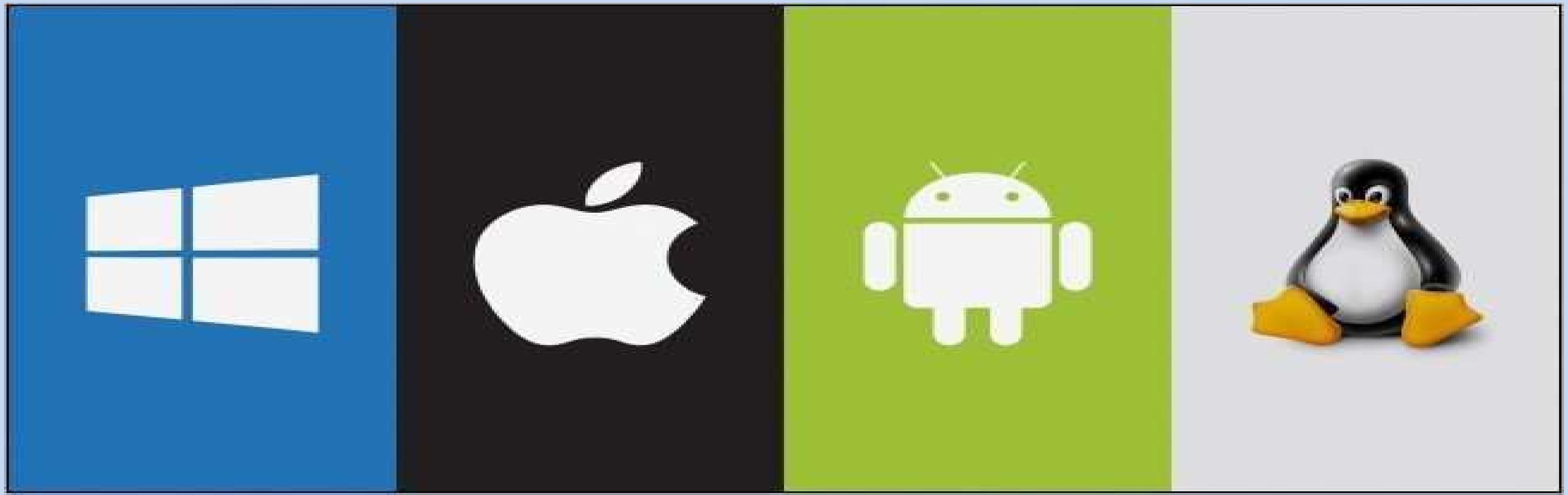
