1 ATAD | Software Requirements

You may run the necessary software through two main methods:

1. Manual software installation **(preferred, if possible)**, or.
   - Either using Windows/WSL or native Linux.
2. Premade Virtual Box Appliance;

> If you are not able to perform a manual software installation, the recommended *fallback* is through a virtual machine.

If you are using **MacOS** there are some *tips* at the end of this document.

1.1 Manual software installation

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
- Valgrind
- Doxygen
- Visual Studio Code (IDE):
  - (Extension) C/C++
  - (Extension) Doxygen Documentation Generator

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.

1.1.1 Installing

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

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

```
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.

1.1.2 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.

1.1.3 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
```

1.1.4 Installing Visual Studio Code

8. 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

1.1.5 VS Code Extensions

9. 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

1.1.6 Test the environment

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

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

11. Open the project with *VS Code*:

```
$> code FirstProgram/
```

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

Compiling (`make` and `gcc`)

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

Invoke the makefile:

```
$> make
```

and run the program:

```
$> ./prog
```

Debugging

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

```
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 **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.

1.2 Virtual Box Appliances

There are two Linux appliances that you can import at:

ATAD Linux Appliances - OneDrive

One is based on Manjaro XFCE (more pretty, but needs more TLC, since its a rolling distribution); the other based on XUbuntu 19.10 (Ubuntu with XFCE).

Both we'll need 10Gb of disk space when imported; this includes 2,5Gb of free disk space. The *default* user configuration is:

- **Name:** ATAD
- **Username:** malloc
- **Password (sudoer):** realloc

1.3 MacOS Support

I have no access to an *Apple* machine to test things out. Here are some links to help you out:

- https://stackoverflow.com/questions/47051457/how-to-set-gnu-make-to-use-gcc-default-on-macos
- https://stackoverflow.com/questions/40650338/valgrind-on-macos-sierra

If you manage to get things working *natively* on MacOS, please contribute the tutorial.

1.4 Author and support

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

Please follow the course *guidelines* when seeking support.