Lab 2 Week 2

Systems Architecture

2023-2024

Degree in Telecommunication Technologies Engineering

C Programming

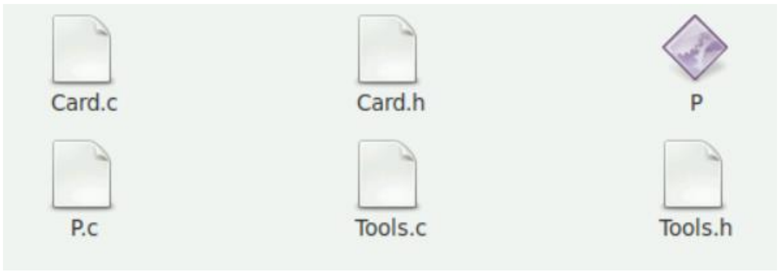
The lab account is in:

https://aulavirtual.lab.it.uc3m.es/#/

In the path `$ cd Documents/AS#2/` you should have several folders for each exercise.

In every folder, you should save the new version of the same main program P.c, the executable P and the modules Tools.c, Tools.h.

In the last exercise, you should have Card.c and Card.h as well.



Part I: Multi Module (Exercises 1, 2, 3)

A project can be composed of several modules. They can be written by us in C or belong to a library such as stdio.lib that the compiler adds on its own.

Exercise 1 – Permissions and cc.sh

Let's create the AS#2 folder and in it the P.c file that will contain a main that invokes the Print_ln function with a string ("Lorem ipsum dolor sit amet.") as argument.

1. Compile.
2. What errors does it give?
3. What is the cause?
4. What is the solution?

As in any operating system, in Linux it is possible to create scripts. Scripts are programs written in the same language in which we write commands in the terminal, in this case the language is called Bash.

In addition to other uses, it will allow us to put in a file the sequence of commands that we want to perform and that because they are a lot or complicated to write we want to group them in a simple command.

Create a file named cc.sh and write in it:

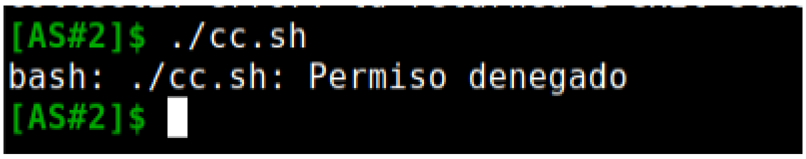
```
#!/bin/bash -v
clear
gcc -Wall -g -o P P.c
```

The **clear** command clears the terminal, so errors and warnings are not mixed with those of the previous compilation. The first line declares the language being used and -v causes the command being executed to be displayed.

Of course you will have to modify the script as your compilation needs change.

We save the file and to run it we go to finish and invoke it.

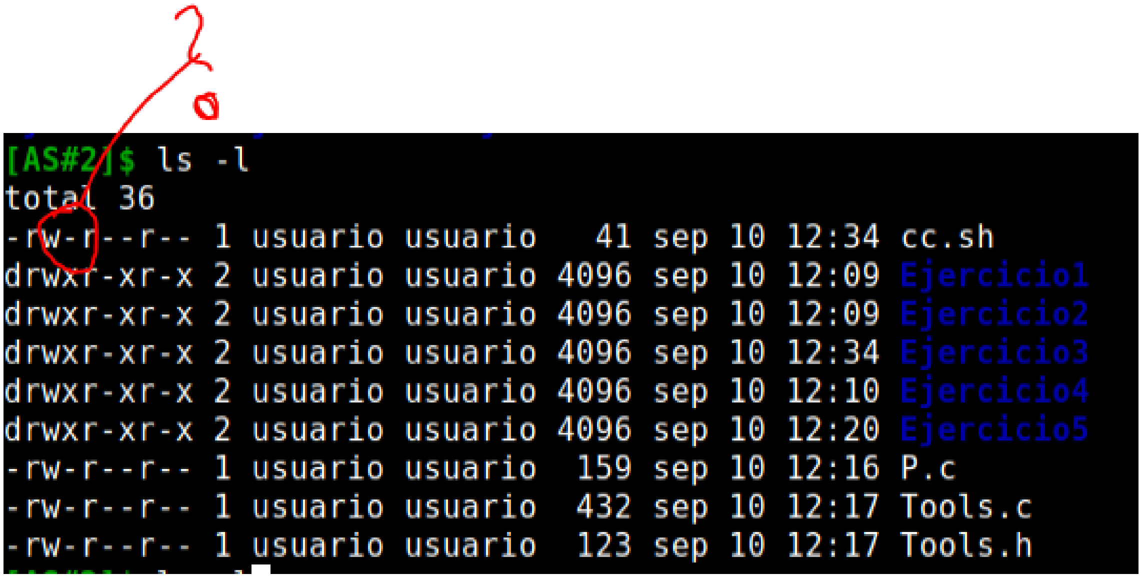
You will probably get this:



```
[AS#2]$ ./cc.sh
bash: ./cc.sh: Permiso denegado
[AS#2]$ 
```

