



**Politecnico
di Torino**

Academic Year 2023/2024

COMPUTER ARCHITECTURE & OPERATING SYSTEMS

HackIOSsim Project

Step-by-Step Installation Guide

Group 20

Sannia Gabriele - S331385
Sabella Mattia Luigi - S285363
Pittalis Domenico - S283602
Nobili Luca - S331461

January 15, 2024

Contents

1	Introduction to the Guide	3
2	Visual Studio Code: Download & Install	4
3	Qemu: Download & Install	8
4	Arm GNU Toolchain: Download & Install	13
5	CMake: Download & Install	17
6	Git: Download & Install	21
7	Make: Download & Install	26
8	Add the Previously Installed Programs to the User Path Environment	27
9	Cloning the FreeRTOS Embedded OS from Git	30
10	Coding Environment Set-up for FreeRTOS	31
11	Run the FreeRTOS Demo Project with QEMU	33

1 Introduction to the Guide

The following is a step-by-step guide intended to help user install and get ready to start writing new programs for the **FreeRTOS** *embedded OS*, and to properly set it up to use the *instruction set architecture (ISA) emulator Qemu*. Please note that this guide is designed to help **only Windows** users.

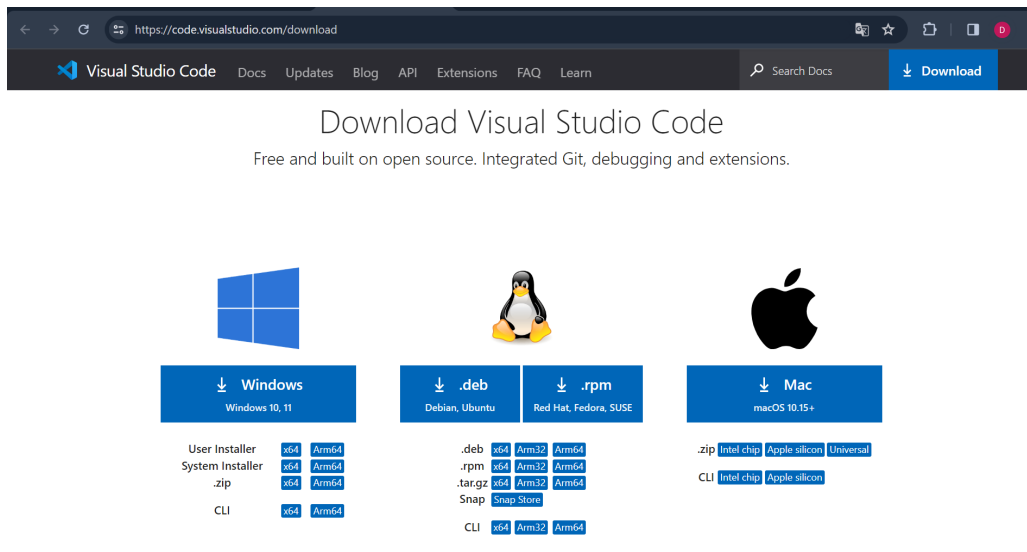
First let us briefly introduce the programs you are going to need to start working:

- **Visual Studio Code**;
- **MYSYS2**;
- **Qemu**;
- **Arm GNU Toolchain**;
- **Cmake**;
- **Git**;
- **Make** (installed with **Chocolatey**);
- **FreeRTOS**;

Beware, the download order of the programs **must not** be strictly followed as it is **not necessary**. However, it is strongly suggested (for time efficiency) to download everything that is needed **before** adding the various programs to the user's *PATH environment*.

