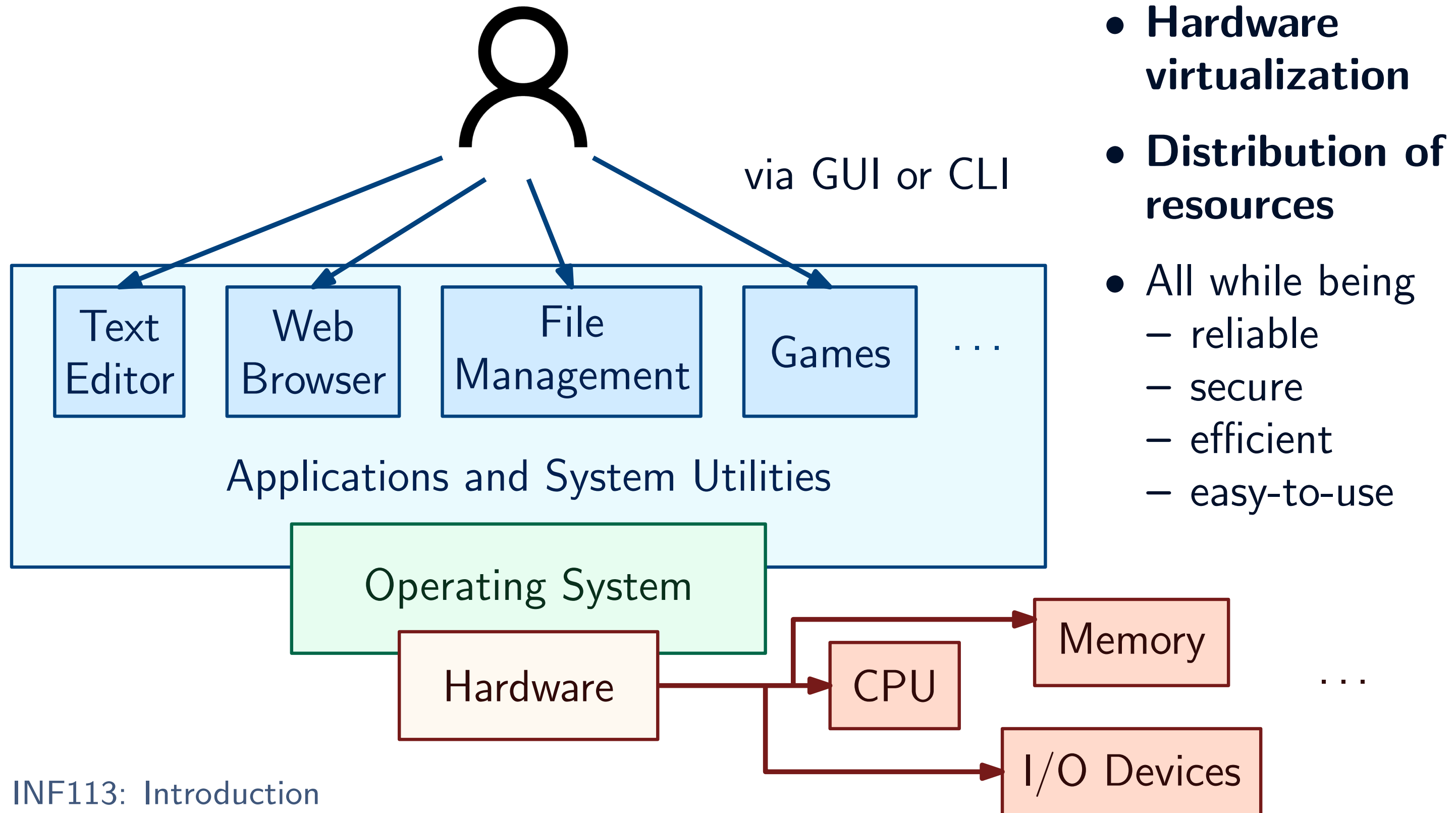
- **Distribution of resources**
- Programs run on the CPU “simultaneously”
- Each of them needs its own memory space
- They all should be able to write to disk without conflicts

What should OS do?