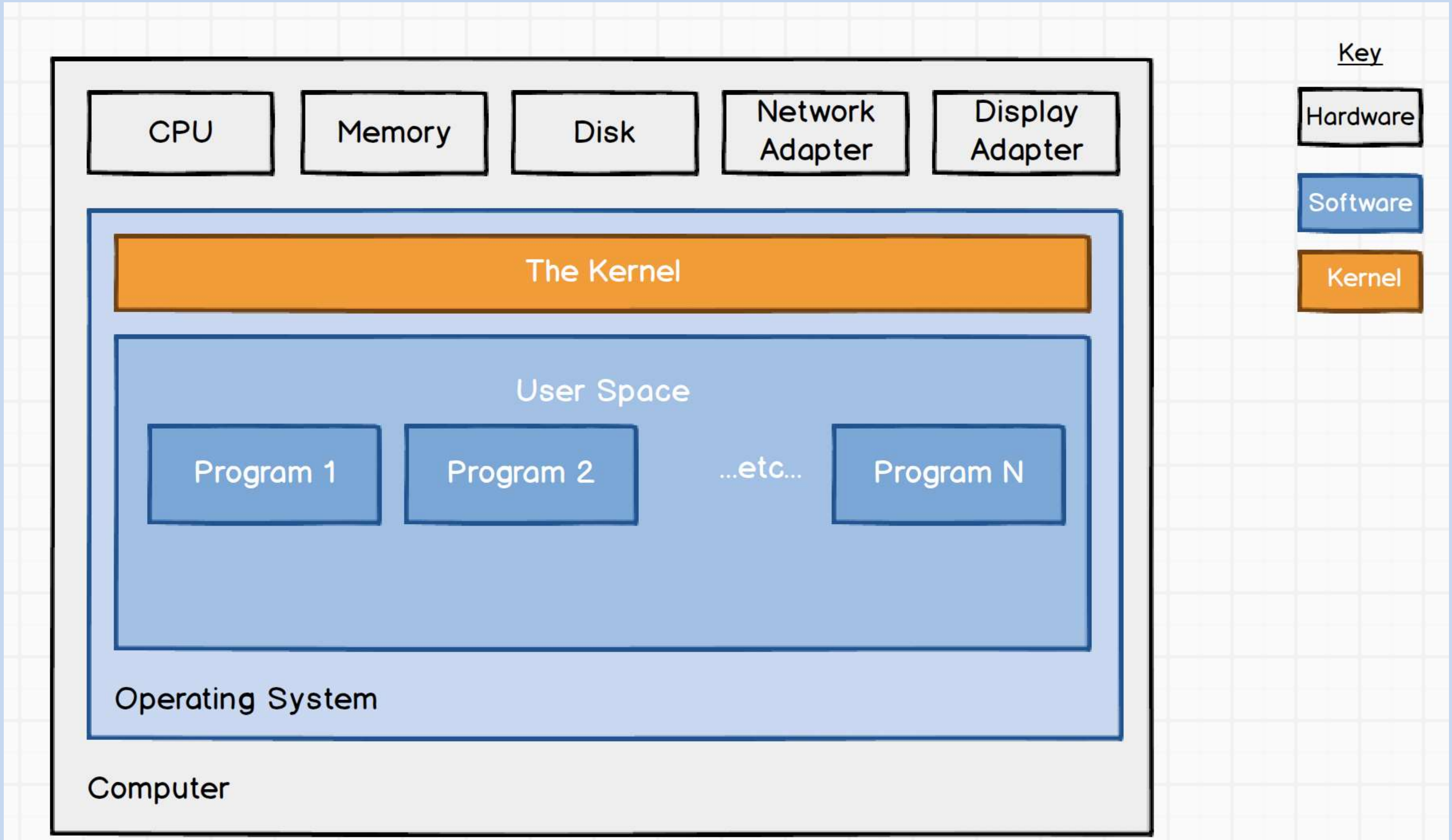