2 Visual Studio Code: Download & Install

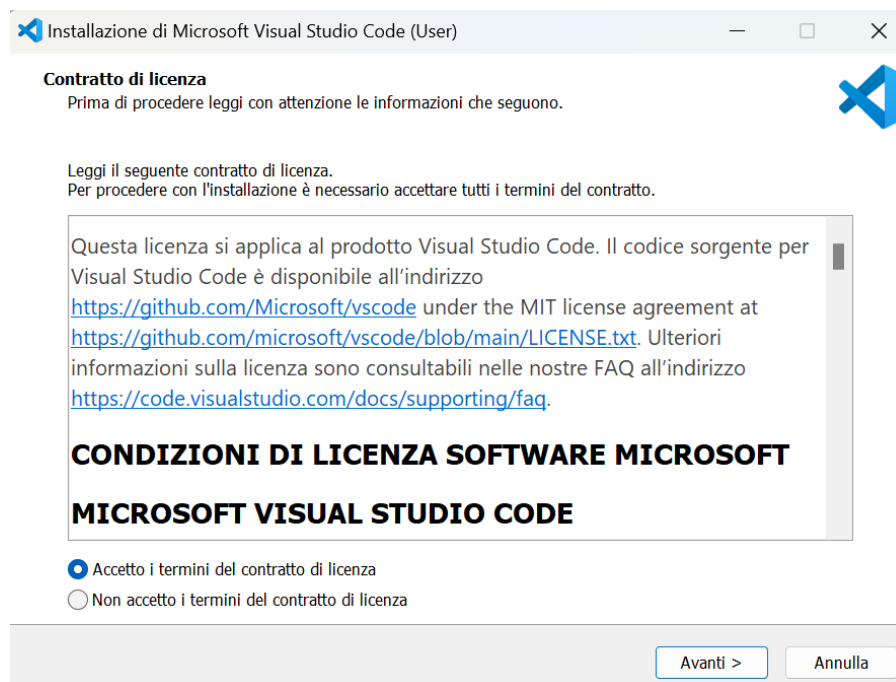
Go to the official website of [Visual Studio Code](https://code.visualstudio.com) in the [download section](#).



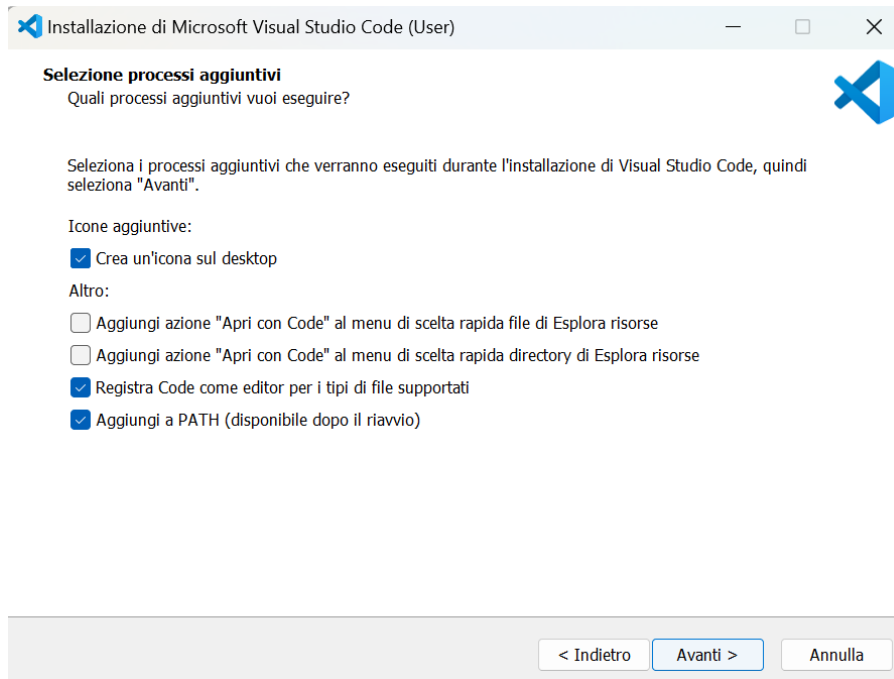
Click the Windows download button.

Once downloaded, run the executable and proceed with the various steps shown in the figures:

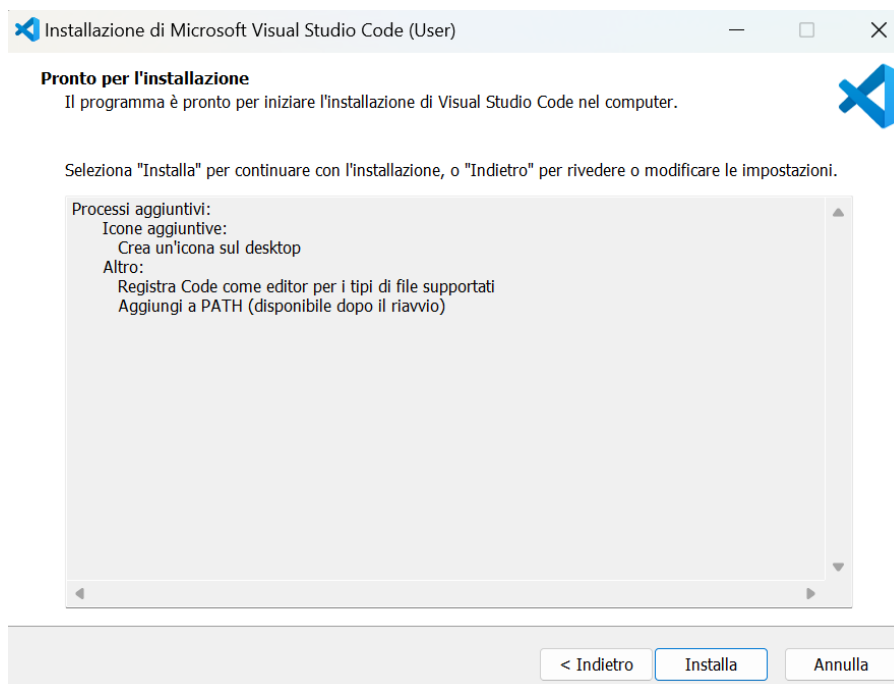
1. Accept the terms:



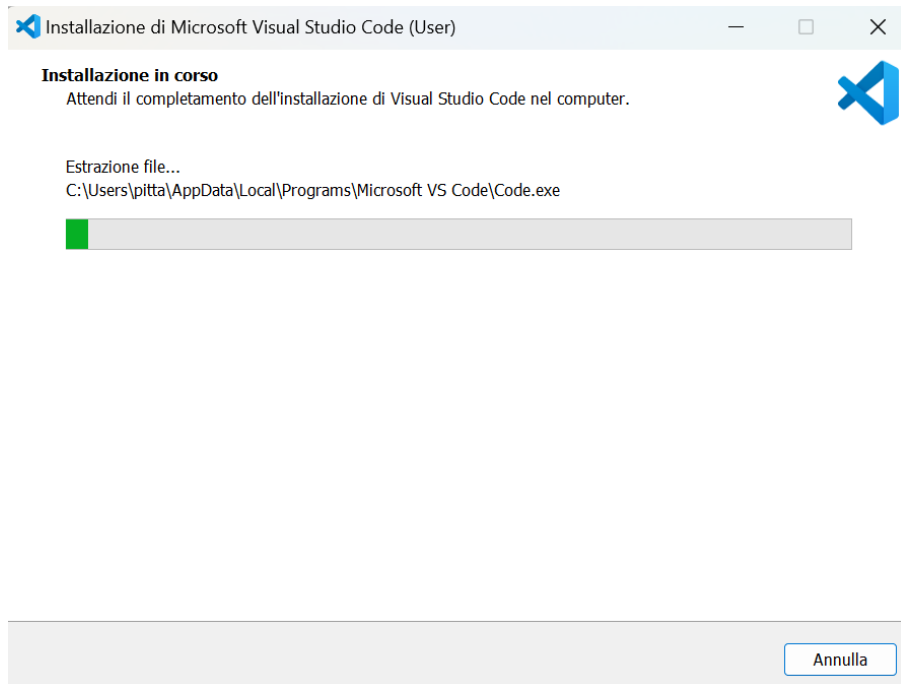
2. Tick the boxes as shown:



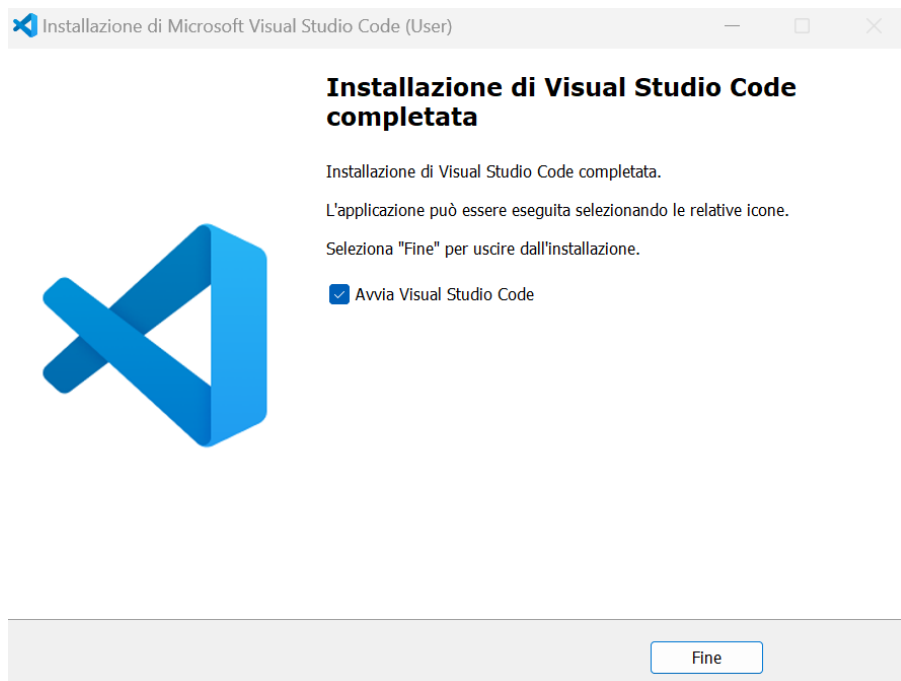
3. Click on "Install":