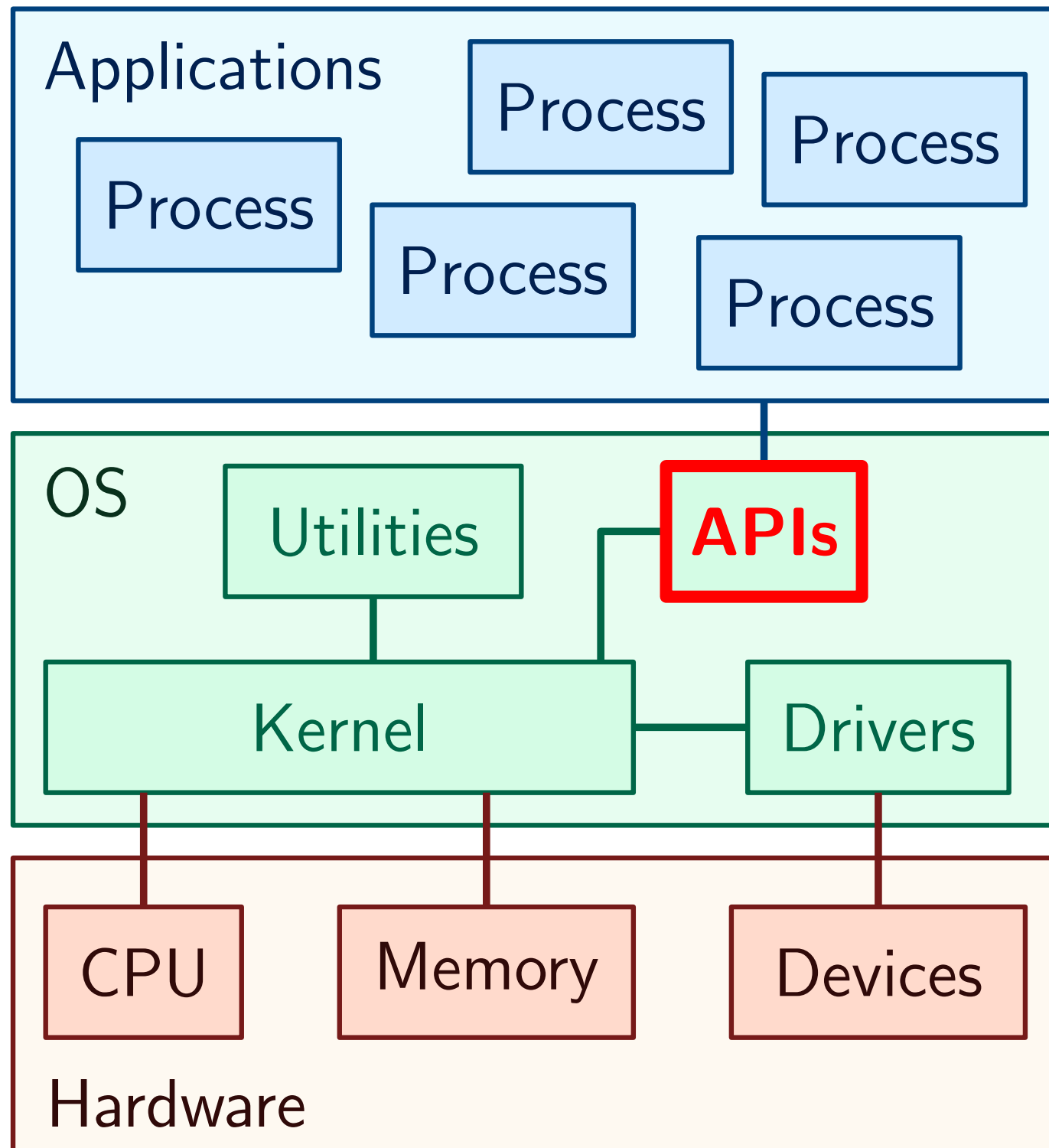


- **Maintaining security**
- Programs should not be able to spy on other programs, overcome access restrictions, etc.