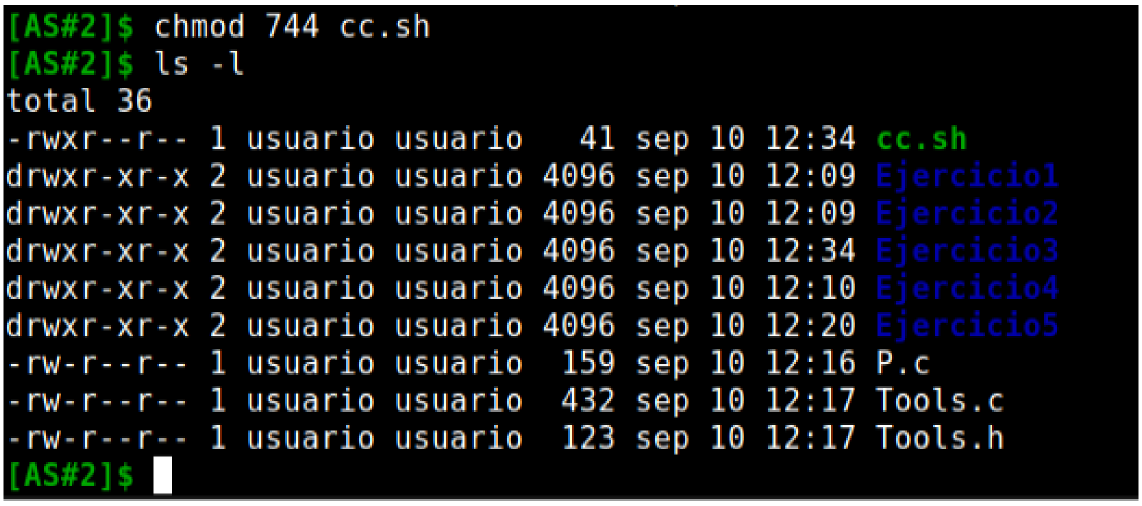
It is because in Linux to be able to execute a program you must have execution permissions on it.

If you run the following you will see that cc.sh does not have the x flag active, neither for .h and .c but that is ok.



```
[AS#2]$ ls -l
total 36
-rw-r--r-- 1 usuario usuario   41 sep 10 12:34 cc.sh
drwxr-xr-x 2 usuario usuario 4096 sep 10 12:09 Ejercicio1
drwxr-xr-x 2 usuario usuario 4096 sep 10 12:09 Ejercicio2
drwxr-xr-x 2 usuario usuario 4096 sep 10 12:34 Ejercicio3
drwxr-xr-x 2 usuario usuario 4096 sep 10 12:10 Ejercicio4
drwxr-xr-x 2 usuario usuario 4096 sep 10 12:20 Ejercicio5
-rw-r--r-- 1 usuario usuario  159 sep 10 12:16 P.c
-rw-r--r-- 1 usuario usuario  432 sep 10 12:17 Tools.c
-rw-r--r-- 1 usuario usuario  123 sep 10 12:17 Tools.h
```