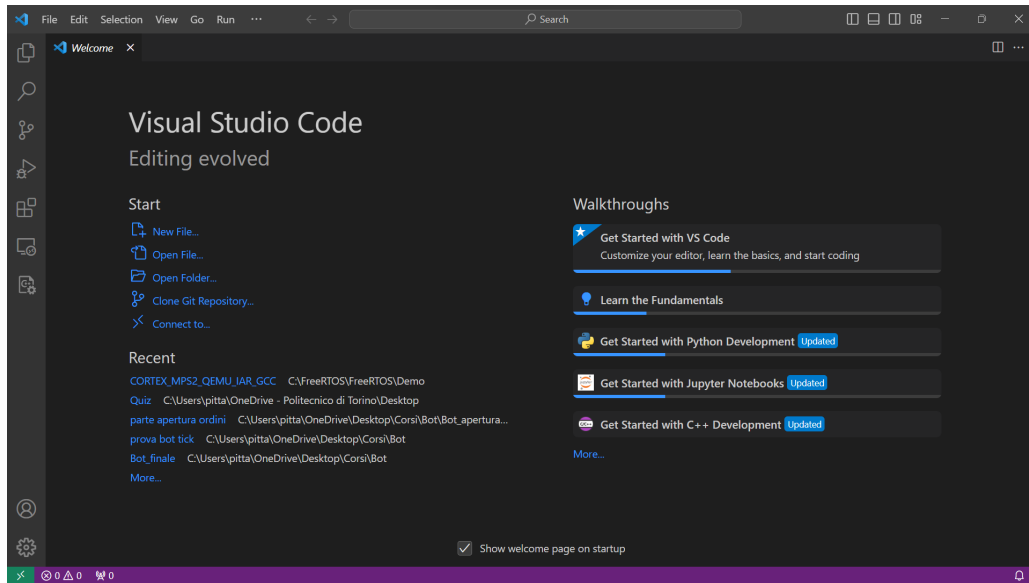
4. Wait for the installation to finish:



5. Try and open VSCode to see if it works properly by ticking the box "Run Visual Studio Code" and clicking on "Finish":

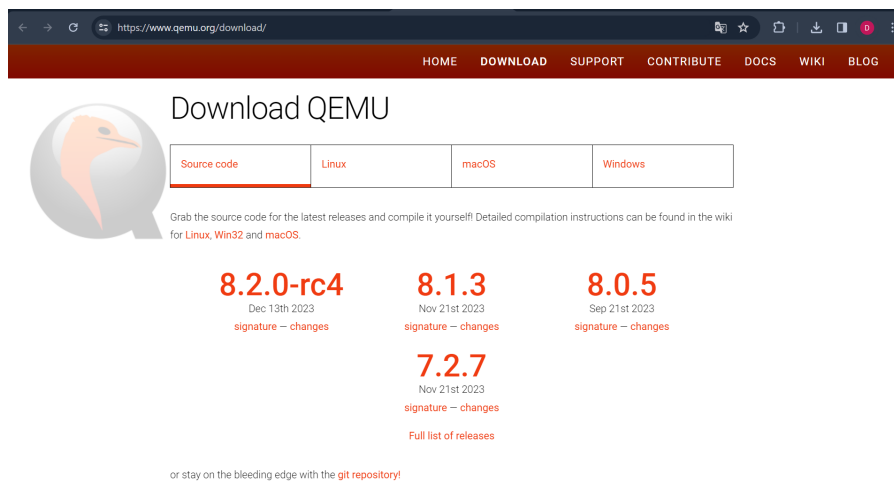


6. Verify that the starting page is similar to the following one:

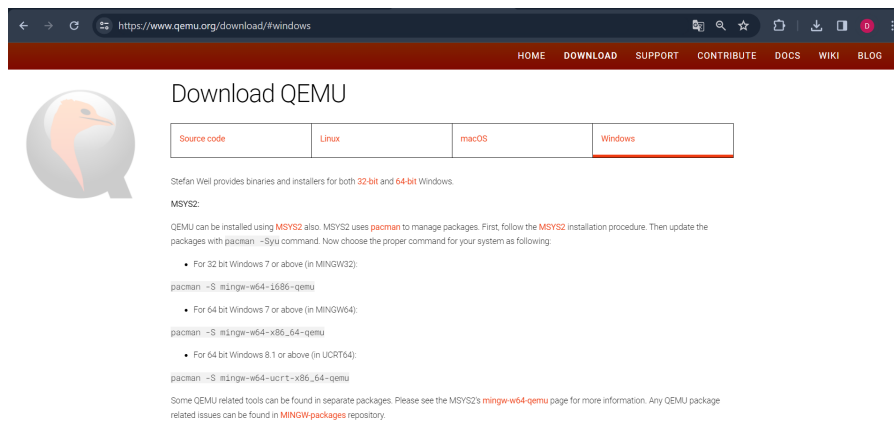


3 Qemu: Download & Install

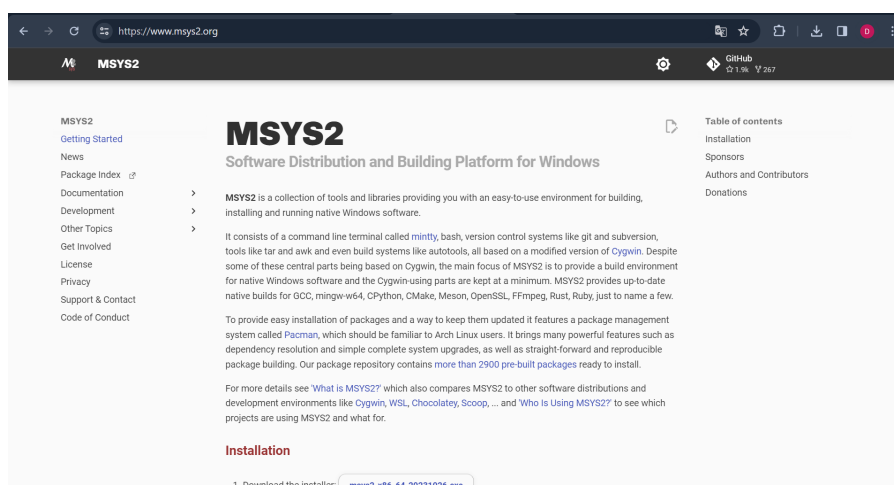
Go to the official website of [Qemu](https://www.qemu.org/) in the [download section](#).



Select the download button for Windows:



To download Qemu, you have to download first **MSYS2**. By clicking on the *red word* "MSYS2", you will be taken to the MSYS2 download page with its installation guide.



Follow the guide thoroughly, once *MYSYS2* is installed, open it and type the most suitable command for your system, choosing from those in shown in the Qemu website and execute it.

MSYS2:

QEMU can be installed using **MSYS2** also. MSYS2 uses **pacman** to manage packages. First, follow the **MSYS2** installation procedure. Then update the packages with **pacman -Syu** command. Now choose the proper command for your system as following:

- For 32 bit Windows 7 or above (in MINGW32):

```
pacman -S mingw-w64-i686-qemu
```

- For 64 bit Windows 7 or above (in MINGW64):

```
pacman -S mingw-w64-x86_64-qemu
```

- For 64 bit Windows 8.1 or above (in UCRT64):

```
pacman -S mingw-w64-ucrt-x86_64-qemu
```

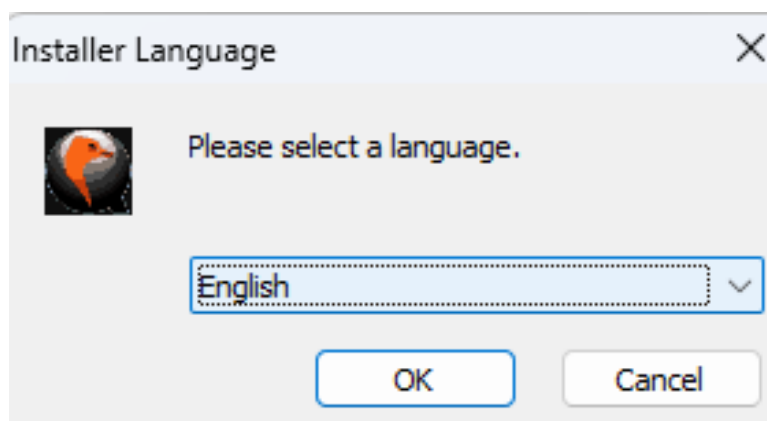
Some QEMU related tools can be found in separate packages. Please see the MSYS2's **mingw-w64-qemu** page for more information. Any QEMU package related issues can be found in **MINGW-packages** repository.