What should OS do?

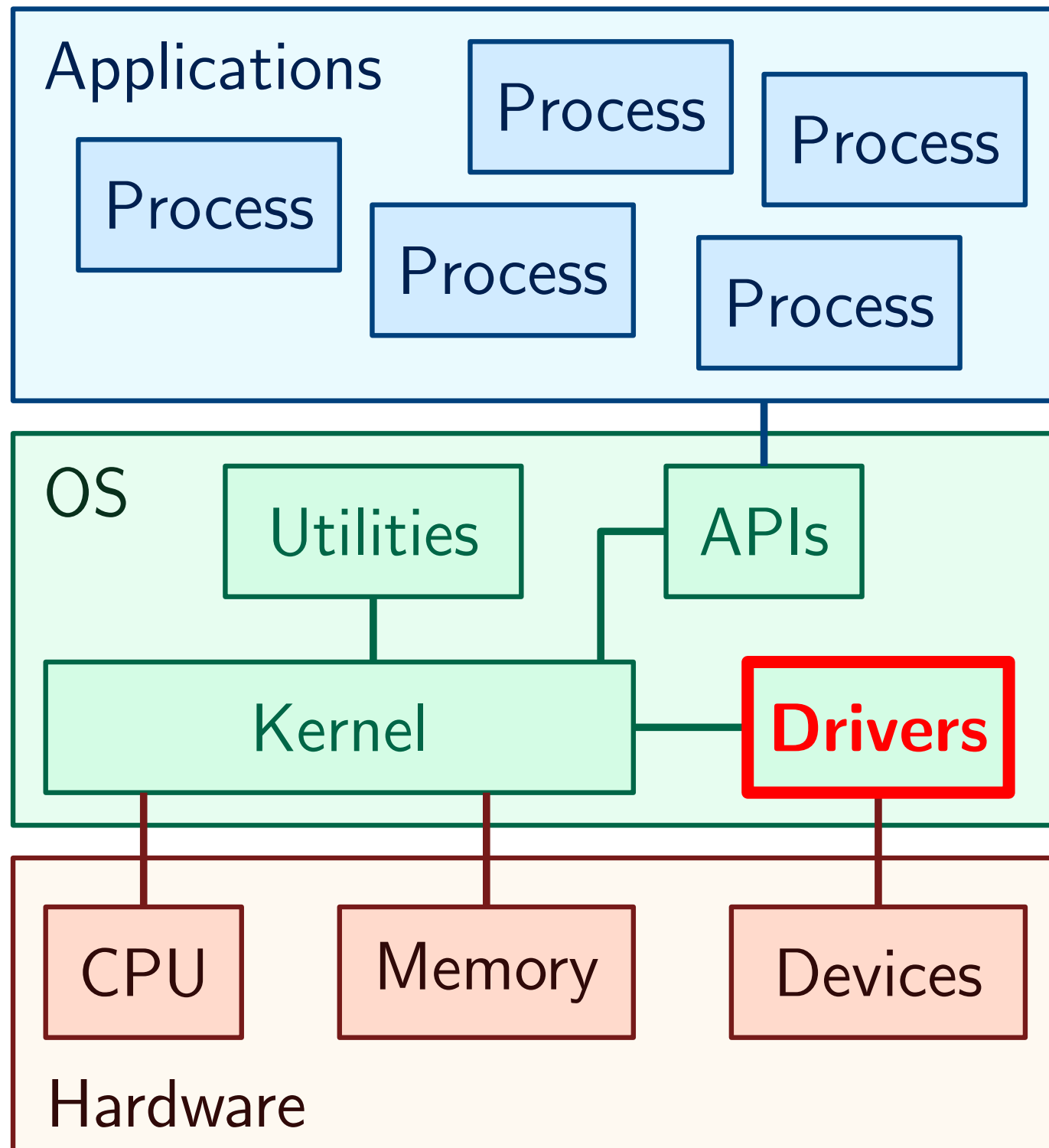


OS Components



- **Kernel**
 - Single program that is the core of the OS
 - Starts first when the system is booted
 - Runs other programs, manages resources
 - Provides access to devices
- **APIs**
 - Functions supplied with the OS
 - Programs may use them to receive services from the kernel
 - E.g., write to a file, reserve memory...
 - So-called *system calls*: program gives up execution to the kernel function

