ATAD | Software Requirements

The installation of all the necessary software is presented, according to your operating system, in the following sections:

1. **Windows/WSL** or native **Linux**.
2. **MacOSX**.

> If you are not able to perform a manual software installation, please see Software_AlternativeMethods.pdf

Software Listing

The complete list of software is as follows:

- Windows Subsystem for Linux (Ubuntu LTS 20.04) – **windows-only**
- GNU Compiler Collection (GCC)
- GNU Project Debugger (GDB)
- GNU Make
- Git (already installed in Windows/WSL and Linux)
- Valgrind
- Doxygen
- Visual Studio Code (IDE):
  - (Extension) C/C++
  - (Extension) Doxygen Documentation Generator

1 | Windows/WSL or Linux

Go to step 6 if you're on a native Linux environment. The commands provided are for *Ubuntu/Debian* systems, but you should easily find the required packages for other distributions and transpose the commands to your package manager.

Installing

Weblink: Install Windows Subsystem for Linux (WSL) on Windows 10 | Microsoft Docs

1. Open PowerShell as Administrator, run the following command and restart.

```
PS> Enable-WindowsOptionalFeature -Online -FeatureName
 ↪  Microsoft-Windows-Subsystem-Linux
```

2. Open **Microsoft Store** and search for "Ubuntu". Install **Ubuntu LTS 20.04 LTS**.
3. Run *Ubuntu* application and wait for installation.
4. During the installation you'll be asked for a **default UNIX user account**. Enter one (no spaces allowed and I recommend all lowercase letters) and your **password** (do not forget this password!)
5. The CLI will be presented afterwards.

Updating Ubuntu image

Weblink: Initialize a new WSL Linux distro | Microsoft Docs

6. Update installed packages with:

```
$> sudo apt update && sudo apt upgrade
```

… this may take a few minutes. Drink some coffee.

Installing compiler, debugger, memory checker and documentation generator

7. Run the following commands:

```
$> sudo apt install build-essential
$> sudo apt install gdb
$> sudo apt install valgrind
$> sudo apt install doxygen
```

8. Make sure *git* is already installed (if it is, there is no need for reinstall):

```
$> sudo apt install git
$> git --version
```

Installing Visual Studio Code

9. Download and install **System installer 64bit** from Download Visual Studio Code - Mac, Linux, Windows.

- For windows users, it may be helpful to check:
  - Get Started with C++ and Windows Subsystem for Linux in Visual Studio Code

VS Code Extensions

10. Install the following extensions from VS Code Marketplace:

- Name: **C/C++**
  Id: ms-vscode.cpptools
  Description: C/C++ IntelliSense, debugging, and code browsing.
  Publisher: Microsoft
  VS Marketplace Link
- Name: **Doxygen Documentation Generator**
  Id: cschlosser.doxdocgen
  Description: Let me generate Doxygen documentation from your source code for you.
  Publisher: Christoph Schlosser
  VS Marketplace Link

Test the environment

11. Clone the `CProgram_Template` course repository and test the toolchain (next section).

```
$> git clone https://github.com/estsetubal-atad/CProgram_Template.git
 ↪  FirstProgram
```

12. Open the project with *VS Code*:

```
$> code FirstProgram/
```

- **Note:** If you're **not using the *docker container*** methodology, when asked if you would like to "reopen folder to develop in a container", **choose "Don't Show Again"**. Otherwise, use that development option.

13. Within the editor you can make changes to the source files.

Compiling (`make` and `gcc`)

14. Open the *integrated terminal*: Menu **Terminal > New Terminal**.

Invoke the makefile:

```
$> make
```

and run the program:

```
$> ./prog
```

Debugging

15. Edit the source file `main.c` and put the following code inside the *main* function:

```c
int main() {

    char str[30] = "Debugging in VS Code";

    int i = 0; //Line 9. Place breakpoint here.
    while(str[i] != '\0') {
        printf("%c\n", str[i]);

        i++;
    }
    printf("Done!");

    return EXIT_SUCCESS;
}
```

- Place a *breakpoint* in the line that contains the `while` instruction. You should get a red dot at that position.

- In the **Run and Debug** tab (leftmost side, with a "lady beetle") you should now see a green play icon ▶ at the top beside "gdb - Debug project". Click on it and the debug will start.
  - This will call `make debug` automatically and run `gdb` over the `prog` executable.
- In the **Variables** panel you can see the current values of `str` (all the positions of the array) and `i`. The variable `i` is not yet initialized, because this instruction at line `9` hasn't been executed yet!
- Add the expression "`str[i]`" to the **Watch** list, before continuing;
- Now use the **Step Over (F10)** command to proceed line by line, watching the values change as the program executes.

---

2 | MacOSX

WARNING: Please consider using **Docker container** method described in Software_AlternativeMethods.pdf, as it will be easier and you'll have the complete toolchain, guarantee.

If you're still aiming to manually install the software on your Mac computer, we'll be using **CLang** and **LLDB** which are themselves *forks* of GCC and GDB, but supported natively by Apple. Consequently, follow the next instructions.

Install CLang

`Clang` may already be installed on your Mac. To verify that it is, open a macOS Terminal window and enter the following command:

```
$> clang --version
```

If `Clang` isn't installed, enter the following command to install the command line developer tools:

```
$> xcode-select --install
```

> `git` is already included with XCode.

Turn off *security prompt* for debugging purposes:

```
$> sudo DevToolsSecurity --enable
```

Install Doxygen

```
$> brew install doxygen
```

This should install fine, since it is supported across all major MacOSX versions.

Install Valgrind

WARNING: *Valgrind* isn't supported for *Apple silicon*, i.e., M1 and later chips.

This is not a guaranteed installation. What has worked is the following:

```
$> brew tap LouisBrunner/valgrind
```

Followed by:

```
$> brew install --HEAD LouisBrunner/valgrind/valgrind
```

These steps are from the following *git repository*: https://github.com/LouisBrunner/valgrind-macos/

Install VS Code and extensions

Follow steps 9 and 10 from the Installing Visual Studio Code section process, e.g., VS Code and C/C++ and Doxygen extensions.

Additionally, install the **CodeLLDB** extension:

https://marketplace.visualstudio.com/items?itemName=vadimcn.vscode-lldb

Test the environment in MacOSX

Follow steps 11 to 15 from the Installing Visual Studio Code section process, but clone the following repository instead:

```
$> git clone https://github.com/estsetubal-atad/CProgram_Template_MacOS.git
 ↪  FirstProgram
```

> This is a different repository that contains separate `makefile` and `launch.json` files for MacOS environments and the previous tools.

Also, when *debugging* you should see a green play icon ▶ at the top beside "lldb - Debug project".

Troubleshooting

WSL

Most of the times the problems arise from not enabling *Virtualization* on your BIOS settings.

References:

- https://docs.microsoft.com/en-us/windows/wsl/troubleshooting

Cannot manually install?

Check Software_AlternativeMethods.pdf for alternative methods.

Author and support

Bruno Silva (bruno.silva@estsetubal.ips.pt)

You should ask your PL teacher for any help regarding these contents and procedures.