Unit 2

OS and Tools

Alberto Casagrande
Email: acasagrande@units.it

A.Y. 2021/2022

As Programmers . . .

we are interested in:

- reading data from an input device
- implementing functions to operate on data sets
- providing results to an output device

As Programmers . . .

we are NOT interested in:

- how the I/O devices work
- where the data are physically stored in memory
- how our programs will be executed
- . . .

As Programmers . . .

we are NOT interested in:

- how the I/O devices work
- where the data are physically stored in memory
- how our programs will be executed
- . . .

We need an abtraction layer between HW and programs

As Programmers . . .

we are NOT interested in:

- how the I/O devices work
- where the data are physically stored in memory
- how our programs will be executed
- . . .

We need an abtraction layer between HW and programs

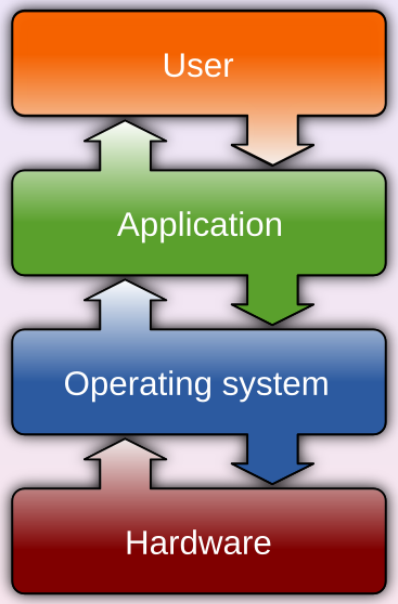
We need an Operating System

Operating Systems

Software that manage resources

- memory
- disks
- CPUs
- ...

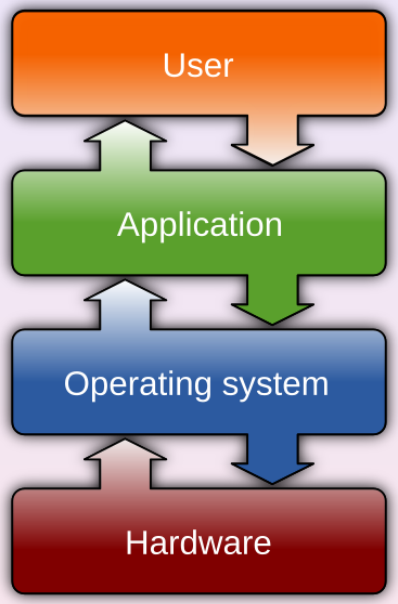
Provide interfaces for programs (Application Programming Interface aka API) and users



Operating Systems

Software that manage resources

- memory
- disks
- CPUs
- ...



Provide interfaces for programs (Application Programming Interface aka API) and users

There exist hundreds of operating systems e.g., Windows, macOS, BeOS, GNU/Linux, iOS, Android, ReactOS

POSIX standard

Is a IEEE standard about:

- Process (i.e., programs in execution) creation and control
- File and directory operations
- C library
- I/O port interface and control
- Command interpreter
- Standard utility and command
- ...

POSIX standard

Is a IEEE standard about:

- Process (i.e., programs in execution) creation and control
- File and directory operations
- C library
- I/O port interface and control
- Command interpreter
- Standard utility and command
- ...

macOS is POSIX-certified, Windows is not POSIX compliant.

GNU/Linux

Is a free (as in speach) OS.

- Linux is the core
- GNU is a project that provides tools and command, e.g., GNU C Compiler - GCC

GNU/Linux

Is a free (as in speach) OS.

- Linux is the core
- GNU is a project that provides tools and command, e.g., GNU C Compiler - GCC

There exist many different distributions. They customize:

- installer
- software manager
- additional software

GNU/Linux

Is a free (as in speach) OS.

- Linux is the core
- GNU is a project that provides tools and command, e.g., GNU C Compiler - GCC

There exist many different distributions. They customize:

- installer
- software manager
- additional software

POSIX compliant, but not certificate (freedom has a price)

GNU/Linux

You will extensively use GNU/Linux during DSSC program (but macOS is fine)

GNU/Linux

You will extensively use GNU/Linux during DSSC program (but macOS is fine)

Let's see how to install it and use it

We will focus on Ubuntu distribution (not the "best", but user-friendly)

How to install Ubuntu

We need:

1. a PC or an Intel Mac with a USB port
2. 16GB of free disk space
3. a USB key (at least 2GB)
4. a network connection

How to install Ubuntu

We need:

1. a PC or an Intel Mac with a USB port
2. 16GB of free disk space
3. a USB key (at least 2GB)
4. a network connection

or

1. a PC running a virtualization environment, e.g., VirtualBox
2. 16GB of free disk space
3. a network connection

How to install Ubuntu

If you have a M1 Mac you can only install it as a Virtual Machine

1. a virtualization environment, e.g., UTM
2. 16GB of free disk space
3. a network connection

How to install Ubuntu

If you opt for a "real" installation, you need to:

- download Ubuntu
- download and install Etcher
- prepare a bootable USB Live key by using Etcher
- reboot your PC and select the USB key ad boot device
- follow the instructions (pay attention and do not delete your OS!!!)

