Lab 2: Introduction to C Programming

COEN-4720 Embedded Systems

Cris Ababei
Dept. of Electrical and Computer Engineering, Marquette University

1. Objective

The objective of this lab is to introduce you to C programming. This is done by writing several programs in C and discussing how to compile, link, and execute on Windows or Linux. The first example is the simplest "hello world" example. The second example is designed to expose you to as many C concepts as possible within the simplest program. It assumes no prior knowledge of C. However, you must allocate significant time to read suggested materials, but especially work on as many examples as possible.

2. Example1: Working on Windows

To get started with learning C, I suggest using a simple and general/generic IDE with a free C/C++ compiler. For example, CodeBlocks and Dev-C++ are nice IDEs for windows. I recommend to use CodeBlocks, which you can download and install on your own laptop/PC from the link below.
http://www.codeblocks.org/
If you do not have an already installed compiler on your computer you should download and install the bundle: codeblocks-20.03mingw-setup.exe

Or you could download just the CodeBlocks IDE: codeblocks-20.03-setup.exe
Install it, and then, separately download a Compiler, which you could install separately.
For example, you could download and install MinGW/GCC compiler as explained here:
https://wiki.codeblocks.org/index.php/Installing_a_supported_compiler

In my case, I installed it in **M:\CodeBlocks**, where I created a new directory **M:\CodeBlocks\cristinel** to store all projects I will work on.

We create a program by creating a new project in CodeBlocks. So, start CodeBlocks and create a new project. In the dialog window, select "Console Application" and click Go. Then select C as the type of language you want to use and click Next. Then, type hello as the Project title, keep **M:\CodeBlocks\cristinel** as the Folder to create the project in. Click Next and then click Finish.
This will automatically create a simple **main.c** template file. Replace its content with the following listing:

```
#include <stdio.h>
int main ()
{
  printf( "Hello World!\n");
  return 0;
}
```

Then save it as **hello.c**. It could be very well saved as **main.c** (the default name) but we just want to change its name. This first program will simply print "Hello World!" at the command prompt.

To compile this program, select **Build->Build**
Which creates the executable **hello.exe** in **M:\CodeBlocks\cristinel\hello\bin\Release**

To execute our program, start a command prompt window **Start->Run** and type **cmd**. Then navigate to the project directory **M:\CodeBlocks\cristinel\hello\bin\Release**, where simply type **hello.exe**. This should execute our program, which simply prints at the prompt "**Hello World!**"

A few notes about this first program:

- The #include command is a preprocessor directive to load the stdio library.
- Execution starts in a function (method) called main:
- There are other signatures for main as we will see.
- Although a return type is declared, nothing needs to be returned.
- The printf method is used for screen output.
- The "newline" character \n is explicitly required.

3. Example1: Working on Linux/Unix

If you have a Linux machine, then you can simply use the GNU compiler. You do not need to install anything else. Use your favorite text editor to create and save the **hello.c** file in a new directory, say **hello/**.

To compile it:

```
$ gcc -Wall hello.c -o hello
```

This produces an executable called **hello** which can be executed as:

```
$ hello
```

or, if you don't have the current directory in your path:

```
$ ./hello
```

Which should print at the prompt "**Hello World!**"

One can also work in Linux-like environments on Windows. For example, **cygwin** is a collection of tools which provide a Linux look and feel environment for Windows. If you want, you can install cygwin and use it for your C programming projects. There is lots of online information that describes how cygwin can be used. Google and search for it. You can download cygwin here: http://www.cygwin.com/

4. Example2: Working on Windows

A program can be split up into multiple files. This makes it easier to edit and understand, especially in the case of large programs. In addition, it also allows the individual parts to be compiled independently.

In our second program, we split up the program into three files, which we create and store in a new project directory in **M:\CodeBlocks\cristinel\newhello**:
main.c
my_utils.c
my_utils.h
For the purposes of this example (which will become clear when you will read the **main.c** source file), also create a simple text file (using any text editor you would like, such as NotePad) called **age.txt** with only one integer in one line. Store this file into: in **M:\CodeBlocks\cristinel\newhello\bin\Release**, because that's where the executable will be created; note that this folder will be created only after you Build for the first time your new project.

The listing of **main.c** is:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "my_utils.h"
```

```
#define NUM_TIMES_TO_PRINT 3

int main(int argc, char *argv[])
{
    // (1) declare variables;
    struct USER_RECORD myself;
    char temp_last_name[50];
    FILE *fp; // file pointer; used to read from pre-saved age.txt;

    // (2) ask user to type in her first name; read from standard input
    // what the user types and store up to 100 characters in member "first_name"
    // of object "myself";
    printf( "Type your first name please and press Enter: \n");
    // read 100 bytes from standard input;
    fgets( myself.first_name, sizeof(myself.first_name), stdin);
    // remove the newline character from the end of the string;
    remove_newline( myself.first_name);

    // (3) ask user to type in her last name; read from standard input
    // what the user types and store what the user types in member "last_name"
    // of object "myself";
    printf( "Type your last name please and press Enter: \n");
    fgets( temp_last_name, sizeof(temp_last_name), stdin);
    // remove the newline character from the end of the string;
    remove_newline( temp_last_name);
    // allocate memory to store the last name; needed because last_name
    // variable inside the USER_RECORD structure is declared as a pointer only;
    myself.last_name = malloc( (strlen(temp_last_name) + 1) * sizeof(char));
    strcpy( myself.last_name, temp_last_name);

