

# Introduction

Johan Montelius

KTH

2021

# What is an operating system?

# What is an operating system?

Abstraction, virtualisation and managing of resource.

# What is an operating system?

Abstraction, virtualisation and managing of resource.

# What is an operating system?

Abstraction, virtualisation and managing of resource.

- Abstraction

# What is an operating system?

Abstraction, virtualisation and managing of resource.

- Abstraction
  - How do we create an abstraction layer that provides an environment for programming of a process?

# What is an operating system?

Abstraction, virtualisation and managing of resource.

- Abstraction

- How do we create an abstraction layer that provides an environment for programming of a process?

- Virtualisation

- How do we create the image of dedicated hardware while in fact we have several process sharing the same hardware?

# What is an operating system?

Abstraction, virtualisation and managing of resource.

- Abstraction
  - How do we create an abstraction layer that provides an environment for programming of a process?
- Virtualisation
  - How do we create the image of dedicated hardware while in fact we have several process sharing the same hardware?
- Resource management
  - Given that we have limited amount of resources, how do we share them in a fair way?



The Operating System

Applications

The Operating System

Applications

a clean interface

The Operating System

Applications

a clean interface

The Operating System

Hardware

Applications

a clean interface

The Operating System

a complete mess

Hardware

*Hardware : CPU, RAM, HD, SSD, NIC, USB....*

*Hardware : CPU, RAM, HD, SSD, NIC, USB....*

x86\_64 Instruction Set Architecture

*Hardware : CPU, RAM, HD, SSD, NIC, USB....*

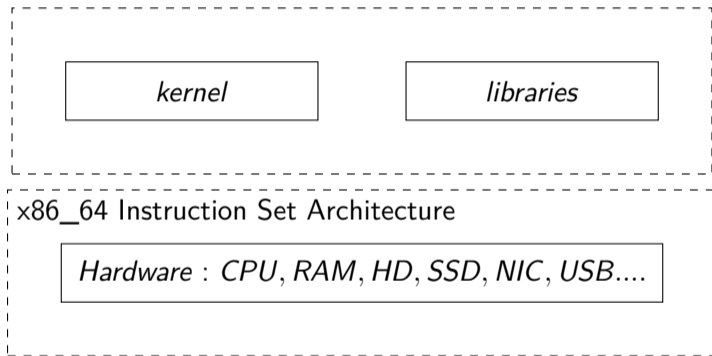


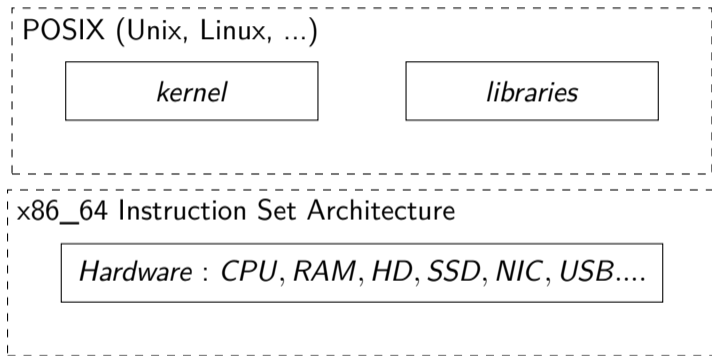
*kernel*

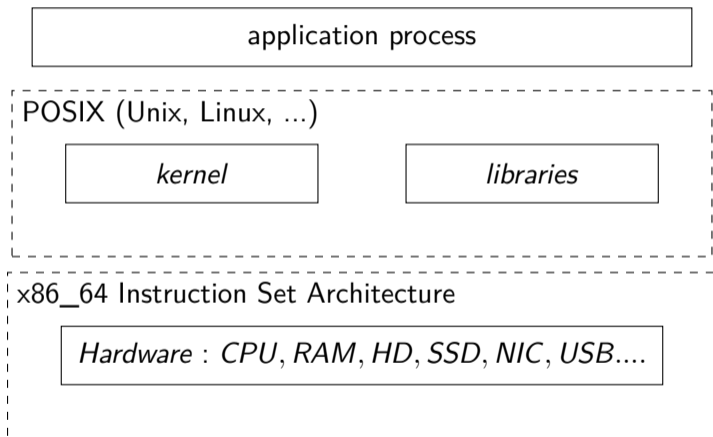
*libraries*

x86\_64 Instruction Set Architecture

*Hardware : CPU, RAM, HD, SSD, NIC, USB....*







Operating system API

## Operating system API

- process handling: fork, exec, wait, ...

## Operating system API

- process handling: fork, exec, wait, ...
- process communication: pipes, ..

## Operating system API

- process handling: fork, exec, wait, ...
- process communication: pipes, ..
- threads handling: pthread\_create, ...



## Operating system API

- process handling: fork, exec, wait, ...
- process communication: pipes, ..
- threads handling: pthread\_create, ...
- managing directory and file ownership

## Operating system API

- process handling: fork, exec, wait, ...
- process communication: pipes, ..
- threads handling: pthread\_create, ...
- managing directory and file ownership
- network handling: socket, listen, accept, ...

## Operating system API

- process handling: fork, exec, wait, ...
- process communication: pipes, ..
- threads handling: pthread\_create, ...
- managing directory and file ownership
- network handling: socket, listen, accept, ...
- ...

## Operating system API

- process handling: fork, exec, wait, ...
- process communication: pipes, ..
- threads handling: pthread\_create, ...
- managing directory and file ownership
- network handling: socket, listen, accept, ...
- ...

