

# Guide to compile and run your practicals

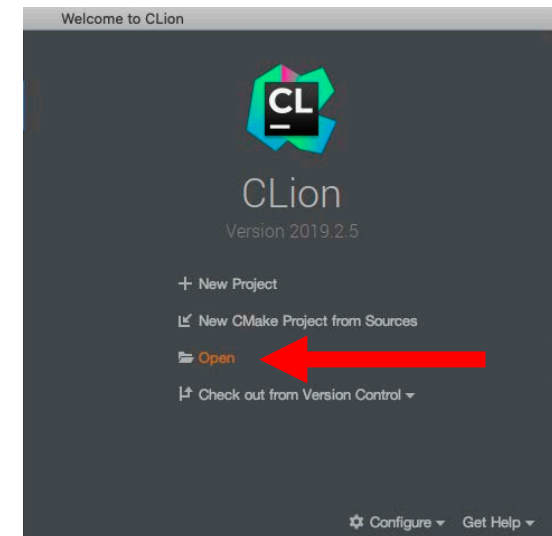
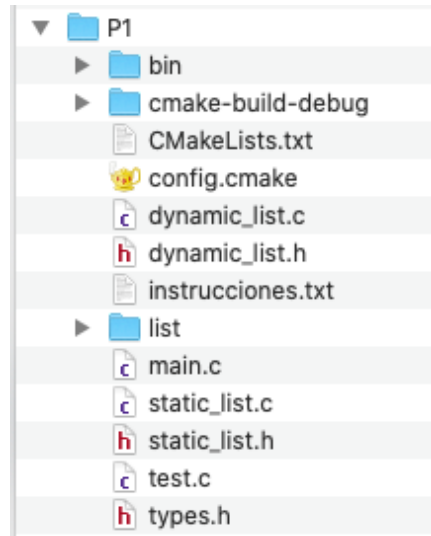
Programming II · Year 2019-20

Faculty of Computer Science



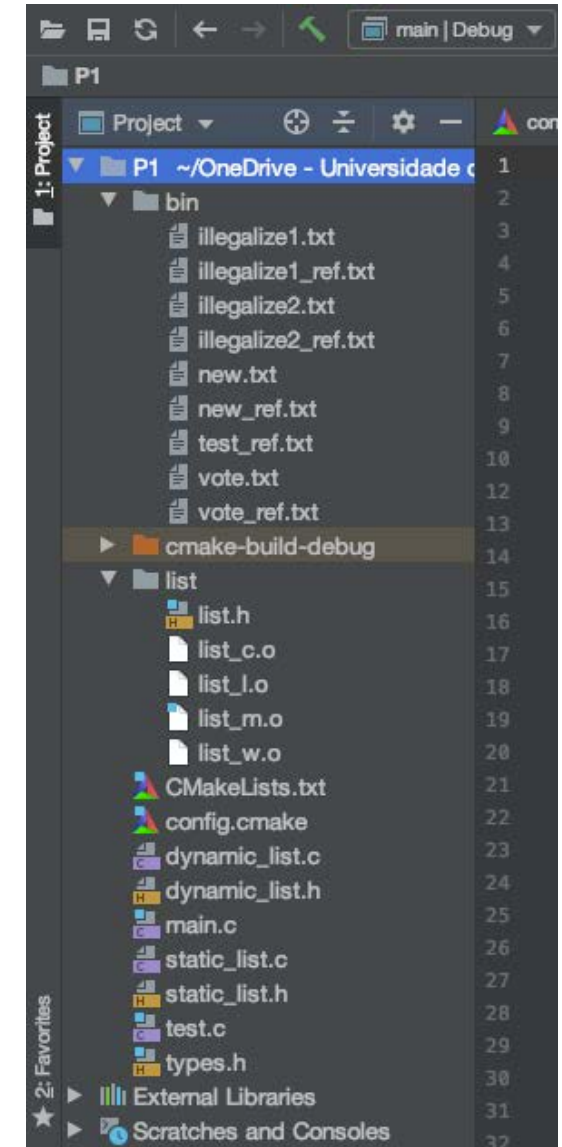
# Template project

1. The first step will be to download and decompress the ZIP file available at Moodle that contains all the files necessary for the practical.
2. Next, open (*Open*) from CLion the folder containing the files of the practical.
  - This folder already includes the definition of a CLion project **ready to compile**.