How to install Ubuntu

If you opt for a "virtual" installation, you need to:

- download Ubuntu
- download and install your preferred virtual environment
- create your VM
- attach the Ubuntu ISO to your VM and boot from it
- follow the instructions (don't worry about messing up with the VM's disk)

How to install Ubuntu

Demo session

Users in Modern OS

Modern OS are multi-users, i.e., they support many users on the same system

Every user has a reserved disk space (dubbed home in POSIX) where they can store personal data and program

So, after the boot, the system asks for a username and a password

Say “Hello” to Command Line

Ubuntu has a fully functional Graphical User Interface (GUI)

We will use the Command-Line user Interface (a.k.a. shell) because it’s:

Say “Hello” to Command Line

Ubuntu has a fully functional Graphical User Interface (GUI)

We will use the Command-Line user Interface (a.k.a. shell) because it’s:

- easier to use (for experts)
- powerful
- programmable
- cool

Say "Hello" to Command Line

Ubuntu has a fully functional Graphical User Interface (GUI)

We will use the Command-Line user Interface (a.k.a. shell) because it's:

- easier to use (for experts)
- powerful
- programmable
- cool

The default shell in Ubuntu is BASH.

Few info about secondary memory

Data are maintained in disks by an OS component called file system

Many kinds of FS e.g., VFAT, Ext4 (GNU/Linux "default"), APFS (macOS), NTFS (Windows)

Data are organized in a *tree* of directories (branches of the tree).

File Systems in POSIX

- the symbol / to distinguish branch levels, e.g., `/home/al`
- / is the root of the tree
- /home contains the users' homes
- ./ is the current directory
- ../ is the parent level
- ~ denotes the current user's home

File Systems in POSIX

- the symbol / to distinguish branch levels, e.g., `/home/al`
- / is the root of the tree
- /home contains the users' homes
- ./ is the current directory
- ../ is the parent level
- ~ denotes the current user's home

Directory names can be composed to specify a *path*

- absolute paths start from the root e.g., `/home/al/Desktop/` or `/user`
- relative paths start from the current active/directory, e.g., `Desktop`, `./Download/`, or `Download/../Desktop`

Demo session

Some simple BASH commands

- ls lists the content of a directory

```
foo@bar:~/$ ls
Desktop     Download   Pictures   Templates
Documents   Music      Public     Videos
```

- pwd prints the name of the current/working directory

```
foo@bar:~/$ pwd
/home/foo
```

Some simple BASH commands (Cont'd)

- mkdir create new directories

```
foo@bar:~/$ mkdir test
foo@bar:~/$ ls
Desktop     Music     Templates
Documents   Pictures  test
Downloads   Public    Videos
```

- cd change directory. Without parameter means "go to home"

```
foo@bar:~/$ cd test
foo@bar:~/test$ cd ../../../usr
foo@bar:/usr$ cd
foo@bar:~/$ cd foo
bash: cd: foo: No such file or directory
```

Some simple BASH commands (Cont'd 2)

- rm delete files/directories

```
foo@bar:~/$ rm -r test
```

- man print command manual pages

```
foo@bar:~/$ man ls
```

- apropos search words in manual pages

```
foo@bar:~/$ apropos compress
```

Some simple BASH commands (Cont'd 3)

- grep print lines matching a pattern

```
foo@bar:~/$ grep .bashrc
.bashrc
foo@bar:~/$ grep 'la' .bashrc
alias la='ls -A'
```

- cat print a file on the stdout
- less print a file on terminal one screenful at a time
- head output the first part of files

```
foo@bar:~/$ head -n 3 .bashrc
# ~/.bashrc: executed by bash(1) for non-login s
# see /usr/share/doc/bash/examples/startup-files
# for examples
```

Pipelining

We can use | (pipe) to use the output of a command as the input of another

```
foo@bar:~/$ echo 'Hi, Foo!'
Hello, man!
foo@bar:~/$ echo 'Hi, Foo!' | sed s/Hi/Hello/
Hello, Foo!
```

Output Redirection

```
foo@bar:~/$ echo 'Hi, Foo!' > msg.txt
foo@bar:~/$ cat msg.txt
Hi, Foo!
foo@bar:~/$ echo 'This is cool' > msg.txt
foo@bar:~/$ cat msg.txt
This is cool
```

```
foo@bar:~/$ echo 'This is better' >> msg.txt
foo@bar:~/$ cat msg.txt
This is cool
This is better
```

File Descriptors

Are positive numbers that “name” files

POSIX systems handle everything as files

stdin, stdout, and stderr have FD 0, 1, and 2.