Now let us begin the actual download of Qemu. Open the link: <https://qemu.weilnetz.de/w64/2023/> and click on "*qemu-w64-setup-20230407.exe*" or the latest version to download the Qemu executable.

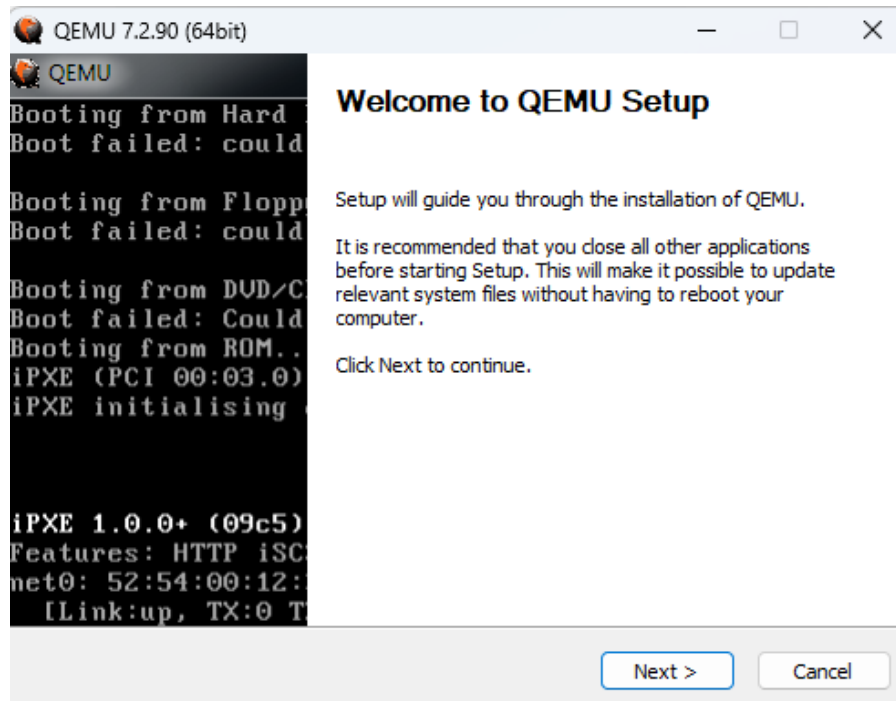


After opening the Qemu executable you just downloaded, the installation screen of the program will open, to avoid errors, follow the steps in the following figures:

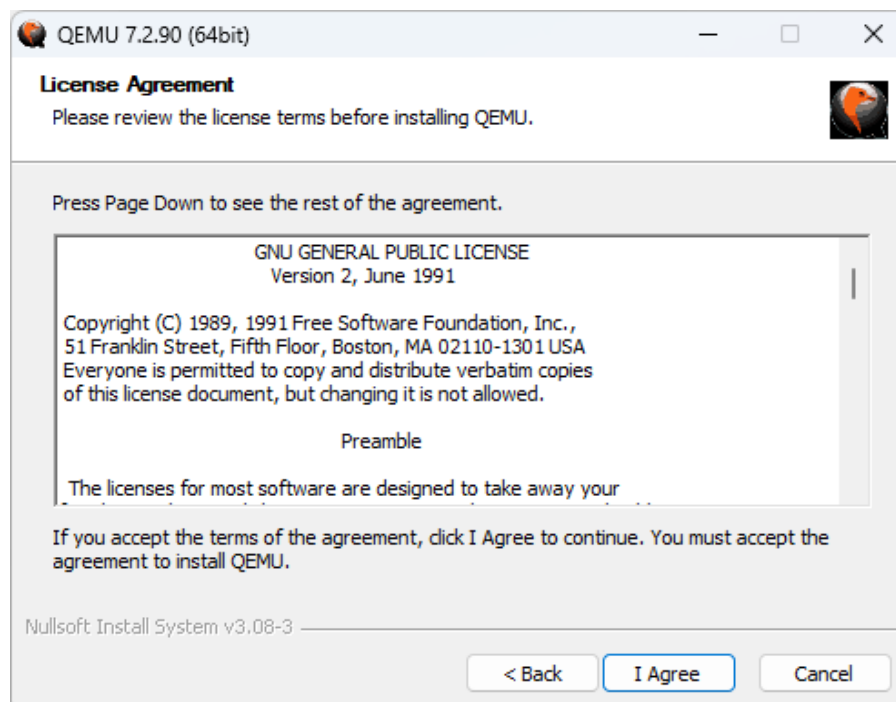
1. Select your preferred language:



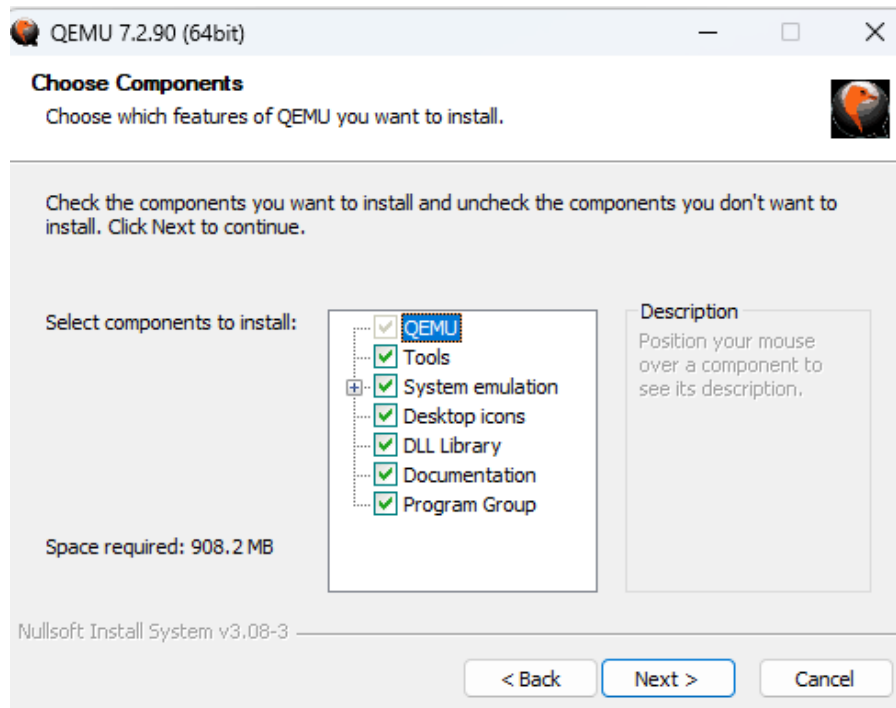
2. Click on "Next":



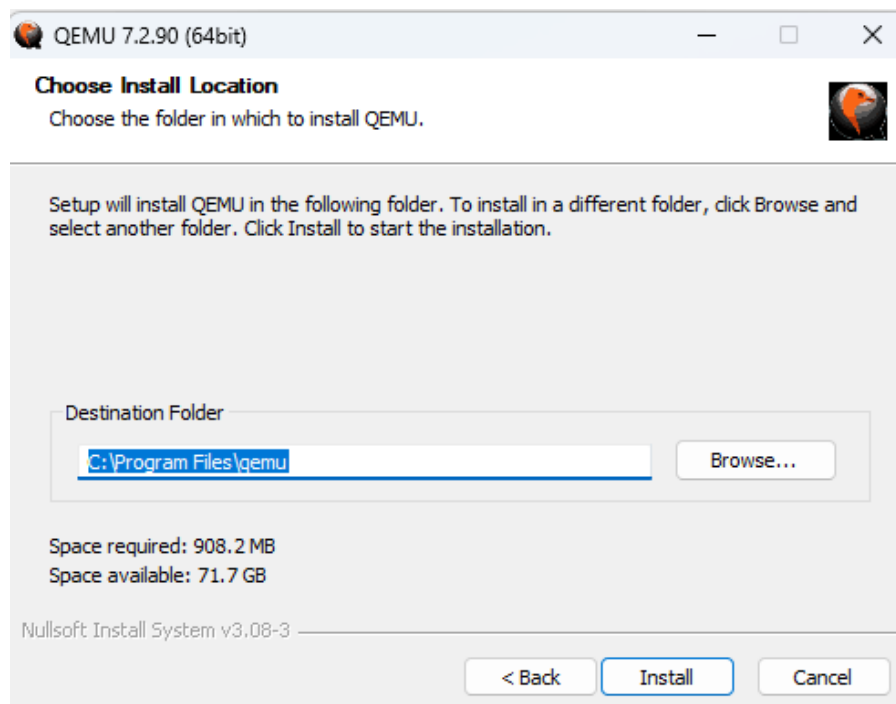
3. Click on "Agree":