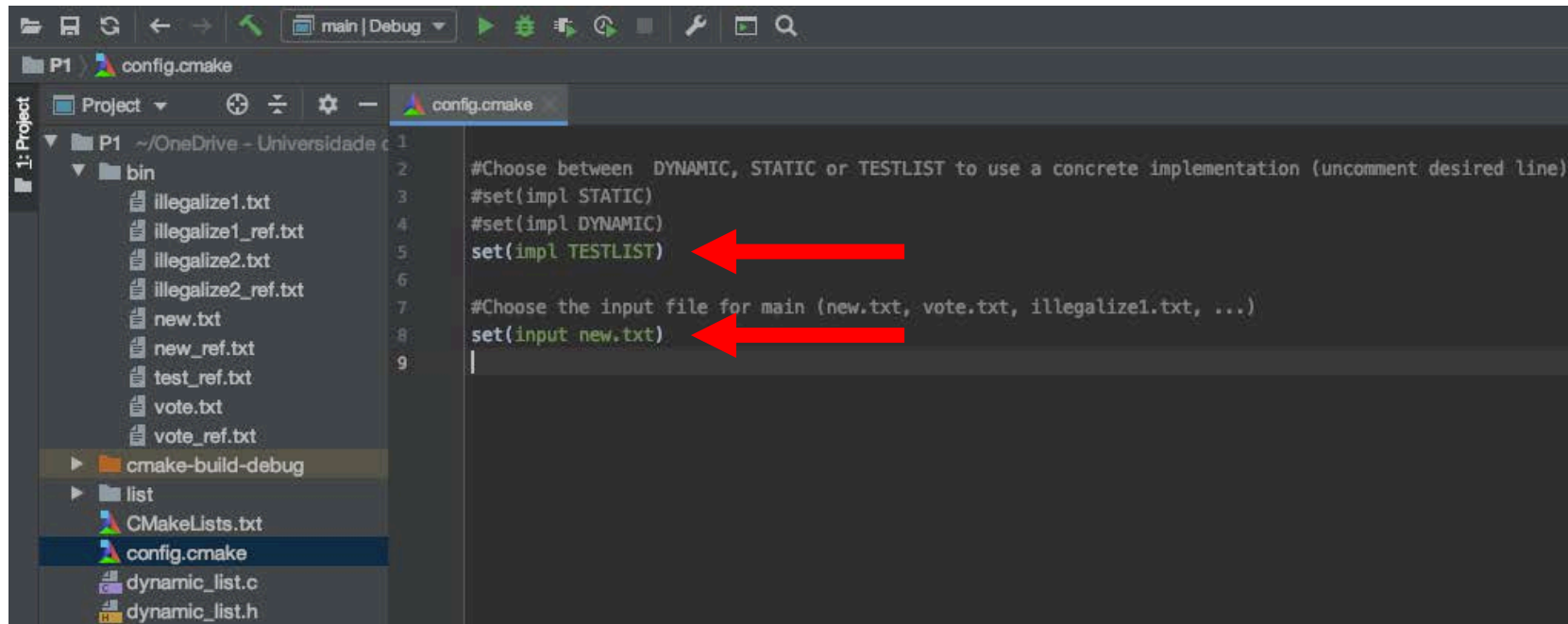
# Template project

- The project includes all the C source files necessary to carry out the practical. **The student must not add any other .c or .h file.**
- **Folder bin:**
  - Where the executables are generated when compiling.
  - Contains the input test files necessary to test the operations of the practical ( .txt files).
- **Folder list:** contains (compiled for Windows, Linux and MacOS) the library of the ADT list to be used at the beginning of the practical.
- **main.c, types.h, dynamic\_list.h, dynamic\_list.c, static\_list.h, static\_list.c**
  - Source files to be completed to carry out the practical.
- **test.c**
  - Program for testing an ADT List.
- **CMakeLists.txt**
  - Configuration file for CMake. **It must not be modified.**
- **config.cmake**
  - File to be modified when testing the program. It must be made according to the instructions included.