You have two options to activate that flag, one is in the terminal

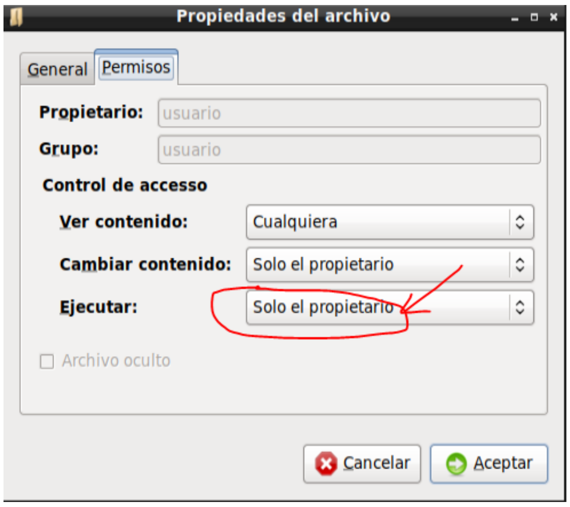


```
[AS#2]$ chmod 744 cc.sh
[AS#2]$ ls -l
total 36
-rwxr--r-- 1 usuario usuario   41 sep 10 12:34 cc.sh
drwxr-xr-x 2 usuario usuario 4096 sep 10 12:09 Ejercicio1
drwxr-xr-x 2 usuario usuario 4096 sep 10 12:09 Ejercicio2
drwxr-xr-x 2 usuario usuario 4096 sep 10 12:34 Ejercicio3
drwxr-xr-x 2 usuario usuario 4096 sep 10 12:10 Ejercicio4
drwxr-xr-x 2 usuario usuario 4096 sep 10 12:20 Ejercicio5
-rw-r--r-- 1 usuario usuario  159 sep 10 12:16 P.c
-rw-r--r-- 1 usuario usuario  432 sep 10 12:17 Tools.c
-rw-r--r-- 1 usuario usuario  123 sep 10 12:17 Tools.h
[AS#2]$
```

Activate the execution permission of cc.sh of the owner. If instead of 744 you put other values (octal) different permissions will be activated. Go get documented about it.

The other way is with the graphical interface:

- In the file manager right-click on the script.
- Select "properties".
- Activate the "permissions" tab.
- And in "execute" select "owner only".
- Accept.



Propiedades del archivo

General | Permisos

Propietario: usuario

Grupo: usuario

Control de accesso

Ver contenido: Cualquiera

Cambiar contenido: Solo el propietario

Ejecutar: Solo el propietario

☐ Archivo oculto

Now we can run the script

The scripts will be needed in the last weeks of the laboratory C Project.

Exercise 2 – Tools.h

When we encounter **[-Wimplicit-function-declaration]** we already know that it is because the function or variable that is indicated is not declared and in C it is mandatory to declare before using.

The immediate solution will be to include the Print_ln declaration before the main, but today we are going to do it in another way.

We are going to create the file Tools.h; this is a header file that is going to contain some of the declarations that we need for our program to work.

As it is a header file we have to protect ourselves against the possibility of including directly or indirectly several times the same file so we will use the common mechanism for this.

The first line of an .h file is always going to be

```
#ifndef <label>
```

For <label> we have a convention; it will be the name of the file with all its letters capitalized and changing the dot for a _ .

This leaves <label> as TOOLS_H

The second line will always be

```
#define <label>
```

Then come the declarations

And the last line will always be

```
#endif
```

Build the **Tools.h** file with its **#if-endif** and as a declaration add the one of our **Print_ln** function in between.

In P you will have to add the corresponding **include.** ***Pay attention*** *to the syntax that is not the same as the stdio include.*

1. Compile.
2. What errors does it give?
3. Why?
4. What is the solution?

Exercise 3 – Tools.c

Now we know that when the compiler says **reference to `Print_ln' without defining** it indicates that the body of the indicated function or, in its case, the variable does not exist in the project.

The first solution that will come to your mind is to go to P.c and define it. Today we are going to do it differently.

The project that we are creating consists only of the P.c module, but it is possible that we make the project consist of more modules and in one of them we put our Print_ln.

WARNING: *no matter how many modules the project has, ONLY ONE OF THEM can contain the main function. Can you give me a couple of reasons?*

Note: *As a general rule the main will be in the module that has the same name as the program we want to create; that is why the main is in P.c and the executable is called P.*

We are going to create the module **Tools.c** (it is not by chance the name because in it we are going to develop all the functions declared in **Tools.h** and we will do it this way because it is a good practice, although not a requirement of the language, this way we will have a module and its header file).