OS Components



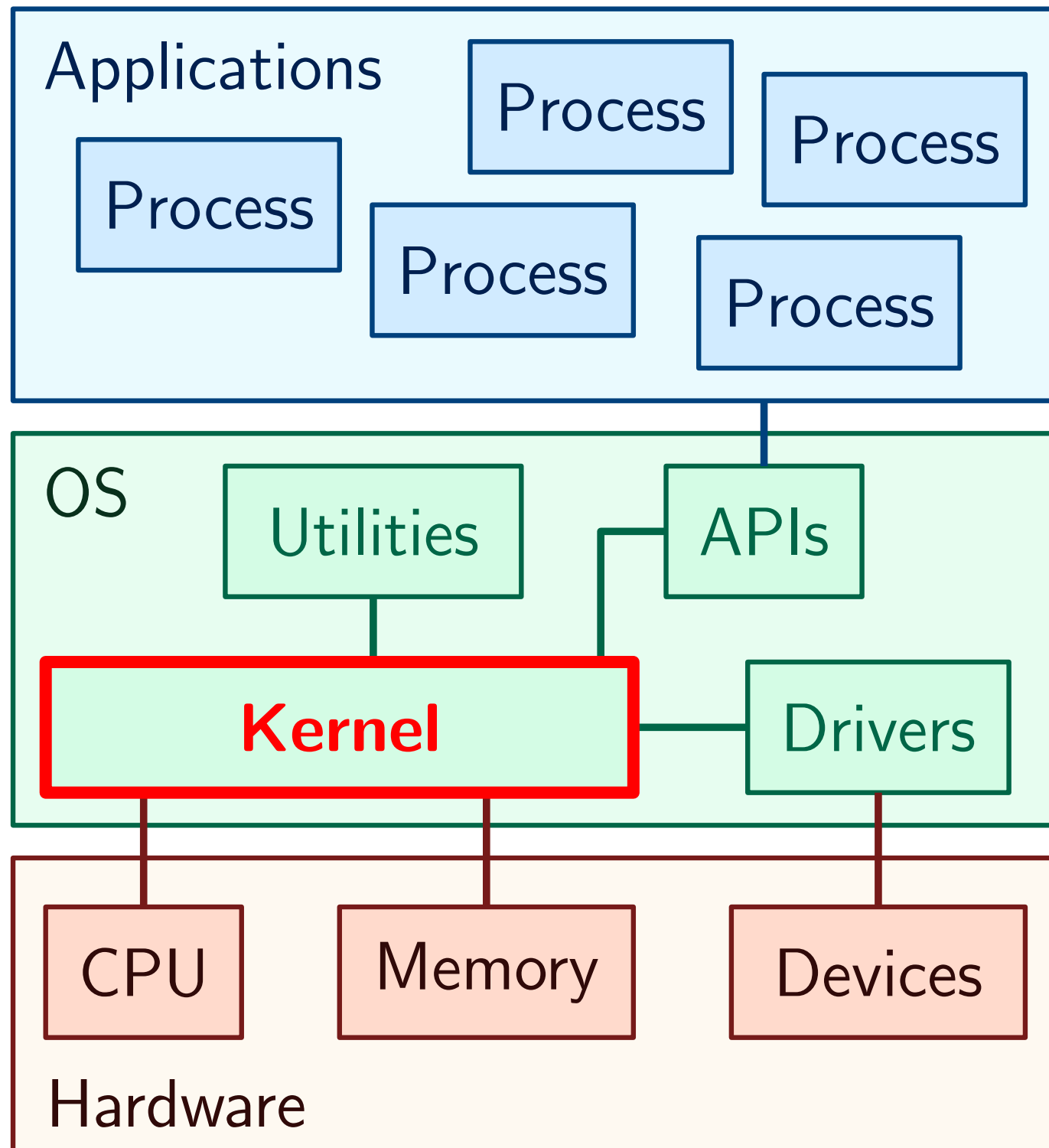
- **Utilities**

- Program tools provided with the OS
- Are more like normal programs
- E.g., window manager, network manager, terminal, ...

- **Drivers**

- Parts of the OS that interact with specific devices
- Graphics cards, Wi-Fi cards, printers, ...
- Usually can be installed/loaded dynamically, i.e., for the specific devices available

