Lab 1
Systems Architecture

Second Course.
Bachelor's Degree in Telecommunication Technologies Engineering
Iria Estévez Ayres
Universidad Carlos III de Madrid.
2023–2024

In order to correctly follow the lab, it is recommended to read in advance the following electronic resources:

- Reading about the most used options of the gcc compiler `https://labgastt.it.uc3m.es/course_notes/main_en.html#compiler`
- Reading about the compiler errors and warnings `https://labgastt.it.uc3m.es/course_notes/main_en.html#gcc_errors`
- printf function explanation at `https://labgastt.it.uc3m.es/course_notes/main_en.html#input_output_printf`

This lab statement includes all the code snippets needed to its deveolpment.

Exercise 1. The compiler

Resources:

- Reading about the most used options of the gcc compiler `https://labgastt.it.uc3m.es/course_notes/main_en.html#compiler`
- Reading about the compiler errors and warnings `https://labgastt.it.uc3m.es/course_notes/main_en.html#gcc_errors`

Workplan:

1. Before you go to class, read the resources above and try to answer the following questions:
   (a) What is the `-o` option for?
   (b) Give an example of a warning that is only shown with the `-Wall` option of `gcc`.
   (c) What is the `-g` option for?
   (d) How can you know the time it takes to execute a command?
   (e) If you get the following warning message when compiling:
   `Warning: implicit declaration of function printf`
   What is happening in our code?
2. At class, check the answers with your lab partner.
3. Copy this program into an editor and name it first.c

```c
int main(int argc, char **argv)
{
  a = 1;
  printf("Hello\n");
}
```

4. Compile it

```
$ gcc -g -Wall -o first first.c
```

5. Looking only at the error messages that the compiler provides by console, explain the errors you find, what they mean and the line where each one is found.

Exercise 2. Printf function

Resources:

- printf function explanation at `https://labgastt.it.uc3m.es/course_notes/main_en.html#input_output_printf`
- Manpage of `printf` (type in the console `man 3 printf`).
- File `my_printf.c` that you should copy into and editor

```c
#include <stdio.h>

int main()
{
    char letter = 'm';
    int num = 60345698;
    int hexadecimal = 0x3FA;
    float float_num = 3.1e33;
    char *string = "ABCDEFGHIJKL";

    printf("%c\n", letter);
    printf("%d\n", num);
    printf("%d\n", hexadecimal);
    printf("%f\n", float_num);
    printf("%s\n", string);
    return 0;
}
```

Workplan:

1. Compile and execute the file `my_printf.c`

```
$ gcc -g -Wall -o my_printf my_printf.c
$ ./my_printf
m
60345698
1018
309999992877687786919139091677184 0.000000
ABCDEFGHIJKL
$
```

2. Modify the format strings used in the printf function to print in the following format. Do this without typing two or more consecutive blank spaces, you can use the tab character or use the format string to indicate the number of spaces you want to print and the number of characters to print.

```
$ ./my_printf
The letter is                m
The number is               60345698
The hexadecimal number is 0X3FA
The real number is          3.10e+33
The string is               ABCDE
$
```

Exercise 3. Using command line arguments

Resources:

- `print_arguments.c` file that you shoold copy into an editor.

```c
#include <stdio.h>

int main(int argc, char **argv)
{
    printf("Hello\n");
    return 0;
}
```

Workplan:

1. Compile and execute the file `print_arguments.c`

```
$ gcc -g -Wall -o print_arguments print_arguments.c
$ ./print_arguments
Hello
$
```

2. Edit the program so that it prints out the string `Hello` followed by the arguments sent through the command line. Each argument should be printed in a separate line. In the following example the executable file is called `print_arguments`:

```
$./print_arguments each word is an argument
Hello:
each
word
is
an
argument
$
```

3. Modify your program again so that it also prints the program name and argument number.

```
$ ./print_arguments one two three words
Hello
Program name = ./print_arguments
ARG[1] = one
ARG[2] = two
ARG[3] = three
ARG[4] = words
$
```

4. Create the file `triangle.c`. The program should print on the screen the arguments received through the command line. The printout will have the shape of a triangle: The arguments will be repeated along several lines, removing from each line the first argument of the previous line. The executable file in the following example is called `triangle`:

```
$./triangle arguments follow a triangular pattern
arguments follow a triangular pattern
follow a triangular pattern
a triangular pattern
triangular pattern
pattern
```