In this file we are going to:

- Make the include of **stdio** because we are going to use **printf**.
- Make the include of **Tools.c** because this way the compiler will warn us if there is no concordance between the declarations and the definitions. *Pay attention to the syntax of this include because it is not the same as the previous one.*
- We are going to define the body of Print_ln that only has to write the string it receives and add a newline.

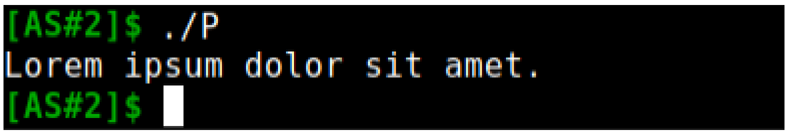
Compile

Since we are now incorporating two modules to the project we will have to add the new one in the compile command

```
gcc -Wall -g -o P P.c Tools.c
```

Run

Everything should work fine, the result should be:



```
[AS#2]$ ./P
Lorem ipsum dolor sit amet.
[AS#2]$ 
```

Additional

1. Try to make a mistake in the definition of the Print_ln, for example:

   ```
   void Print_ln(int i) {<whatever code>}
   ```

2. And then try to remove the include from Tools.h

Part II: Strings (Exercises 4, 5, 6)

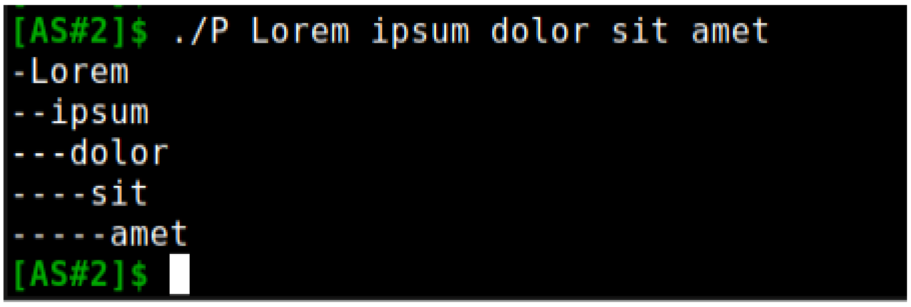
Remember that in C there is no String as we know them, instead there are *null terminated strings* that are arrays of characters whose end is indicated by the terminator character that we can write as the integer value 0 or the character '\0'.

Exercise 4 – main and TPrint

Modify P so that main loops to invoke TPrint by passing it each of the command line arguments, and the index number of that argument.

In the Tools module we are going to define the TPrint function that should create a loop to print as many “- “ as indicated by the second argument and then the first argument.

The result should be:



```
[AS#2]$ ./P Lorem ipsum dolor sit amet
-Lorem
--ipsum
---dolor
----sit
-----amet
[AS#2]$ 
```

Exercise 5 – Tinit and TPrint

We are going to define in the Tools module the variable tabs that will be an array of 20 characters. This variable does not need to be known by anyone else so we will not declare it in Tools.h (it is the closest we have to a private variable).

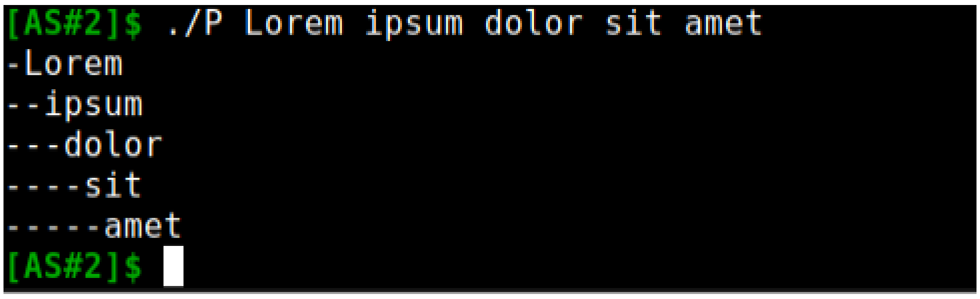
As we don't want to use literals for the size of the array (20) it should be the N_TABS tag set to 20.