Operating Systems
Sistemas Operativos
Shell

Prof. Amine Berqia

Email :
bamine@ualg.pt

Operating systems

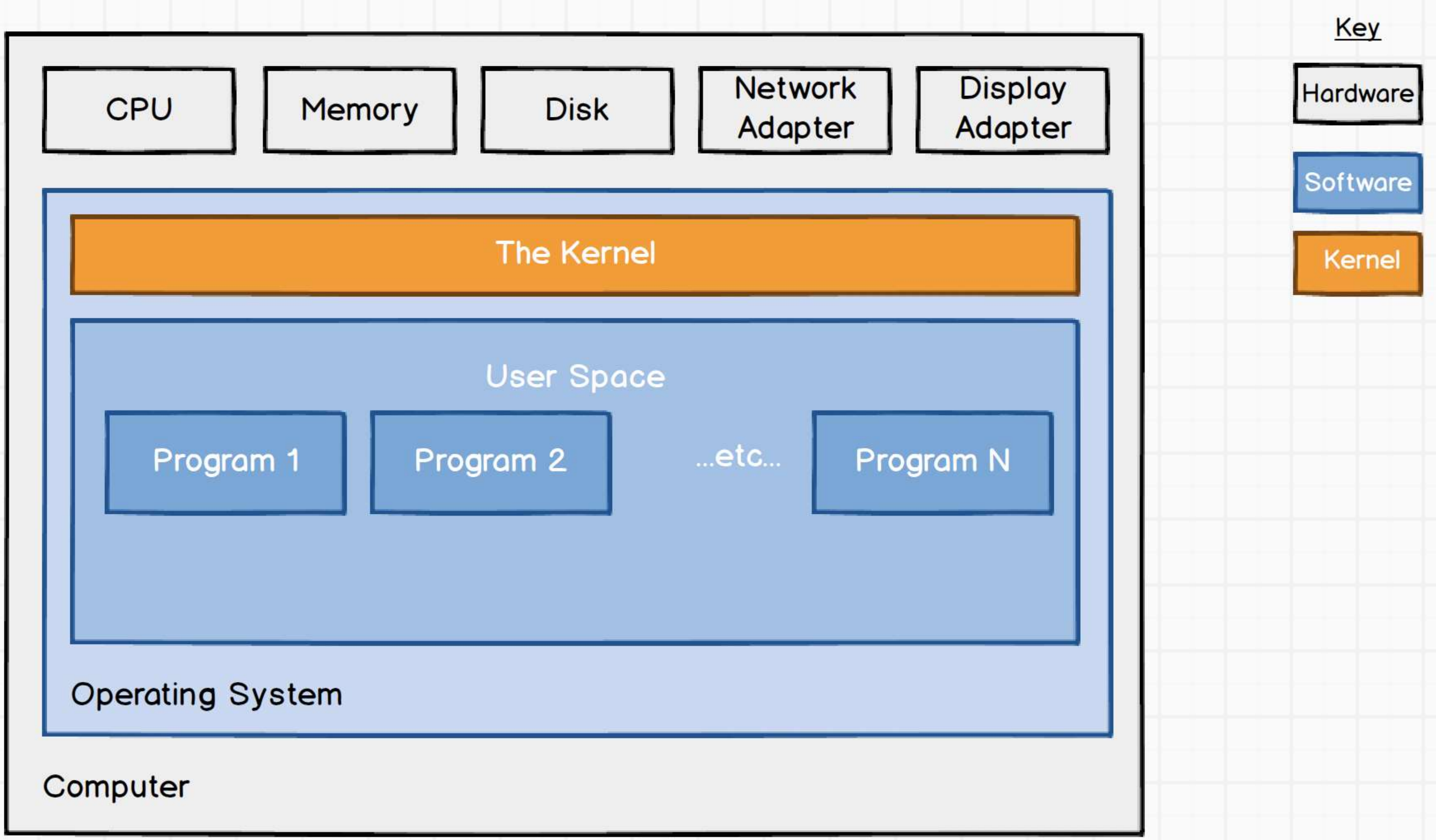


The operating system is generally broken down into two parts - the *kernel* and *user space*

Operating systems
The kernel

This is the part of the operating system that is responsible for the most sensitive tasks: interfacing with physical devices, managing the resources that are available for users and programs, starting up the various systems that are needed, and so on.

Operating systems



The operating system is generally broken down into two parts - the *kernel* and *user space*

Bash as an example of a shell

Variables: There are three types of shell variables

- ✓key parameters
- ✓positional parameters
- ✓special parameters.

Bash as an example of a shell

Special variables or parameters are given a specific meaning:

$# the number of arguments passed to the program
$ references all positional parameters
$0 the name of the program or script to be executed
$$ the number of the process to be executed
$! the number of the last process running in the background
$? the status of the last command not executed in the background
$1 contains the first parameter
$2 contém o segundo parâmetro;
....

Bash as an example of a shell

Make a script in BASH to read a name and write Hello name.

```
#!/bin/bash
read -p ' Your name : ' name
echo "Hello $name"
```

Bash as an example of a shell

mathematical operations

In bash, variables are all strings. Bash itself is not really capable of manipulating numbers.

```
#!/bin/bash
let "a = 5"
let "b = 2"
let "c = a + b"
echo $c
```

Bash as an example of a shell

Operators:

-lt less than

-le less or equal

-eq equal to

-gt greater than

-ne not equal to

. . .

