

# System 101

Maël Auzias

ENSIBS - UBS

March 2016



Figure: [sys-101.auzias.net](http://sys-101.auzias.net)

# Course details

## Objectives

- ▶ How do *computers* work?
- ▶ What are they made of?
- ▶ What is an OS?



## Course details



### Evaluation

- ▶ Short test at the end of each practice
- ▶ Final exam (1 hour)
- ▶ All equal weighting

### Material

- ▶ Slides available at [sys-101.auzias.net](http://sys-101.auzias.net) (github too)
- ▶ **To read:** Modern Operating System - Andrew Tanenbaum. ISBN-13: 978-0133591620

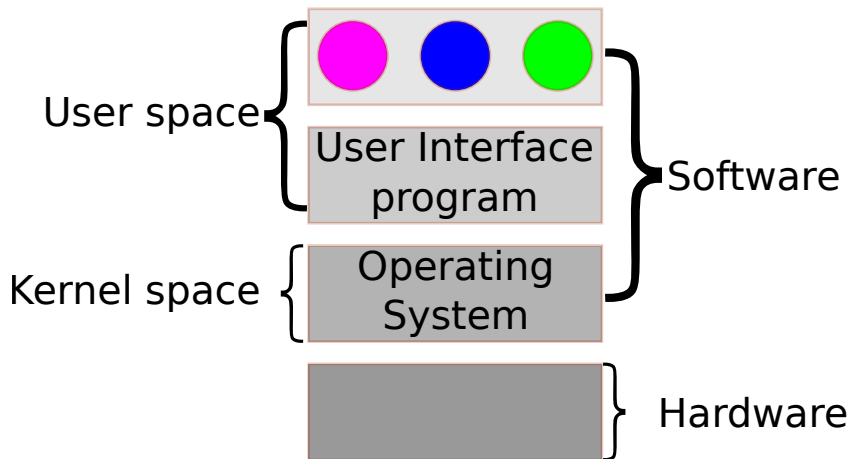
# Presentation Outline

What is an OS?

OS Concepts

Processes and Thread

## Operating System



# Operating System

## Two basic unrelated functions

- ▶ provide application programmers a clean abstract set of resources,
- ▶ manage hardware resources.

## Customers

OS real customers are **programs developpers**, not end users of theses developped programs.

# The OS as an API provider

## Abstraction challenge

- ▶ Hardware design, as well made as it can be, only offer awkward and ugly interface to communicate.
- ▶ Instruction set, memory organization, I/O, bus structure are not user friendly.
- ▶ Programmers need not to worry about all of that thanks to the **abstraction level** provided by OS.

# The OS as the resource manager

## Resource challenge

- ▶ Orderly allocation of the processors, memories, I/O devices for all the programs competing for them.
- ▶ Software resources (files, DB, network access) are also managed by the OS.
- ▶ Multiplexing:
  - ▶ Time multiplexing (CPU, printer),
  - ▶ Space multiplexing (RAM, disk).



## The OS history

1945-55 First generation: vacuum tubes

1955-65 Second generation: Transistors and Batch Systems

1965-80 Third generation: ICs<sup>1</sup> and Multiprogramming<sup>2</sup>

1980-now Fourth generation: Personal Computers

future Fifth generation: any suggestion?

---

<sup>1</sup>Integrated Circuit

<sup>2</sup>several programs running at once

## The OS Zoo

**Mainframe** Thousands of disks and millions gigabytes of data (high end web servers, servers for business-to-business transactions). These OS are focused on executing many jobs at once.

**Server** Multiple users served at once through a network, they provide print/file/web services.

**Multiprocessor** Multiples CPUs are hosted into one system (also called, according to what and how they share it: parallel computers, multicomputers, or multiprocessors).

**Personal Computers** Usually used for game, spreadsheet, word processing and web browsing (laptop, desktop).

## The OS Zoo

**Handheld** Small computers offering telephony, address book, web apps. They are becoming more and more sophisticated and blurring the difference between personal computers and handheld computers.

**Embedded** Microwave ovens, (non-smart) TV and swatches, (not connected) cars, Bluray readers... They usual do not allow user-installed softwares.

**Sensor Node** Usually small and simple to run on constraint devices with little RAM/ROM and battery life (TinyOS).

**RTS<sup>3</sup>** Industrial process control, avionics, military...

**Smart Card** Credit card.

---

<sup>3</sup>Real-Time System

# Presentation Outline

What is an OS?

OS Concepts

Processes and Thread

# OS Concepts

## Overview

- ▶ Processes,
- ▶ Address spaces,
- ▶ Files,
- ▶ Input/Output,
- ▶ Permissions.

## Processes

A process is a program being executed. Each process:

- ▶ has an address space (**core image**),
- ▶ has a register (program counter and stack pointer),
- ▶ has a list of open files,
- ▶ has a list of related processes
- ▶ and all the details needed to run a program.

# Process

## OS management of Processes

- ▶ Execute,
- ▶ Save execution state (file pointers list, number of bytes to be read next) in a **process table**<sup>4</sup>,
- ▶ Stop.

A process corresponds to its **core image** and its **process table entry**.

