Build+Link Exercise [6 points in total]

This repository is for those who just started working on software development and don't have much knowledge about build systems. Build systems are just **tools**, and one shouldn't get lost learning how to deal with it, it is what it is, a tool. And one just need to learn how to use it.

Table of Conents

What is the exercise about

You have one very **very** simple main program, that invoke 2 methods from one library **ipb_arithmetic**. You need to build the main program, the library objects and link all together for a desktop target.

There is an example binary in order to try it before you start working.

Directory structure

I tried to make the directories as similar as they would look in real world applications:

Source Files This is usually the place where you put your source code:

- src
- subtract.cpp
- sum.cpp
- main.cpp

Include This is usually where the API of your application/library is.

- include
- ipb_arithmetic.hpp

Results Artifacts Here you should put the output binaries and the output libraries.

- results
- lib
- bin
    - binary_example

NOTE : If it happens that you are working with third part libraries,that are not open-source, it's very likely that you will be given only the header files (`#include`) and the result artifacts: the library and some binaries. It's quite a good idea to keep this in mind, and know how thoe libraries were built.

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 an example

build.sh:

```
#!/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.

Task 1: Build the `ipb_arithmetic` library [1 points]

1. Create an empty directory in your source tree, let's call it `./build`.
2. Compile the two objects needed for building the library **ipb_arithmetic**: So far your build command should look something like:

```
sh c++ -c -Idir/ src/<filenmae.cpp> -o build/<name.o>
```

Once you have the two objects built for the library your `build` directory should look something like:

```
sh build ├── subtract.o └── sum.o
```

1. Now it's time to make an archive of these two objects and create your first library:

```
sh # always start the name of your library with 'lib' ar rcs build/
libipb_arithmetic.a build/sum.o build/subtract.o
```

Task 2: Build the example program [1 points]

Now you can build your main program, for this, you will need to link this program with the library

you have just created. You can see an animated example of how to achieve this: Link C++ library

After your build is ready, you must places the binaries that came from the build process in the `results/` directory.

Exercise 2: Welcome to `CMake` [2 points]

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 18.04` it's version `3.10.2`, this means that this is the manual you should read: https://cmake.org/cmake/help/v3.10/

Building everything manually is tough right? That's why humans have created something that we call build-systems. `CMake` is pretty popular nowadays and it's the defacto build system generator for `C++` projects.

Complete the **Exercise 1** but use `CMake` this time, remember, google is your friend in these cases.

In this case you need to provide as many `CMakeLists.txt` files as you think are neccessary, just putting the right files in your repository and making sure that everything builds propperly would be enough.

Exercise 3: Install your library [2 points]

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.

1. Create a `./install/` directory on this same repository
2. Provide a `install.sh` script that install all the neccessary files to ship your library and share it it others(basically, everything that's in the `results` directory, now you see the convinience of having this directory)
3. 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.

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

After solving this 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

11 directories, 23 files
```