Bash as an example of a shell

To summarize: As in most programming languages, you can create shell variables that temporarily store values in memory. A variable called variable is accessible by writing **$variable**.

The **echo** command displays text or the contents of a variable on the console.

read waits for keyboard input from the user and stores the result in a variable.

You can perform mathematical operations on numbers using the **let** command.

Some variables are accessible everywhere, in all scripts: these are the environment variables. They can be listed with the **env** command.

The parameters sent to our script (like ./script) are passed in numbered variables**: $1, $2, $3** ... The number of parameters sent is indicated in the**$#** variable.

Bash as an example of a shell

control structures: if

```
if [ test ]
then
....
else
....
fi
```

You will notice - and it is very important - that there are spaces inside the [] . We should not write [test] but [test]!

There are three different types of tests that can be performed in bash: tests on strings; tests on numbers; tests on files...

Bash as an example of a shell

A script to check if two names are different or not:

```
#!/bin/bash

if [ $1 != $2 ]

    then
    echo " The 2 names are different!"
    else
    echo "The 2 names are identicals!"
fi
```

```
$ ./names.sh Paul Paula

The 2 names are different!
```

Bash as an example of a shell

A script to check whether a student is admitted to the exam or not:

```
#!/bin/bash

if [ $1  -ge  12 ] then
    echo "A" else
    echo "NA"
fi
```

Bash as an example of a shell

check if the parameter exists:

```
#!/bin/bash

if [ -z $1 ]

then
    echo " Parameter does not exist"
else
    echo "Parâmetro: $1"
fi

            ...
```

Bash as an example of a shell

a script that asks the user to enter the name of a directory and checks if it is a directory:

```
#!/bin/bash

read -p 'Insira um diretório: ' directory
if [ -d $directory ]
then
	echo "OK"
else
	echo "No"
fi
```

-e $file: for file.

-x $file: check if the file is executable

$file1 -nt $file2: check if file1 is newer than file2.

$file1 -ot $file2: ????

……

Bash as an example of a shell

In an if, it is possible to do several tests at the same time

&& **||**

```
#!/bin/bash

if [ $# -ge 1 ] && [ $1 =  'Portugal' ]
.....
```

Bash as an example of a shell

If - elif - else

```
#!/bin/bash

if [ $1 = "Pierre" ]
then
	echo "Salut Pierre !"
elif [ $1 = "Pedro" ]
then
	echo "Ola Pedro"
elif [ $1 = "John" ]
then
	echo "Hi John ?"
else
	echo  "Who r u!"
fi
```

Bash as an example of a shell

case: test multiple conditions at the same time

```
#!/bin/bash

Read opt
case $opt in
1)date +%D;;
2)date +%a;;
3)echo "End"; exit;;
*) echo "options are [1 2 3]";;
esac
```

Unix
Bash as an example of a shell

Loops

For, While, Until

```
for var in val
do
   . . . .
 done

for i in `seq 1 10`;
do
   . . . .
 done
```

```
while [ test ]
do
      echo 'Action'
done

until [ test ]
do
      echo 'Action'
done
```

Bash as an example of a shell

We'll ask the user to say "Yes" and repeat this action until they've done what we wanted. Let's create an exwhile.sh script:

```
#!/bin/bash
echo Say Yes
read response
while [ -z $response ] || [ $response != "Yes" ]
do
echo Say Yes
read response
done
```

Bash as an example of a shell

Make a bash script called exfor1.sh using the for control structure that loops through the /bin and /etc directories for 2 seconds and displays them. Finally, return to the home directory.

```
#!/bin/bash
inicial= "pwd"
for dir in /bin /etc
do
cd $dir
pwd
ls bash*
sleep 2
done
cd $inicial
```

Bash as an example of a shell

Implement a counter in bash called exuntil1.sh using the until control structure with values between 1 and 20.

```
#!/bin/bash
inf=1
sup=20
corrente=$inf

until [ $corrente -gt $sup ]; do
  echo "Valor corrente: " $corrente
  let "corrente = corrente + 1"
done
```