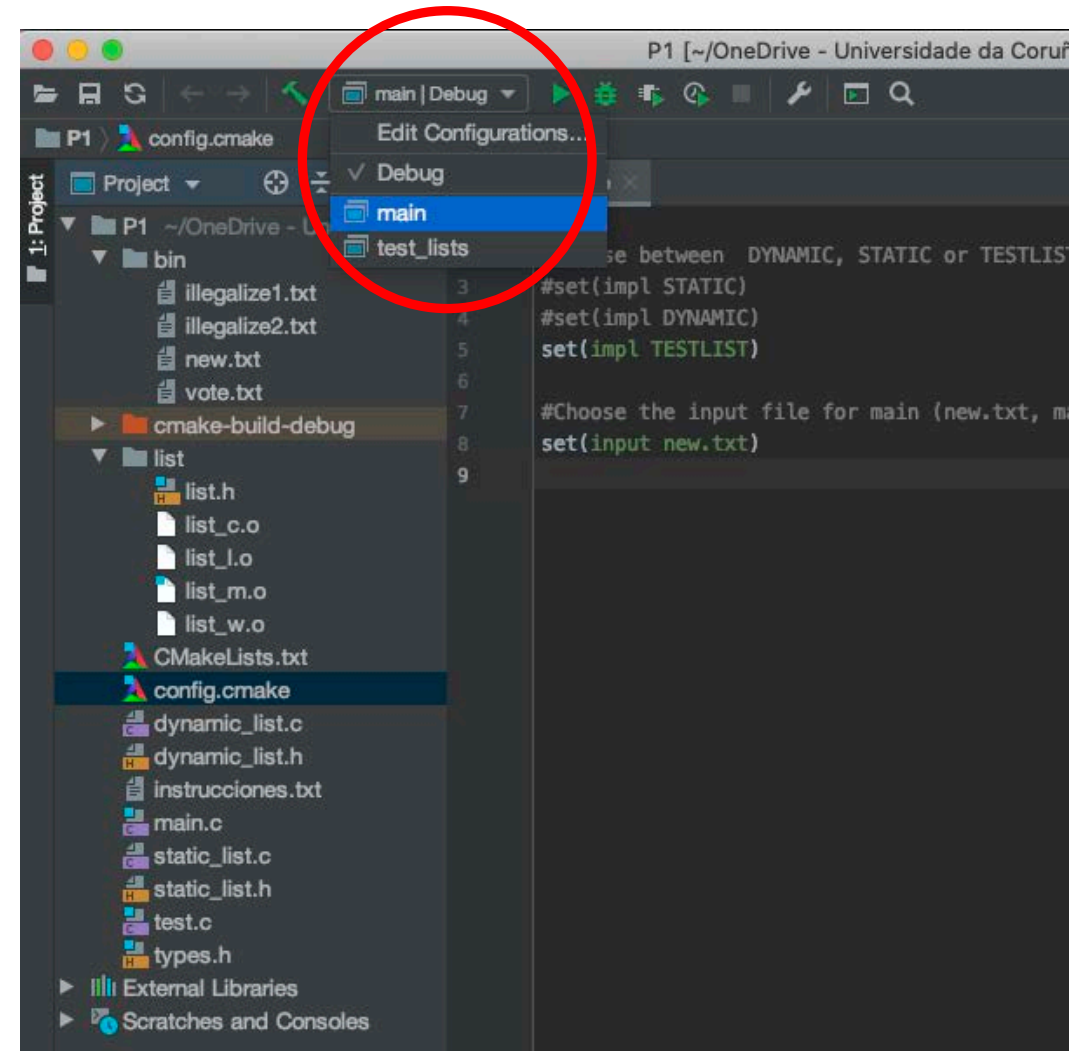
# Configuring compilation and execution

- Go to file **config.cmake** and select the following options (by commenting or uncommenting the corresponding lines):
  - Version of the ADT List implementation to be used to compile and run the program.
  - Input file for the program (to be chosen among the **.txt** files in folder **bin**).