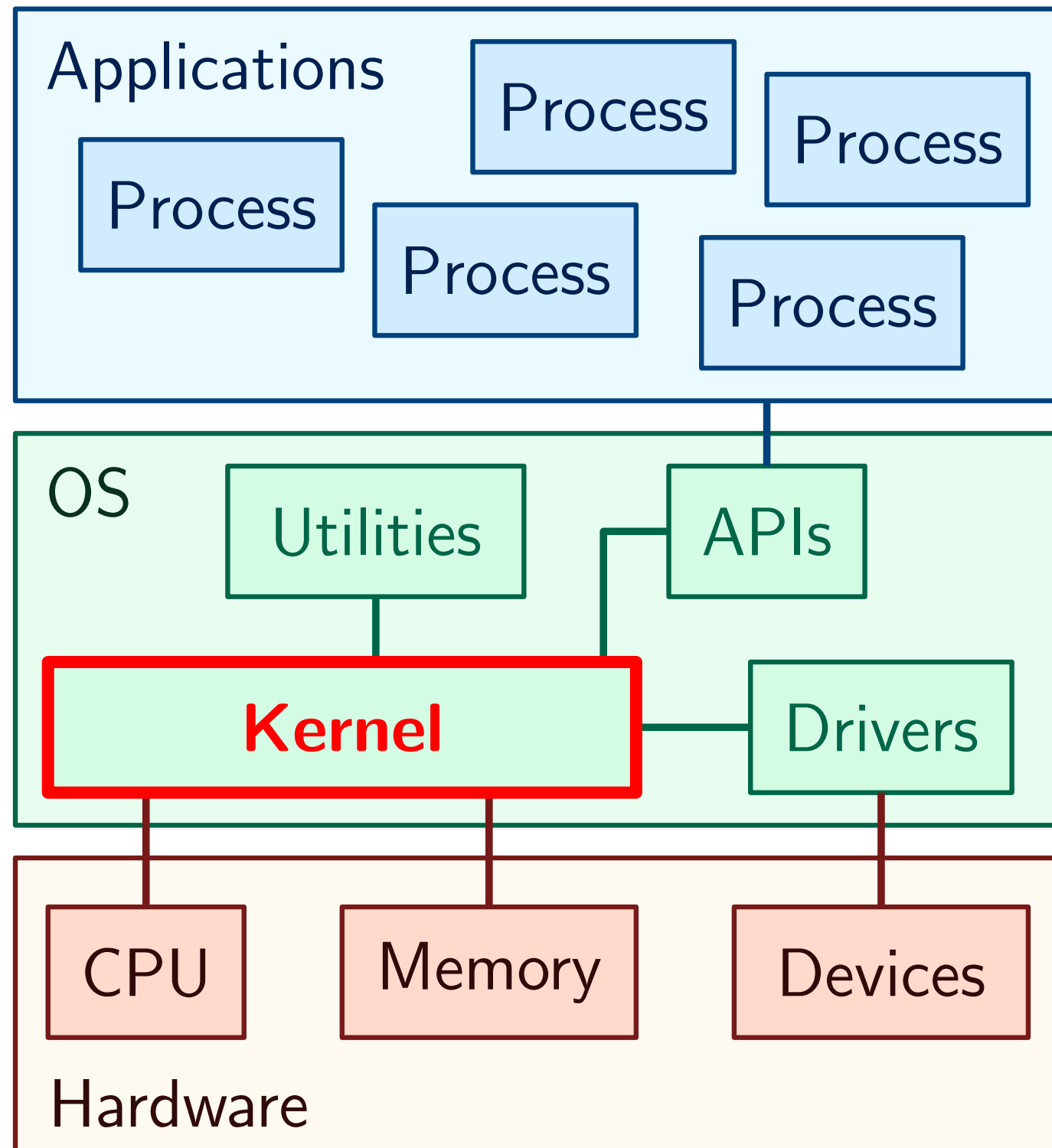
OS Components



- **Kernel**
 - Single program that is the core of the OS
 - Starts first when the system is booted
 - Runs other programs, manages resources
 - Provides access to devices

Kernel is our main focus
How does it manage all those things?

