Concurrent Programming

Week 1 – Home Work

Programming environment

During the semester the compiler that is going to be used is **gcc**. It compiles C/C++ source code into executables.

Students

Although it can be executed from the command line it is advantageous to use an IDE (Integrated Development Environment).

The IDE seamless integrates the writing of the code the compilation, error finding, debugging and execution all inside a single environment.

The IDE that is suggested is the Microsoft Visual Code (VSCode).

VSCode has versions to Linux, MAC OS X and Windows.

In the case of Linux and Mac OS X, VSCode uses the installed compilers.

In Windows it can use the Windows Subsystem for Linux (that replicates a Linux OS inside Windows).

The installation of VSCode is straightforward and its configuration to compile C/C++ code is also simple.

It is only necessary to follow the documentations provided by Microsoft.

In the middle of this document there is a small exercise for you to solve.

1 Install Visual Studio Code

The way to install Visual Studio Code is described in

- https://code.visualstudio.com/docs/setup/setup-overview

This simple tutorial guides the installation on the three different operating systems: Linux, MAC OS X and windows

After installation, students should follow the GET STARTED guided tour to get acquainted with VSCode:

- https://code.visualstudio.com/docs/getstarted/introvideos

2 Install C++ Extensions

In order to compile C programs it is necessary to install the C++ extension.

2.1 Linux

If the student is using Linux, it is necessary to install the gcc compiler and the necessary libraries. In Ubuntu this is done with the command:

```
sudo apt-get install build-essential gdb
```

in other Linux distributions the command can be different :/

2.2 Windows

In Windows it is necessary to install:

- Windows Subsystem for Linux
- install and configure one Linux distribution inside the WSL (for instance Ubuntu)
- install the **Remote-WSL** extension inside VSCode

2.3 MAC OS X

In MAC OS X it is necessary to guarantee that you have the Xcode installed

2.4 Instructions

All the necessary installations are defined in on of these guides:

- https://code.visualstudio.com/docs/cpp/config-linux
- https://code.visualstudio.com/docs/cpp/config-wsl
- https://code.visualstudio.com/docs/cpp/config-clang-mac

After installing all the dependencies guarantee that everything is running smoothly by executing the example presented in the previous guides.

This example is in C++, but if you can run it, you can run C programs also.

3 Debug C programs

https://code.visualstudio.com/docs/cpp/cpp-debug

The VS Code allows debugging of C program. It allows the regular execution of applications with the added functionalities:

- If the application crashes it is possible to evaluate the state of the application
- it is possible to insert breakpoints
- it is possible to execute the application step by step
- it is possible to verify and change the values of variables
- it is possible to execute python expressions that will change the state of the applications

Don’t forget to follow the debugging guide:

- https://code.visualstudio.com/docs/cpp/cpp-debug

4 Exercise 1

In this exercise students will use VS Code to edit, compile and run a program.

Follow the next steps:

- Create a folder and inside it edit the lab1.c file
  - you can create a file called **lab1.c** and copy to it the C code
  - copy the **lab1.c** code into the suitable directory
- Compile the program
- Run the program
  - Type your student number
  - Take note of the printed value

Why does the random function always return the same number?

Is **random()** really random?

Debug the program line by line to understand if everything is ok!

5 Collaborate while programming

https://docs.microsoft.com/en-us/visualstudio/liveshare/use/vscode

https://docs.microsoft.com/en-us/visualstudio/liveshare/quickstart/share

https://docs.microsoft.com/en-us/visualstudio/liveshare/quickstart/join

https://docs.microsoft.com/en-us/visualstudio/liveshare/quickstart/browser-join

The Visual Studio Code offers a set of tools that allow multiple user to interact in the same workspace and files.

This allows remote users (in different computers) to control the same workspace:

- All users see the same interface
- All users can change in real time the same file.
- All users can execute the applications and interact with them

To activate this functionality it is necessary to install the Visual studio live share.

Follow the instructions in:

- https://docs.microsoft.com/en-us/visualstudio/liveshare/use/vscode

You will be required to sign in. You can use your TEAMS/Técnico account to do so.

After sharing a session, the remote partner can access your work from his installed VSCode or even from the browser:

- https://docs.microsoft.com/en-us/visualstudio/liveshare/quickstart/join
- https://docs.microsoft.com/en-us/visualstudio/liveshare/quickstart/browser-join

Now all participants can cooperate on the same workspace in real time.

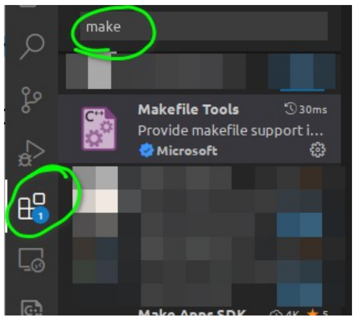
If necessary, an audio connection can be established to help partners to communicate.