## The C Standard Library (ISO C18)

## Operating system API

- process handling: fork, exec, wait, ...
- process communication: pipes, ..
- threads handling: pthread\_create, ...
- managing directory and file ownership
- network handling: socket, listen, accept, ...
- ...

## The C Standard Library (ISO C18)

- memory allocation: malloc, free, ...

## Operating system API

- process handling: fork, exec, wait, ...
- process communication: pipes, ..
- threads handling: pthread\_create, ...
- managing directory and file ownership
- network handling: socket, listen, accept, ...
- ...

## The C Standard Library (ISO C18)

- memory allocation: malloc, free, ...
- signal handling: signal, raise, kill, ..

## Operating system API

- process handling: fork, exec, wait, ...
- process communication: pipes, ..
- threads handling: pthread\_create, ...
- managing directory and file ownership
- network handling: socket, listen, accept, ...
- ...

## The C Standard Library (ISO C18)

- memory allocation: malloc, free, ...
- signal handling: signal, raise, kill, ..
- file operations: fopen, fclose, fread, fwrite, ....

## Operating system API

- process handling: fork, exec, wait, ...
- process communication: pipes, ..
- threads handling: pthread\_create, ...
- managing directory and file ownership
- network handling: socket, listen, accept, ...
- ...

## The C Standard Library (ISO C18)

- memory allocation: malloc, free, ...
- signal handling: signal, raise, kill, ..
- file operations: fopen, fclose, fread, fwrite, ....
- ...



## Operating system API

- process handling: fork, exec, wait, ...
- process communication: pipes, ..
- threads handling: pthread\_create, ...
- managing directory and file ownership
- network handling: socket, listen, accept, ...
- ...

## The C Standard Library (ISO C18)

- memory allocation: malloc, free, ...
- signal handling: signal, raise, kill, ..
- file operations: fopen, fclose, fread, fwrite, ....
- ...

## Command Line Interpreter

## Operating system API

- process handling: fork, exec, wait, ...
- process communication: pipes, ..
- threads handling: pthread\_create, ...
- managing directory and file ownership
- network handling: socket, listen, accept, ...
- ...

## The C Standard Library (ISO C18)

- memory allocation: malloc, free, ...
- signal handling: signal, raise, kill, ..
- file operations: fopen, fclose, fread, fwrite, ....
- ...

## Command Line Interpreter

- shell: the text based interface

## Operating system API

- process handling: fork, exec, wait, ...
- process communication: pipes, ..
- threads handling: pthread\_create, ...
- managing directory and file ownership
- network handling: socket, listen, accept, ...
- ...

## The C Standard Library (ISO C18)

- memory allocation: malloc, free, ...
- signal handling: signal, raise, kill, ..
- file operations: fopen, fclose, fread, fwrite, ....
- ...

## Command Line Interpreter

- shell: the text based interface
- scripting languages

## Operating system API

- process handling: fork, exec, wait, ...
- process communication: pipes, ..
- threads handling: pthread\_create, ...
- managing directory and file ownership
- network handling: socket, listen, accept, ...
- ...

## The C Standard Library (ISO C18)

- memory allocation: malloc, free, ...
- signal handling: signal, raise, kill, ..
- file operations: fopen, fclose, fread, fwrite, ....
- ...

## Command Line Interpreter

- shell: the text based interface
- scripting languages
- ...

```
int counter = 0;

void hello(char *name){
    printf("Hello: %s, %d\n", name, counter);
}

int main() {
    char *me = argv[1];
    while(counter != 10) {
        counter++;
        hello(me);
        sleep(1);
    }
    return 0;
}
```

Operating System

*Hardware : CPU, 8GB RAM, ....*

A: 2 GB RAM

Operating System

*Hardware : CPU, 8GB RAM, ....*

A: 2 GB RAM

B: 2 GB RAM

Operating System

*Hardware : CPU, 8GB RAM, ....*



A: 4 GB RAM

B: 4 GB RAM

Operating System

*Hardware : CPU, 8GB RAM, ....*

A: 4 GB RAM

B: 4 GB RAM

C: 32 GB RAM

Operating System

*Hardware : CPU, 8GB RAM, ....*

Hypervisor

*Hardware* : *CPU, 8GB RAM, ....*

OS: Linux

Hypervisor

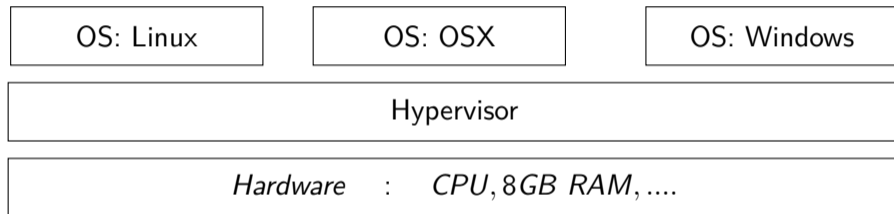
*Hardware : CPU, 8GB RAM, ....*

OS: Linux

OS: OSX

Hypervisor

*Hardware : CPU, 8GB RAM, ....*





- Time: scheduling, how do we divide the execution time among processes



- Time: scheduling, how do we divide the execution time among processes
- Memory: efficient allocation and deallocation, malloc/free...

- Time: scheduling, how do we divide the execution time among processes
- Memory: efficient allocation and deallocation, malloc/free...

to implement an operating system

Why is it hard to implement an operating system?



Start programming today.