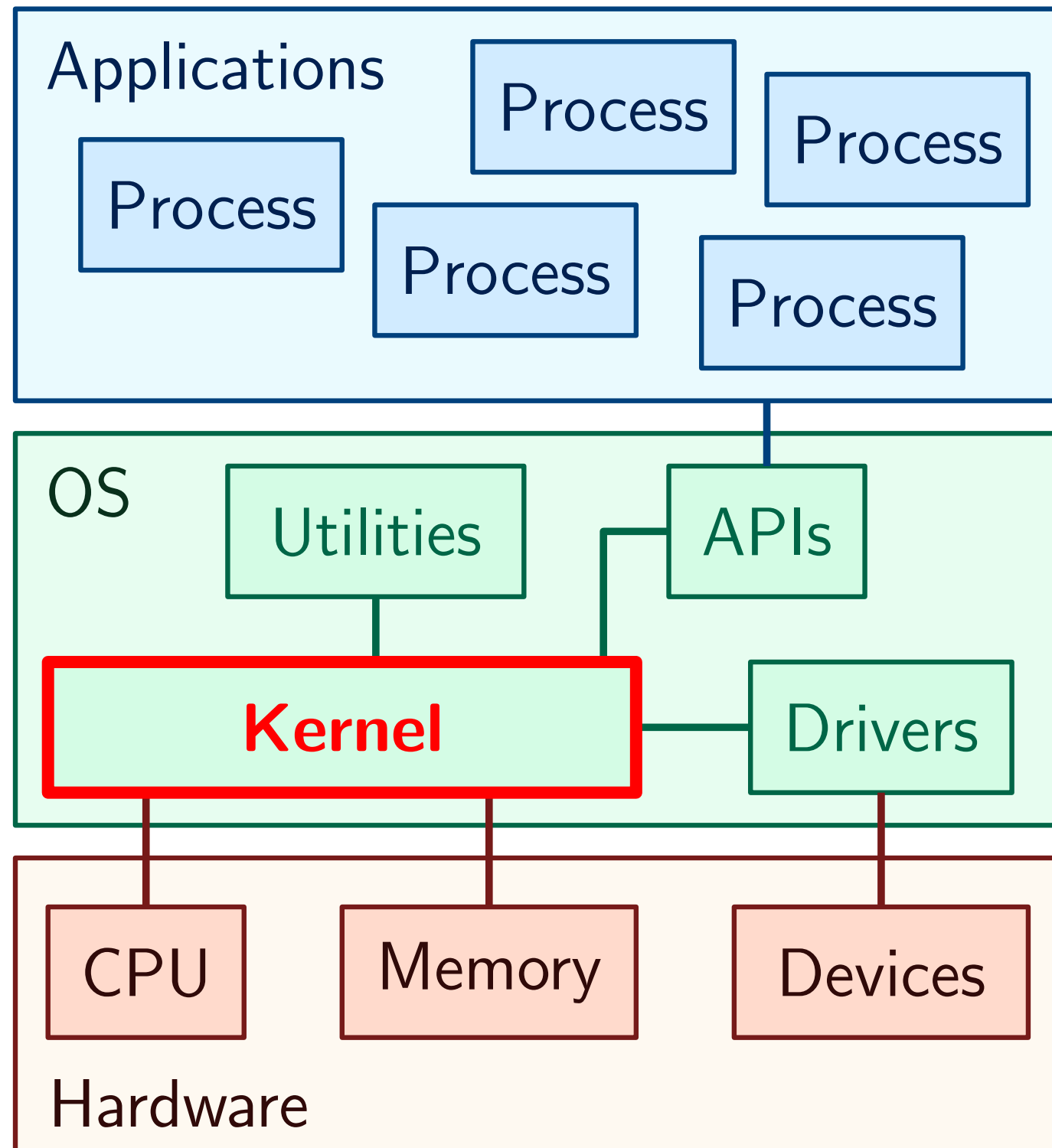
What we will cover



- **Topic 1: Processes**

- How to run the process in the “sandbox” so that it can still safely access the outside?
- How to run a multitude of processes “at the same time”?
- How to schedule processes “optimally” and “fairly”?