File Descriptors (Cont'd)

We can stream data to either stdout and stderr

```
foo@bar:~/$ echo 'WHAT?!?!' >&1
WHAT?!?!
foo@bar:~/$ cat msg.txt
This is cool
This is better
foo@bar:~/$ cat msg.txt | grep cool
This is cool
foo@bar:~/$ cat msg.txt >&2 | grep cool
This is cool
This is better
```

File Descriptors (Cont'd 2)

We can also data from either stdin, stdout and stderr

```
foo@bar:~/$ cat test.txt
cat: test.txt: No such file or directory
foo@bar:~/$ (echo "N" >&1; echo "Y" >&2)
N
Y
foo@bar:~/$ (echo "N">&1;echo "Y">&2)1>test.txt
Y
foo@bar:~/$ cat test.txt
N
foo@bar:~/$ (echo "N!";echo "Y">&2)1>test.txt
Y
foo@bar:~/$ cat test.txt
N!
```

(Extended) Regular Expressions

Are patterns to describe strings.

E.g., `[az]T.` describes strings beginning with either `aT` or `zT` and having 3 characters

- `.` any single character
- `-` denotes a range e.g., `a-z`
- `?` the prev item at most once e.g., `a?`
- `+` the prev item at least once
- `*` the prev item occurs from 0 to many times
- `()` bound a sub-RE
- `|` match both RE e.g., `(it)|(comm)`
- `[ ]` choose one in the set e.g., `[a-z]`

(Extended) Regular Expressions (Cont'd)

^ and $ denote begin and end of a line, respectively.

```
foo@bar:~/$ grep h.*b.* msg.txt
This is better
foo@bar:~/$ grep "\(b.*\)\|\(cool\)" msg.txt
This is cool
This is better
```

The escape character \ is needed because grep uses basic regular expression by default.

Programmability

Shells can be programmed to perfom complex tasks

```
foo@bar:~/$ for i in $(seq 1 3); do
> echo test_${i};
> done > test.txt
foo@bar:~/$ cat test.txt
test_1
test_2
test_2
```

If we have enough time, we will focus on it next week.

A user to rule them all . . .

Not all users must have the same privileges

E.g.,

- The system owner should be able to do everything
- A “host user” should not mess-up other users’ data

A user to rule them all . . .

Not all users must have the same privileges

E.g.,

- The system owner should be able to do everything
- A "host user" should not mess-up other users' data

Modern OS provide a "superuser" to rule all the system files/programs.

root in POSIX systems, administrator in Windows.

Impersonate Superuser

Ubuntu and macOS implement a mechanism to impersonate superuser.

sudo (Super User DO) lets authorized users to impersonate superuser.

```
foo@bar:~/$ mkdir /test
mkdir: cannot create directory '/test': Permission
denied
foo@bar:~/$ sudo mkdir /test
[sudo] password for al:
foo@bar:~/$
```

Conda

Conda is package and environment management system for Windows, macOS, and GNU/Linux.

It is useful to search, install, and update software at user level.

```
(base) foo@bar:~/$ conda search jupyter
# Name    Version                 Build        Channel
jupyter     1.0.0    py37hd43f75c_7      pkgs/main
jupyter     1.0.0    py38hd43f75c_7      pkgs/main
jupyter     1.0.0    py39hd43f75c_7      pkgs/main

(base) foo@bar:~/$ conda install jupyter
...
```

Conda Environment

Moreover, conda also provides environments

They contain all the software needed for a specific project

If the clang compiler is required for the project JoJo ...

```
(base) foo@bar:~/$ conda create --name JoJo
...

(base) foo@bar:~/$ conda activate JoJo

(JoJo) foo@bar:~/$ conda install clang
...

(JoJo) foo@bar:~/$ clang
clang-10: error: no input files
```

Conda Environment (Cont'd)

At the end of the working day ...

```
(JoJo) foo@bar:~/$ conda deactivate

(base) foo@bar:~/$ clang
Command 'clang' not found.
```