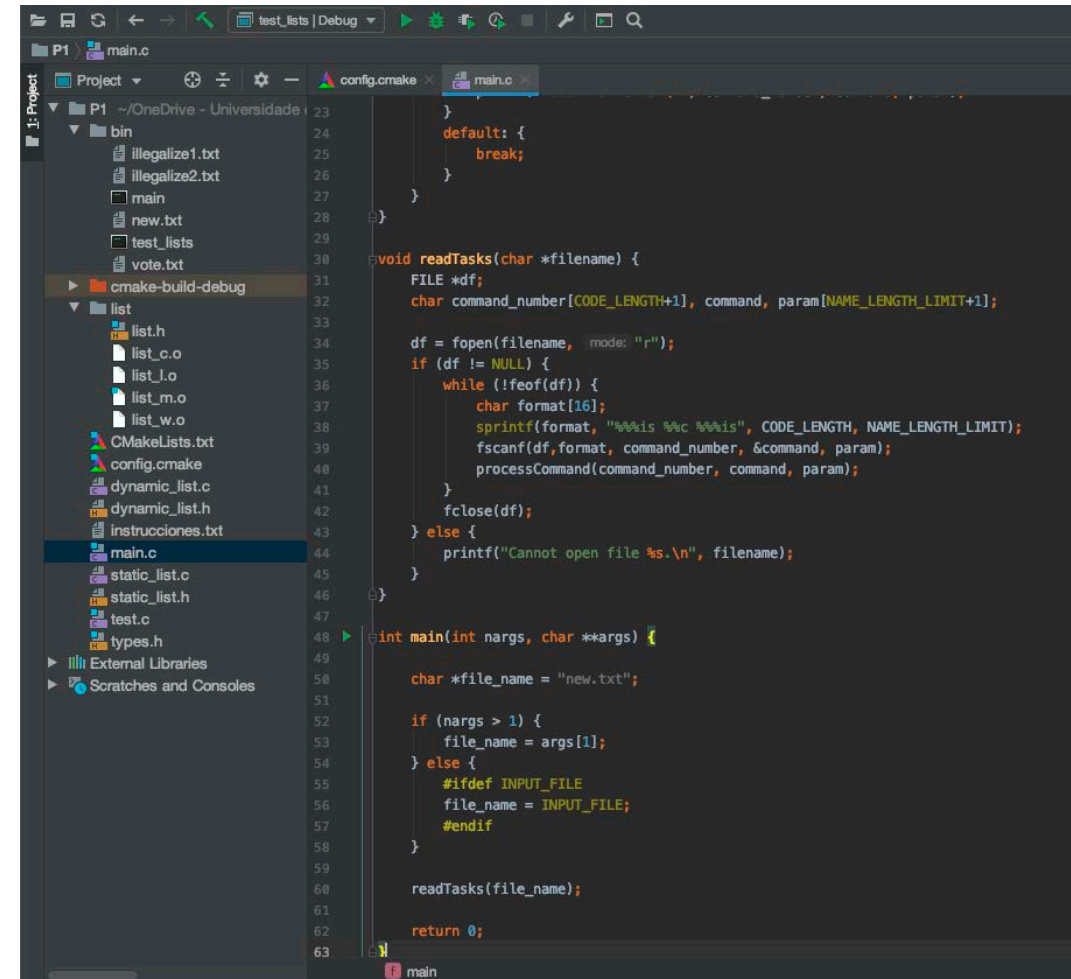
# Configuring compilation and execution

- The template project is already configured with two compilation and run *targets*:
  - **main**: to compile and run the main program of the practical.
  - **test\_lists**: to compile and run the testing program that checks the proper functioning of the ADT List implementations.
- You must select the corresponding one at the CLion drop-down menu according to which one you want to **compile** or **run**.



# Programming the code of the practical

- Complete the C source code of the **.c** and **.h** files included with the project.
- File **main.c** contains the basis source code of the main program, including the code necessary for reading the input files.