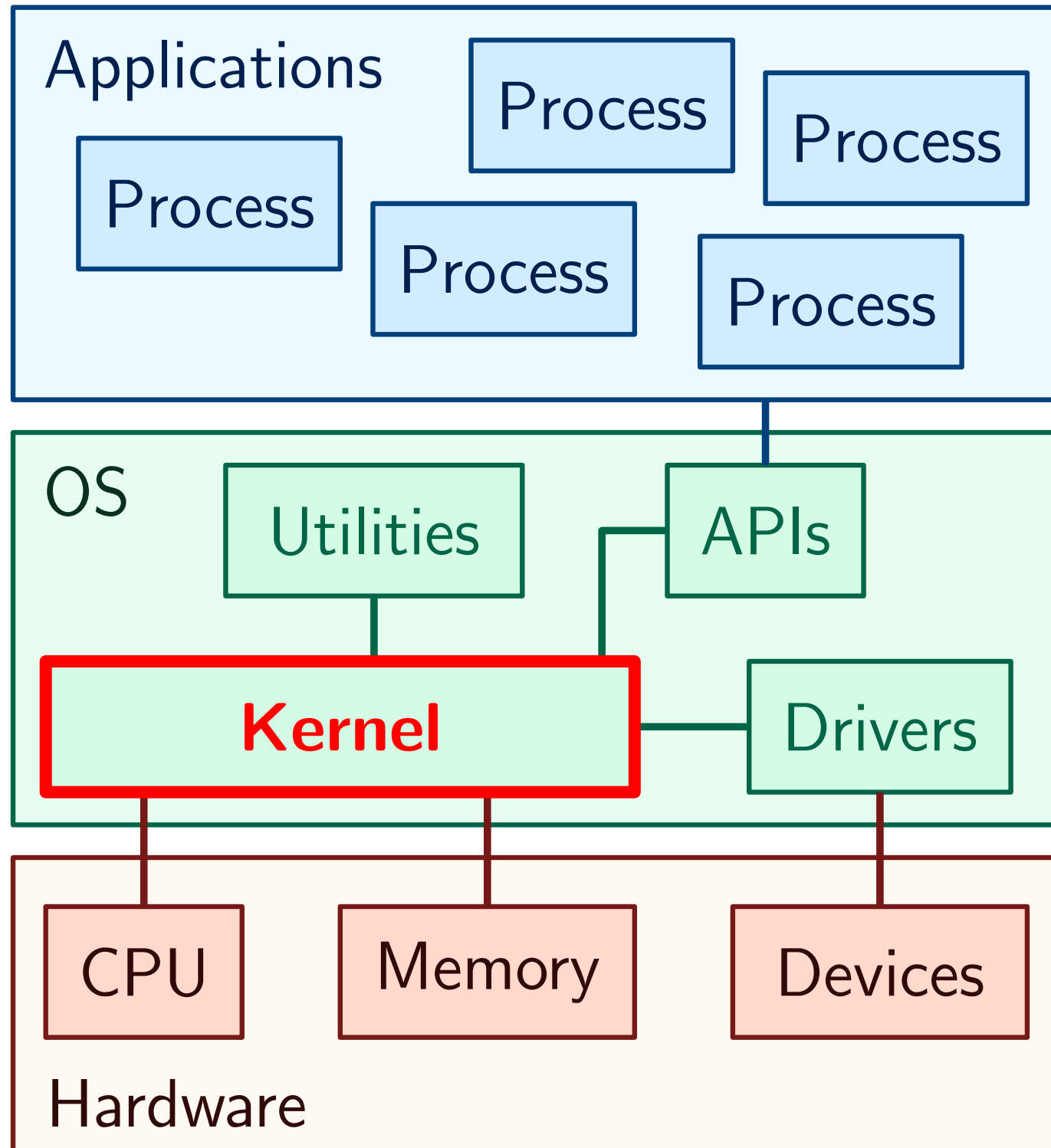
What we will cover



- **Topic 2: Memory**

- How does a program work with memory?
- How to provide a program with enough memory without prior knowledge?
- How to manage multiple programs sharing the same physical memory?

What we will cover



- **Topic 3: Concurrency and storage**
 - How can concurrent threads access and modify the same memory space without crashing?
 - How does one efficiently emulate a file system on the disk?
 - How to avoid data corruption in the presense of unreliability?

Why study OS?

- **Systems programming:** Basic requirement for working with low-level software like drivers and system utilities
- **High-performance computing:** Squeezing out more speed/storage efficiency by understanding how the processes and resources are managed
- **Deployment:** Modern deployment pipelines heavily use tools that heavily rely on OS isolation concepts: Docker, Kubernetes, etc
- **Low-level troubleshooting:** Issues like memory leaks, undefined behaviours, concurrency errors will become way more intuitive
- **Security:** Better idea of which vulnerabilities are feasible within the modern OS
- **Bonus:** You will get useful tools at your disposal as a by-product of the course

Tools: Ubuntu and Linux



Ubuntu

- Ubuntu/Linux is the perfect OS to study OS
 - Great kernel design, in terms of OS principles
 - Open-source and fully transparent
 - Ubuntu and other Linux distributions maintain easy access to low-level aspects of the OS
- But it is also a great system to work with