---

<sup>4</sup>An array or linked list

## Child Process

### Process life

- ▶ A **system call** starts a process.
- ▶ Binary code is executed.
  - ▶ The process can create other processes, called **child processes** (and so on – tree).
  - ▶ Processes can communicate together using **IPC**<sup>5</sup>.
- ▶ The OS may send **alarm** signal (interruption) to the process.
- ▶ The process executes a **system call** to terminates itself.

---

<sup>5</sup>Inter Process Communication



## Users Process

### Process life

- ▶ **UID**<sup>6</sup> is a unique number assigned to each system user.
- ▶ Every process started has the UID of the user who started it.
- ▶ Every child process has the UID of its parent.
- ▶ One UID is called the super-user. The super-user has all permission.
- ▶ Users may also be members of groups. Each group has a **GID**.

---

<sup>6</sup>User Identification

# Address Space

An address space is a memory location (from 0 to some maximum) and contains:

- ▶ executable program,
- ▶ program's data,
- ▶ its stack.

## Address Space

- ▶ Physical memory,
- ▶ Virtual memory.

# File

OS are expected to hide all peculiar disk operations to offer abstracted model of device-independent file management.

- ▶ System calls are required to:
  - ▶ Create, remove, read and write files; create and remove directories.
- ▶ File system also match a tree structure.

## File

- ▶ Path
  - ▶ Absolute: from the root directory, starting with /
  - ▶ Relative: from the current directory<sup>7</sup>, starting with "." or a directory name.
- ▶ Several files may have the same name.
- ▶ Each file has a unique absolute path (and an infinity of relative ones).
- ▶ Mounted file system, merging trees.
- ▶ Special files (so I/O devices are abstracted to be used through same system calls as files do):
  - ▶ Block files,
  - ▶ Character files.
  - ▶ Special files are kept in /dev.

---

<sup>7</sup>each process has a current working directory. A system call allows process to change their working directory

## Special Files

I/O devices are abstracted to be used through same system calls as files do.

- ▶ Devices:
  - ▶ Block files,
  - ▶ Character files.
  - ▶ Special files are kept in `/dev`.
- ▶ Pipe:
  - ▶ IPC mean,
  - ▶ a special system call needs to be performed to know it's not a real file.

# Tree

- ▶ Both process and file are structured as tree.
- ▶ Process tree are usually not very deep, unlike file trees.
- ▶ Process hierarchy are usually short-lived (minutes or less) while directories may exist for years.
- ▶ Ownership and protection differs too.

## Permissions

**u g o : user group other**

- ▶ Three 3-bit fields
  - ▶ read
  - ▶ write
  - ▶ execute
- ▶ **rwX rwx rwx** do-what-ever-you-want-file
- ▶ **rwX rwx r-x** web-file
- ▶ **rw- rwx rwx** virus
- ▶ **r-x — —** personal-backup.tgz

| Right | File        | Directory            |
|-------|-------------|----------------------|
| r     | can read    | can list files       |
| w     | can write   | can add/delete files |
| x     | can execute | can go through       |

Figure: Permissions meaning



# Presentation Outline

What is an OS?

OS Concepts

Processes and Thread

# Processes and Thread

## Overview

- ▶ Why processes are so important?
- ▶ What differences between processes and threads?
- ▶ ==

# Processes

- ▶ Most important abstraction,
- ▶ Turn single-CPU into multiple virtual CPU
- ▶ Enable pseudo concurrent operations (pseudoparallelism),
- ▶ Without, modern computing could not exist.

# Processes

## Processes

- ▶ are instance of executing program
- ▶ include current values of
  - ▶ program counter,
  - ▶ registers,
  - ▶ variables.
- ▶ have their own virtual CPU (multiprogramming) – considerations about time management, RTS.

## The student partying (a fictionnal analogy)

- ▶ The student, at a home party, makes a cocktail.
  - ▶ Student: CPU, recipe: program, drinks: data, glasses: resource, action: process.
- ▶ While pouring the last ingredient her/his phone rang and s/he answers.
  - ▶ Student: CPU, phone-skill: program, phone: resource, phone call details: data, action: process.

A process is an activity having a program, input, output and a state. OS uses scheduling algorithm to determines when to stop/start which process.

# Processes

- ▶ Most important abstraction,
- ▶ Turn single-CPU into multiple virtual CPU,
- ▶ Enable pseudo concurrent operations (pseudoparallelism),
- ▶ Without, modern computing could not exist.

# Processes

## Creation

- ▶ System initialization,
- ▶ Process creation done by a running process,
- ▶ User request,
- ▶ Initiation of a batch jobs.

## Termination

- |                |                              |
|----------------|------------------------------|
| ▶ Voluntary    | ▶ Involuntary                |
| ▶ Normal exit, | ▶ Fatal error,               |
| ▶ Error exit.  | ▶ Killed by another process. |

## Process states

### State

- a. Running,
- b. Ready,
- c. Blocked.

### Transition

1. Scheduler pick another process.
2. Scheduler pick this process.
3. Input available.
4. Input required.



# Threads

## Threads

1. Processes within a process.
2. Enable to decompose big task into multiple sequential smaller tasks ..
3. .. while **sharing** a memory space.
4. Easier, faster, to create and destroy than processes as they are lighter.

Web browser example with multiple threads.

I hope you liked it and learnt something new !



Figure: [sys-101.auzias.net](http://sys-101.auzias.net)