    // (4) ask user the gender;
    printf( "What's your gender? Type F or M. \n");
    myself.gender = getc( stdin);

    // (5) in this example's directory we have created a text file
    // that contains only one line with an integer number; we mean for
    // this number to be the age of the user; we do this only to
    // also illustrate reading from a file;
    fp = fopen("age.txt", "r"); // you should error-check this; not done here;
    fscanf( fp, "%d", &myself.age);
    fclose(fp);

    // (5) entertain user;
    printf("--- \n");
    long i;
    for ( i=0; i < NUM_TIMES_TO_PRINT; i++) {
        print_hello_message( &myself);
    }

    // (6) clean up and return;
    free( myself.last_name);
    return 0;
}
```

The listing of **my_utils.c** is:

```
#include <stdio.h>
#include <string.h>
#include "my_utils.h"

void print_hello_message( struct USER_RECORD *this_user)
{
    if ( this_user->gender == 'F') {
        printf( "Hello MRS. %s %s, you are %d years old! \n",
            this_user->first_name, this_user->last_name, this_user->age);
```

```
    } else if (this_user->gender == 'M') {
        printf( "Hello MR. %s %s, you are %d years old! \n",
            this_user->first_name, this_user->last_name, this_user->age);
    } else {
        printf( "Hello %s. No soup for you! \n", this_user->first_name);
    }
}

char *remove_newline( char *s)
{
    int len = strlen( s);
    // if there's a newline truncate the string
    if (len > 0 && s[len-1] == '\n') {
        s[len-1] = '\0';
    }
    return s;
}
```

The listing of **my_utils.h** is:

```
struct USER_RECORD {
    char first_name[100];
    char* last_name;
    char gender;
    int age; // in years;
};

void print_hello_message( struct USER_RECORD *this_user);
char *remove_newline( char *s);
```

The file **my_utils.c** contains the source of two functions, which we utilize by calling them from within the **main.c** file. Our program also includes the header file **my_utils.h**, which contains the declaration of the two functions defined in **my_utils.c**. These declarations are used to ensure that the types of the arguments and return value match up correctly between the function calls and the function definitions.

The difference between the two forms of the include statement #include "FILE.h" and #include <FILE.h> is that the former searches for FILE.h in the current directory before looking in the system header file directories. The include statement #include <FILE.h> searches the system header files, but does not look in the current directory by default.

Now, start **CodeBlocks** and create a new project called **newhello** in **cristinel/** directory. Add the above three files to the project and build. Do not forget to also create the **age.txt** file and store it in **newhello/** before executing the program. Start a command prompt window, then navigate to **M:\CodeBlocks\cristinel\newhello\bin\Release** and execute **newhello.exe**. Observe and study.

5. Example2: Working on Linux/Unix

Create a new directory, say **project2/** (it does not necessarily need be called newhello/), where you should copy the three files:
main.c
my_utils.c
my_utils.h
Do not forget to also create the **age.txt** file and store it in **project2**.

To compile these source files with gcc, we use the following command:

```
$ gcc -Wall main.c my_utils.c -o newhello
```

This produces an executable called **newhello** which can be executed as:

```
$ newhello
```

or, if you don't have the current directory in your path:

```
$ ./newhello
```

6. Lab Assignment

Part 1:
Read the source code of the second example. Identify the lines of code inside the main() function that you did not know before what they would do. Search more about them (you can start with the resources from [1]), get an understanding of what they do, and briefly explain those lines of code only in your lab report. If you claim that you knew everything, then, just state that in this Part 1 of your report.

Part 2:
Write a new C program that reads two matrices from two different text files (**matrixA.txt**, **matrixB.txt**), multiplies the two matrices, and writes the result into a third text file (**matrixC.txt**).

The listing of file **matrixA.txt** is:

```
1 2 3 4
8 7 6 5
```

The listing of file **matrixB.txt** is:

```
1 1
1 1
1 1
1 2
```

Part 3 (optional but recommended):
Search online for simple examples C programs and verify them. Learn via examples found online and/or discussed in the above C Programming pointers.

You must write (typed, not handwritten!) a report to describe what you did, what problems you faced and how you solved them. Upload the report on D2L using the file naming convention: **lab2_report_lastname.pdf**.

7. Others

CodeBlock is only one example of "C compiler + IDE (integrated development environment)" that one can use on Windows. There are many others out there. You can search for others, read online about them (advantages, disadvantages) and select to work with whichever you feel more comfortable. Some examples include those listed in [2].

The compiler of choice for most users of the Linux operating system is the GNU C++ Compiler (GCC), although many others are available too. Linux is primarily a console-based environment. So GCC is a command-line compiler rather than a point-and-click GUI.

Most Mac users recommend the G++ compiler which is part of the free Xcode package. There is also a version of Eclipse for the Mac.

8. Credits and References

[1] C Programming pointers:
--C Tutorial, http://www.cprogramming.com/tutorial/c-tutorial.html
--Teach Yourself C in 21 Days, http://kldp.org/files/c+in+21+days.pdf

[2] Additional IDEs:
--Eclipse http://www.eclipse.org/cdt/
--Visual C++ Express http://www.microsoft.com/visualstudio/eng/products/visual-studio-express-products
--See many more on the list(s) at https://www.bloodshed.net/