The screenshot shows a CMake IDE interface. On the left, the 'Project' pane displays the file structure of the project 'P1'. The 'list' directory is expanded, showing files like list.h, list.c.o, list\_i.o, list\_m.o, list\_w.o, CMakeLists.txt, config.cmake, dynamic\_list.c, dynamic\_list.h, instrucciones.txt, main.c, static\_list.c, static\_list.h, test.c, and types.h. The 'main.c' file is selected. The main editor on the right shows the code for 'main.c'. The code includes a function 'readTasks' and a 'main' function. The 'main' function takes 'nargs' and '\*\*args' as arguments. It sets 'file\_name' to 'new.txt' by default, or to 'args[1]' if 'nargs' is greater than 1. It then calls 'readTasks(file\_name)' and returns 0. The 'readTasks' function takes a 'filename' and opens it in 'r' mode. It reads the file line by line, parsing the command and parameters, and then calls 'processCommand'.

```
23 }
24
25 }
26
27 }
28
29
30 void readTasks(char *filename) {
31     FILE *df;
32     char command_number[CODE_LENGTH+1], command, param[NAME_LENGTH_LIMIT+1];
33
34     df = fopen(filename, "r");
35     if (df != NULL) {
36         while (!feof(df)) {
37             char format[16];
38             sprintf(format, "%16s %c %16s", CODE_LENGTH, NAME_LENGTH_LIMIT);
39             fscanf(df, format, command_number, &command, param);
40             processCommand(command_number, command, param);
41         }
42         fclose(df);
43     } else {
44         printf("Cannot open file %s.\n", filename);
45     }
46 }
47
48 int main(int nargs, char **args) {
49
50     char *file_name = "new.txt";
51
52     if (nargs > 1) {
53         file_name = args[1];
54     } else {
55         #ifdef INPUT_FILE
56         file_name = INPUT_FILE;
57         #endif
58     }
59
60     readTasks(file_name);
61
62     return 0;
63 }
```

# Running from the console

- It is also possible to run the program from the console using the compiled binary files (generated at folder **bin**).
- Throughout the development of the practical, a **script** for the automatic testing of the program will be provided. This script will have to be run from the console.