 - Streamlined package management
 - Terminal is the best
 - Everything can be configured
 - Great for automation
 - Few vulnerabilities and good reliability
- Almost exclusively, servers use Linux-based systems
 - Ubuntu skills transfer to other Linux distributions

but sometimes you are unlucky with the drivers :)

Tools: CLI and Bash

- Command-line tools are great!
- Direct and flexible integration with everything that is there in the OS
- Not limited by what was chosen to fit in the window of a GUI app
- Easy automatization, also for everyday tasks
 - Change extension for 100 files
 - Compress/modify metadata for 50 000 photos from your vacation
 - Move all large files to an external hard drive
- Sometimes you just have to use command-line: SSH to a remote server, running commands on a Docker container, your window manager crashes
- Tutorial: <https://swcarpentry.github.io/shell-novice/01-intro.html>, also other resources in **Modules** on Mitt

Tools: C

- C is the “closest to the machine” language
- Great for systems programming and high-performance
- But also notorious for “shooting yourself in the foot”
- Becomes super-powerful upon accepting OOP—meet C++!
 - Widely used for resource-intensive apps, high-performance computing, game development
 - Having experience with C helps a lot, especially with troubleshooting
 - Drags some compatibility issues, but is constantly modernized
- We will have a brief tutorial on Friday
 - Next homework is focused on programming in C
- Interactive online tutorial: <https://www.learn-c.org/en/Welcome>