But

```
(base) foo@bar:~/$ conda activate JoJo

(JoJo) foo@bar:~/$ clang list
# packages in environment at /home/foo/miniconda3/envs/J
# Name      Version      Build  Channel
...
clang          10.0.1    default_h6b8c85e_2
...
```

Managing Conda Environments

To list all the user's environments

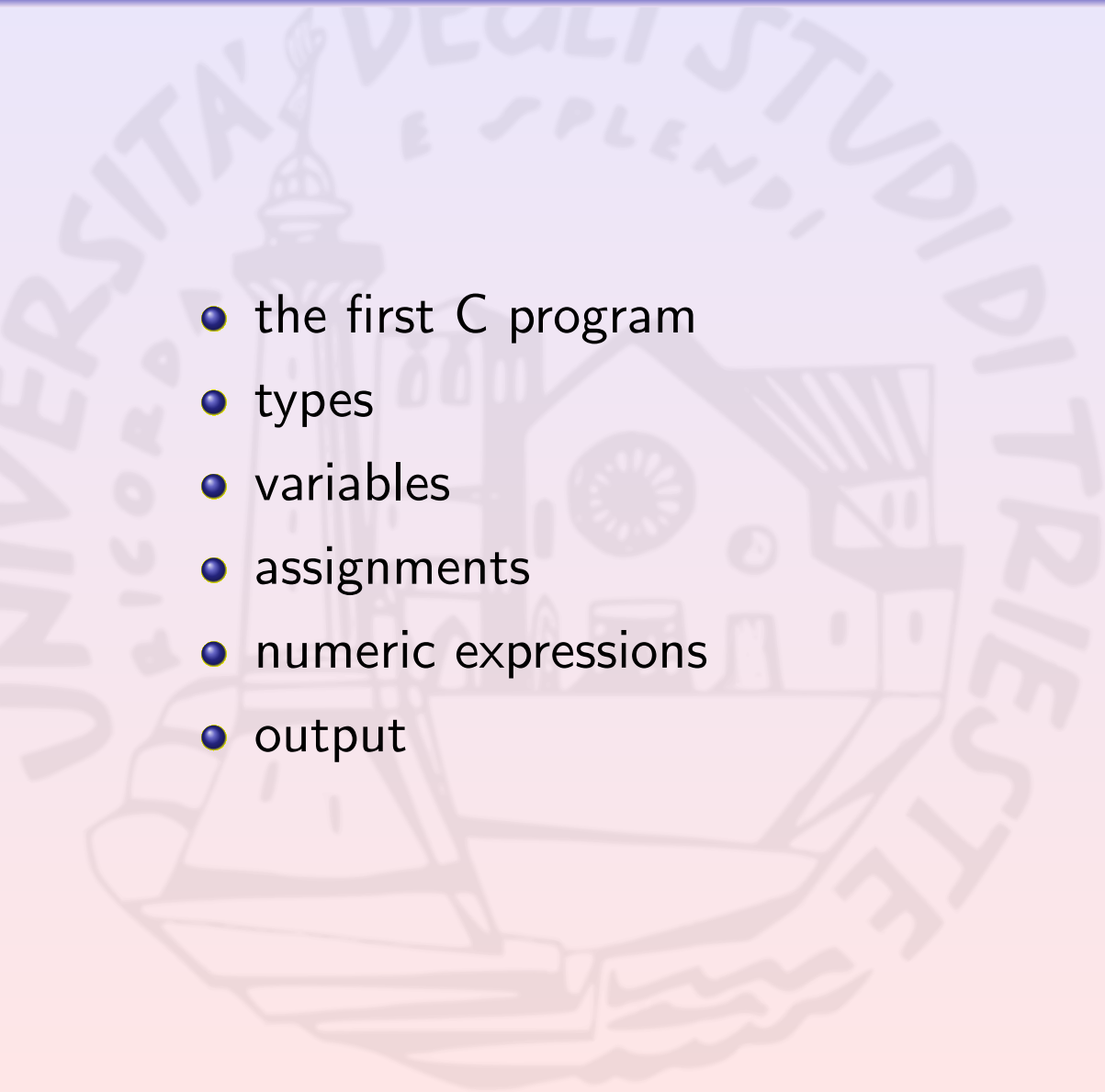
```
(base) foo@bar:~/$ conda env list
# conda environments:
#
base                  *  /home/foo/miniconda3
JoJo                     /home/foo/miniconda3/envs/JoJo
```

To remove an environment

```
(base) foo@bar:~/$ conda env remove --name JoJo

Remove all packages in environment /home/foo/miniconda3/
```

Coming next...



- the first C program
- types
- variables
- assignments
- numeric expressions
- output