# Homework. 1: Bash and Build System

Tiziano Guadagnino, Saurabh Gupta, E-Mail tiziano.guadagnino@igg-uni-bonn.de

Handout : 30.10.2023
Handin: 05.11.2023 at 23:59:59 (CET)

To do this homework you will need to download the files from e-Campus. All the needed files are in the `homework_1.zip` file.

Once you have forked `https://gitlab.igg.uni-bonn.de/teaching/cpp-homeworks` and cloned your own repository extract the `homework_1.zip` archive into `cpp-homework/homework_1` folder:

```
$ git clone https://gitlab.igg.uni-bonn.de/<YOUR_USER_NAME>/cpp-homeworks
$ cd cpp-homeworks/homework_1
$ mv ~/Downloads/homework_1.zip . # or replace ~/Downloads with your path
$ unzip homework_1.zip && rm homework_1.zip
```

Once you successfuly extracted the files, your working directory should look like the following:

```
|-- homework_1
|   |-- task_1
|   |   `-- test_folder
|   `-- task_2
|       |-- include
|       |   `-- ipb_arithmetic
|       |-- results
|       |   |-- bin
|       |   `-- lib
|       `-- src
|-- homework_2
|-- ...
```

## A Using the terminal (2 points)

This exercise focuses on using the terminal efficiently.
Every question in this exercise must be answered with a command on a single line. You should save each of these lines into the file `homework_1/task_1/commands.sh`.

Make sure all commands run from within `homework_1/task_1/` folder correctly.

1. Count how many lines are there in "`data.dat`".
2. Count how many lines of those contain "`dolor`" or "`dalor`"?
3. Count how many words are there in "`data.dat`"?
4. Count how many of those start with "`mol`"?

*Hint:* you might want to use `wc` command.

## B Build Systems Exercise (8 points)

### B.1 Exercise 1: Build by hand

In this exercise you should suffer a little bit. Build the library and the main program in a really **old fashion** way.
When you invoke the compiler, it normally does preprocessing, compilation, assembly and linking. The overall options allow you to stop this process at an intermediate stage. For example, the -c option says not to run the linker. Then the output consists of object files output by the assembler.

- This means that you will need to use the `-c` flag to compile objects without the need of the linker
- You will also need to specify the compiler where to look for header files, use the `-I./include/` option for this purpose.
- To pick a proper name for this artifact, use the `-o` option

### Prepare your results for submission

This is the easiest task, once you have manually completed all the stepts, and you are completly sure that you've built both the library and the example program sucessfuly, you need to provide a **build script**.
What is this? In your case will be a simple **bash** script with all the comands you need to run, taking into account that the starting working directory is the root of this repository. Here is a **build.sh** example

```
#!/usr/bin/env bash
echo "This is a bash script, this is the first line"
echo "This is the second one, it will run after the first one"
```

NOTE: make sure your build script is called **build.sh** and it's located on the root of this reposiotry.

### B.1.1 Build the ipb_arithmetic library

Create an empty directory in your source tree, let's call it `./build`. As shown on the lecture you need to create a static library. Once you have the two objects built for the library your **build** directory should look something like:

```
build
|-- subtract.o
`-- sum.o
```

### B.1.2 Build the example program

Now you can build your main program, for this, you will need to link this program with the library you have just created. After your build is ready, you must places the binaries that came from the build process in the `results/` directory.

## B.2 Exercise 2: Welcome to Make

After suffering with manual building, you will get the chance to improve a bit your experience by using one of the most popular build systems, **Make**.

This exercise is rather simple, you need to re-do the Exercise 1(manual build) but using **Make**. For this purpose, you must first read the official documentation, and then write a **Makefile**. In particular, we selected a subset of the documentation, so you do not need to read it all:
Please read from `https://www.gnu.org/software/make/manual/make.html#Rule-Introduction`:

- Chapter 2
- Chapter 3 sections 3.1 and 3.2
- Chapter 4 sections 4.1 4.2 4.3 4.4 4.5 4.6

## B.3 Exercise 3: Welcome to CMake

**NOTE**: CMake is always evolving and changing, this means than when searcihng for online documentation you should always check which version of the documentation you are reading, otherwise you will struggle a lot. On **Ubuntu 20.04** it's version **3.21.3**, this means that this is the manual you should read: https://cmake.org/cmake/help/v3.21/

Building everything manually is tough right? **CMake** is pretty popular nowadays and it's the defacto build system generator for **C++** projects.

Complete the Exercise 1 but this time using **CMake**. Remember, google is your friend in these cases.
You need to provide as many **CMakeLists.txt** files as you think are necessary, just putting the right files in your repository and making sure that everything builds properly would be enough.

## B.4 Exercise 3: Install your library

It's very unlikely that you will be writing libraries during this course,but it's a good idea to have some insights of what's going on whenever you install someone else's library on your system.

Update your `CMakeLists.txt` files to add a new target such as when you run `make install` the build system will install the required files for you on the `./install/` directory.

## B.5 Hints

**HINT 1**: You should use the `CMAKE_INSTALL_PREFIX` variable to point to the local `./install` directory, otherwise `cmake` will attempt to install the library on your system(and for that you need super user access)

**HINT 2**: the `install` function might be of help

**HINT 3:** After solving the build system exercises myself, this is how my working directory looks like (before running `cmake`):

```
|-- build
|   |-- libipb_arithmetic.a
|   |-- main.o
|   |-- subtract.o
|   |-- sum.o
|   `-- test_ipb_arithmetic
|-- build.sh
|-- CMakeLists.txt
|-- include
|   |-- ipb_arithmetic
|   |   |-- subtract.hpp
|   |   `-- sum.hpp
|   `-- ipb_arithmetic.hpp
|-- install
|   |-- include
|   |   |-- ipb_arithmetic
|   |   |   |-- subtract.hpp
|   |   |   `-- sum.hpp
|   |   `-- ipb_arithmetic.hpp
|   `-- lib
|       `-- libipb_arithmetic.a
|-- LICENSE
|-- README.md
|-- results
|   |-- bin
|   |   |-- example_output
|   |   `-- test_ipb_arithmetic
|   `-- lib
|       `-- libipb_arithmetic.a
`-- src
    |-- CMakeLists.txt
    |-- main.cpp
    |-- subtract.cpp
    `-- sum.cpp
```