In Tools we are going to add the function TInit that will receive an integer as an argument. What it will do is to fill tabs with as many “-” as the argument indicates and to make it a string at the end it will add a \0. Before it will verify that the “-” and \0 fit in the array. If not, it will save in tabs an empty string (the one whose first character is \0).

We have to modify TPrint so that it receives a single argument and prints tabs followed by that argument.

We have also to modify the main so that inside the loop it invokes first TInit with the index and then TPrint with the argument.

The result has to be the same:



```
[AS#2]$ ./P Lorem ipsum dolor sit amet
-Lorem
--ipsum
---dolor
----sit
-----amet
[AS#2]$ 
```

Exercise 6 - TPrint

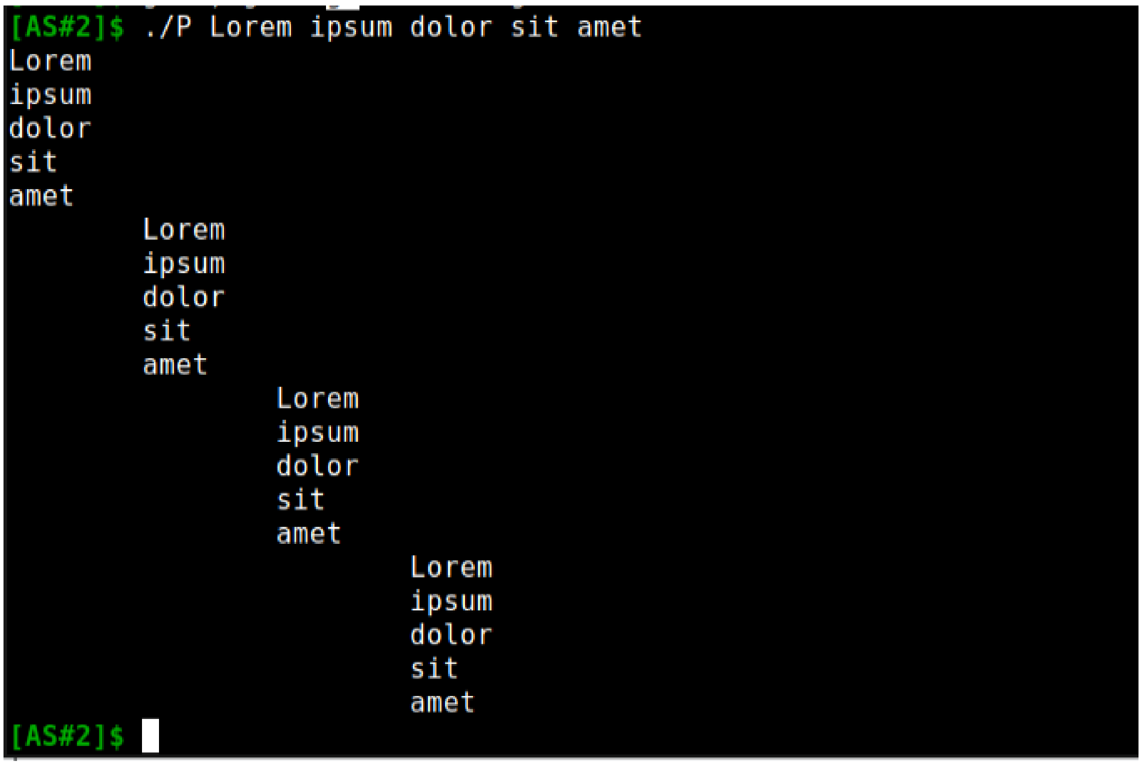
We are going to change the “-” of TPrint for tabulators (tabs).

And we are going to put in the main the following code:

```c
#include <stdio.h>
#include "Tools.h"

int main(int ac, char**av)
    {
    for (int i=0; i<4; i++)
        {
        TInit(i);
        for (int k=1; k<ac; k++)
            {
            TPrint(av[k]);
            }
        }
    }
```

When executed, the result should be:



```
[AS#2]$ ./P Lorem ipsum dolor sit amet
Lorem
ipsum
dolor
sit
amet
        Lorem
        ipsum
        dolor
        sit
        amet
                Lorem
                ipsum
                dolor
                sit
                amet
                        Lorem
                        ipsum
                        dolor
                        sit
                        amet
[AS#2]$
```

We have built a mechanism that allows us to place information in columns.