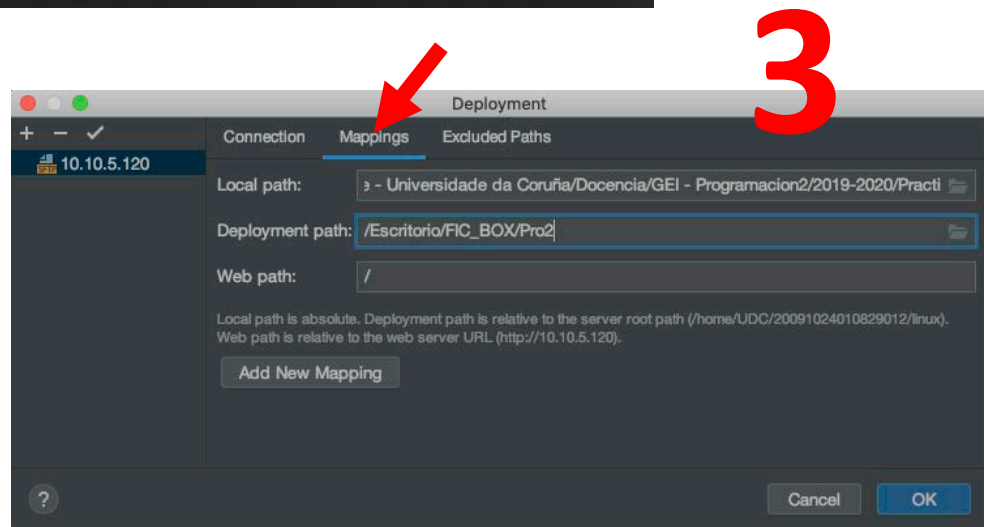
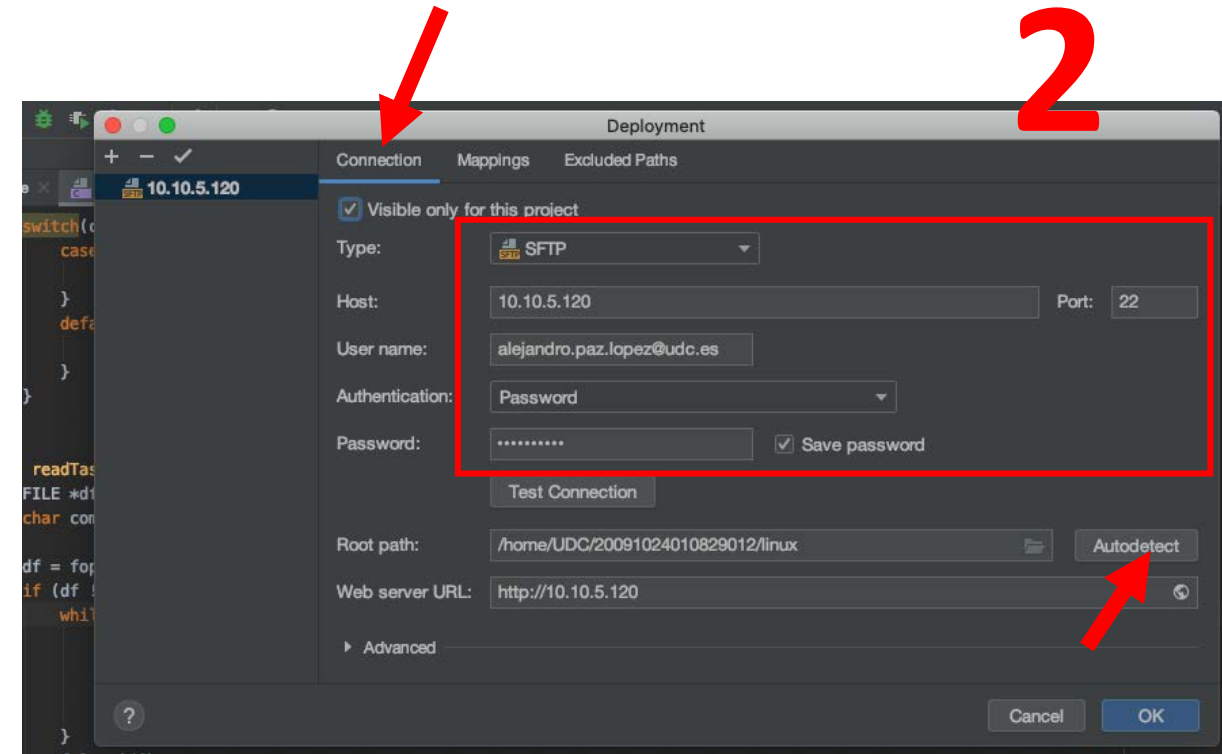
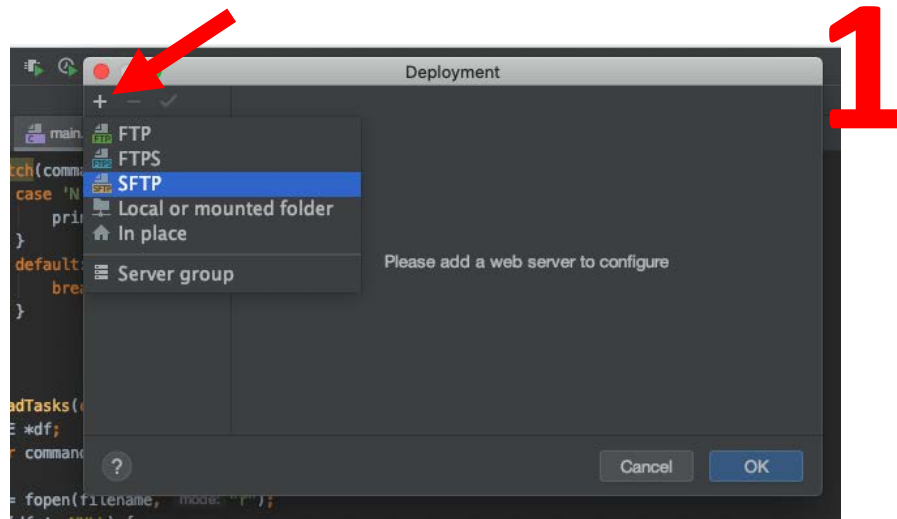
```
(base) taliesyn:P1 alex$ cd bin/  
(base) taliesyn:bin alex$ ll  
total 112  
-rw-r--r--  1 alex  staff   95B 21 ene 11:54 illegalize1.txt  
-rw-r--r--  1 alex  staff   95B 21 ene 11:54 illegalize2.txt  
-rwxr-xr-x  1 alex  staff  18K 10 feb 11:26 main*  
-rw-r--r--  1 alex  staff   74B  4 feb 13:25 new.txt  
-rwxr-xr-x  1 alex  staff  18K 10 feb 11:26 test_lists*  
-rw-r--r--  1 alex  staff   95B 21 ene 11:54 vote.txt  
(base) taliesyn:bin alex$ ./main new.txt
```