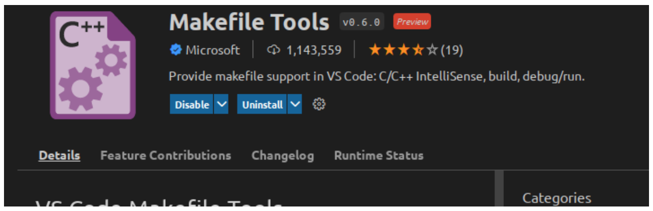
6 Use of makefiles

https://github.com/microsoft/vscode-makefile-tools

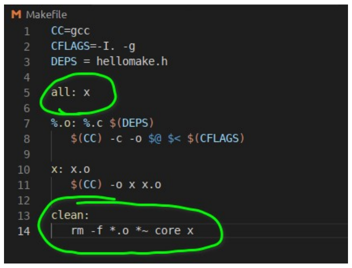
The Visual Studio Code can use a standard Makefile to compile a project. To accomplish this, it is necessary to create a Makefile with certain simple structure and install an extension.

Students should install the **Makefile Tools** extension from Microsoft:





After this extension is installed, students should create a Makefile file with the following required rules (as illustrated in the example):



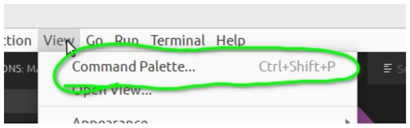
```
Makefile
CC=gcc
CFLAGS=-I. -g
DEPS = hellomake.h

all: x

%.o: %.c $(DEPS)
	$(CC) -c -o $@ $< $(CFLAGS)

x: x.o
	$(CC) -o x x.o

clean:
	rm -f *.o *~ core x
```

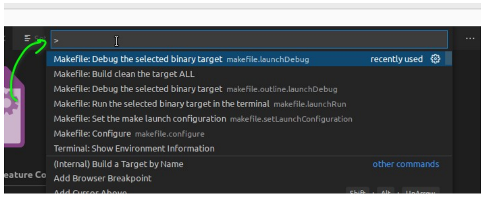
- all
- clean

All other compilation rules can follow any common pattern.

To be able to debug the compiled programs it is necessary to include the **-g** compilation flag.

To compile, and run the programs from the Makefile it is necessary to issue Vscode commands through the **Command Pallet**:

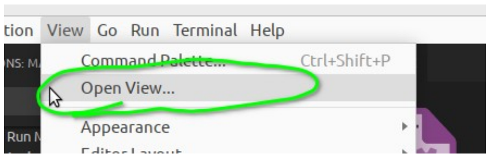


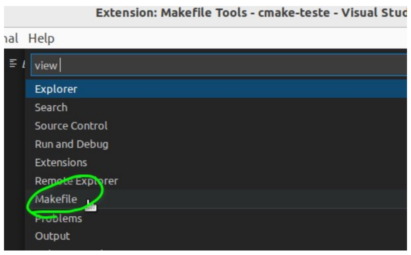

>
Makefile: Debug the selected binary target makefile.launchDebug
recently used
Makefile: Build clean the target ALL
Makefile: Debug the selected binary target makefile.outline.launchDebug
Makefile: Run the selected binary target in the terminal makefile.launchRun
Makefile: Set the make launch configuration makefile.setLaunchConfiguration
Makefile: Configure makefile.configure
Terminal: Show Environment Information
(Internal) Build a Target by Name
other commands
Add Browser Breakpoint

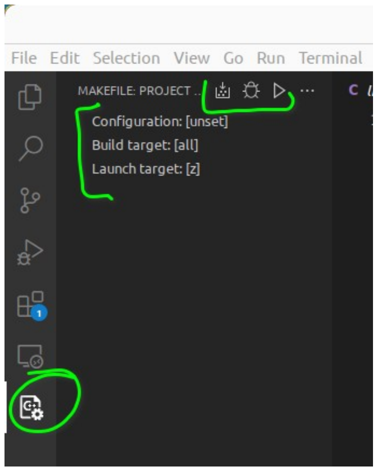
- **Makefile: Build the current target** – equivalent to **make all**

- Makefile: **Build clean the target All** – equivalent to **make clean all**
- **Makefile: Set the make launch configuration** – defines the application to be executed or debugged
- **Makefile: Debug the selected binary target** – debugs the selected applications
- **Makefile: Run the selected binary target in the terminal** – runs the selected application

It is possible to execute these steps using the mouse if the **Makefile view** is activated:


tion View Go Run Terminal Help
Command Palette... Ctrl+Shift+P
Open View...
Appearance
Editor Layout





Configuration: [unset]
Build target: [all]
Launch target: [z]

7 Exercise 2

Create a Makefile for the program of Exercise 1 and verify the various commands to compile, execute and debug inside VSCode.