Part III: random and pauses (Exercises 7 and 8)

Exercise 7 - Random

In Tools we are going to add a function to Tools:

```
int Random(int min, int max);
```

It returns an integer that will be a random number between min and max.

- You should verify that min<max. If not, it will return min. You will have to use the rand() function. Go get documented on the internet.
- Modify the main so that it presents random numbers in the ranges (0-1000), (2000,2230), (1780,1800), (2310,4220) y (1000,999).

Exercise 8 - Pause

Let's add one more function to Tools:

```
void Pause(int nmilis)
```

That receives an integer and waits for that number of milliseconds, you will have to use the usleep function *(go get documented on the Internet).*

- Modify main so that after printing the random number, it pauses for that number of milliseconds and when it returns it prints a line of 10 dashes.

Let's add one more function to Tools:

```
void Pause(int nmilis)
```

Part IV: Pointers (Exercise 9)

Exercise 9 – Card_fill and Card_dump

We are going to create a new module that we will call Card. This module is going to declare the Card type that will be a structure with the following fields:

- id: an integer
- user: an array of NAME_SIZE characters
- nif: an array of NIF_SIZE characters
- n_items: an integer
- cost: a float

It will also contain the function:

```
            Card_fill(Card *p_card)
```

It will fill in the fields of the given card:

1. The first time it receives the call, this function completes the token with the following data: {1,”Pedro”, “08277292P", 7, 2.43}
2. The second time {2, "Wilma", "52489618W", 22, 134.97}
3. The third time {3, "Betty", "45781697B", 11, 234.197}
4. The fourth time {4, "Pablo", "45781697P", 43, 845.248}
5. The rest of the times, it will keep incrementing the id but will repeat sequentially the rest of the information from step 1 to 4.

You will need to document the **strncpy** function.

It will also contain the function:

```
            Card_dump(Card *p_card, char *array);
```

We assume that the array length >=sizeof(Card). This function will copy byte by byte the data from the card to the array.

You will need to document the **memcpy** function.

The main will do the following 10 times:

- Fill a card
- Dump the file
- Print, without line changes, the hexadecimal values of the dumped elements.

The result of the execution will be:

```
AS#2]$./P
01 00 00 00 50 65 64 72 6F 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 30 38 32 37 37 32 39 32 50 00 00 B7 07 00 00 00 1F 85 1B 40

02 00 00 00 57 69 6C 6D 61 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 35 32 34 38 39 36 31 38 57 00 00 B7 16 00 00 00 52 F8 06 43
```

```
03 00 00 00 42 65 74 74 79 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 34 35 37 38 31 36 39 37 42 00 00 B7 0B 00 00 00 6F 32 6A 43

04 00 00 00 50 61 62 6C 6F 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 34 35 37 38 31 36 39 37 50 00 00 B7 2B 00 00 00 DF 4F 53 44

05 00 00 00 50 65 64 72 6F 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 30 38 32 37 37 32 39 32 50 00 00 B7 07 00 00 00 1F 85 1B 40

06 00 00 00 57 69 6C 6D 61 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 35 32 34 38 39 36 31 38 57 00 00 B7 16 00 00 00 52 F8 06 43

07 00 00 00 42 65 74 74 79 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 34 35 37 38 31 36 39 37 42 00 00 B7 0B 00 00 00 6F 32 6A 43

08 00 00 00 50 61 62 6C 6F 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 34 35 37 38 31 36 39 37 50 00 00 B7 2B 00 00 00 DF 4F 53 44

09 00 00 00 50 65 64 72 6F 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 30 38 32 37 37 32 39 32 50 00 00 B7 07 00 00 00 1F 85 1B 40

0A 00 00 00 57 69 6C 6D 61 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 35 32 34 38 39 36 31 38 57 00 00 B7 16 00 00 00 52 F8 06 43

[AS#2]$
```