# Configuring CLion to synchronize with the reference server

- The program must work properly on the reference server (IP 10.10.5.120). In order to access this server from outside the teaching network, it is necessary to setup the VPN on your computer:  
<https://www.udc.es/rede/vpn.html>
- It is possible to configure CLion to enable the option for synchronizing the source code with the reference server. This makes subsequent testing in it easier.
- To do this, you need to setup the option *Deployment* (*Tools->Deployment->Configuration*)
  - Setup the tab *Connection*.
  - Setup the synchronization folder at the tab *Mappings*.
- Once configured, option *Tools->Deployment->Upload to...* will upload the source files to the server.
- *The following slides show this process.*



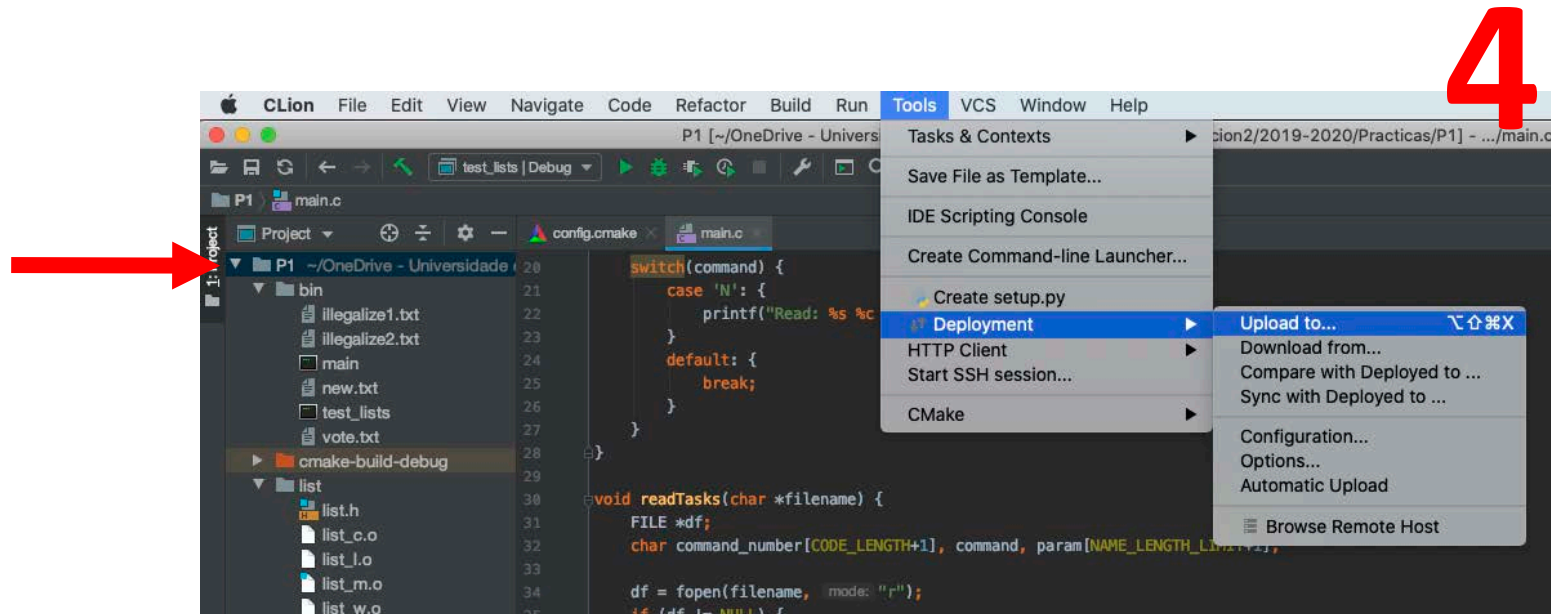
# Configuring CLion to synchronize with the reference server



# Configuring CLion to synchronize with the reference server

- Every time the menu option *Tools -> Deployment -> Upload to...* is run, changes will be synchronized.
  - Don't forget to **select the project folder** in the CLion file view. Only this way **all files** will be uploaded.
- Another option consists in using another program (e.g. **Filezilla** ) to upload the files to the server.

Select the  
project folder



# Remote SSH session to the reference server

- It is possible to open an SSH connection to the reference server from CLion itself by using the menu option:  
*Tools -> Start SSH Session...*
- At the bottom of the IDE you will see a **Terminal** from which we can run the test **script**, for example.
- This option is not available in some versions of CLion.
- Again, another option consists in using another program (e.g. **FPuTTY** ) to open a remote session to the reference server.