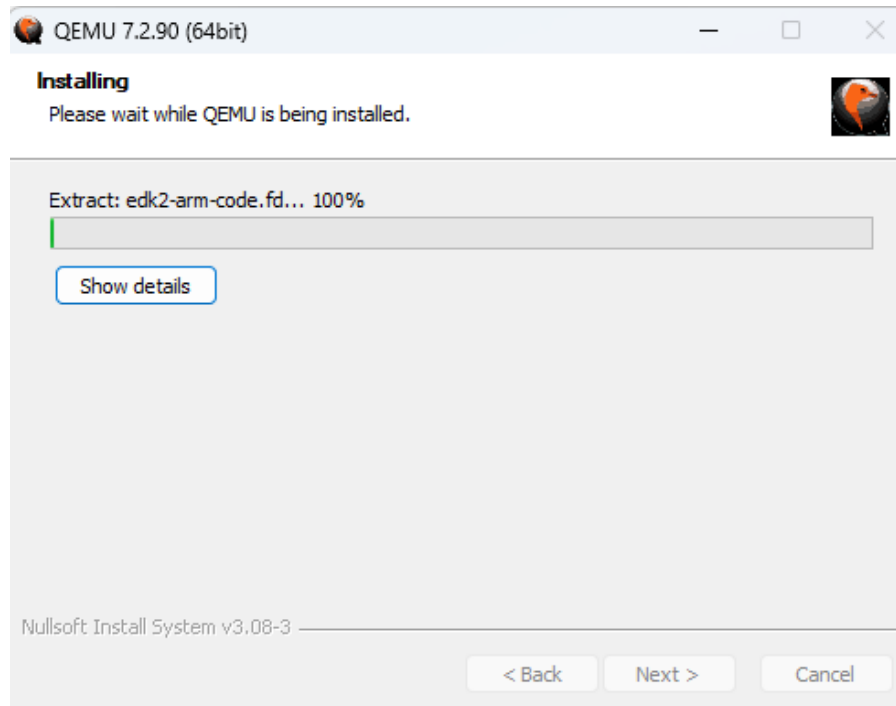
4. Tick **all** the boxes and then click "Next":



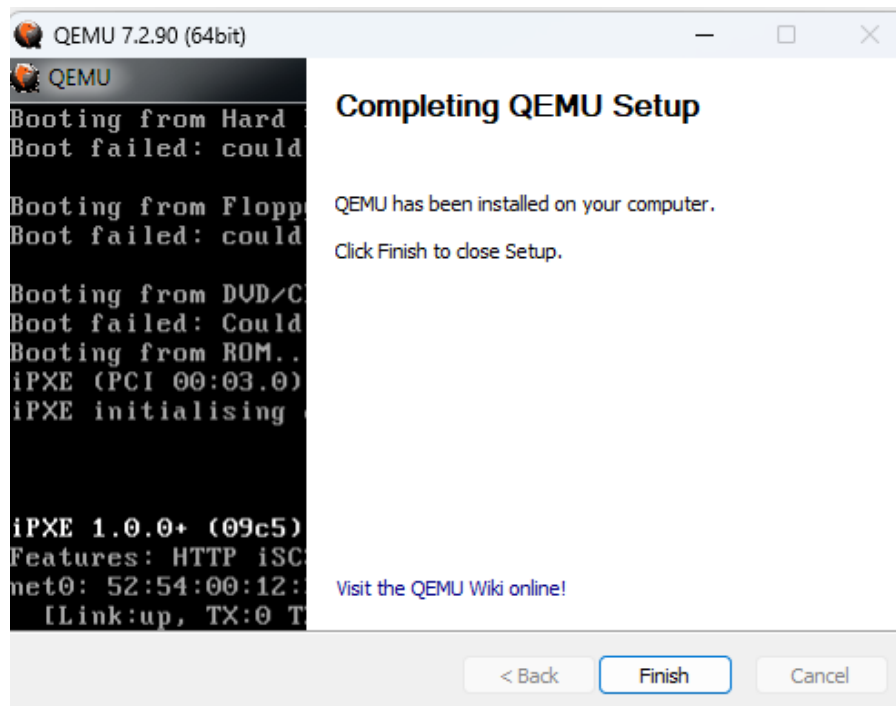
5. Chose an installation path and then click "Install":



6. Wait for the installation to finish:



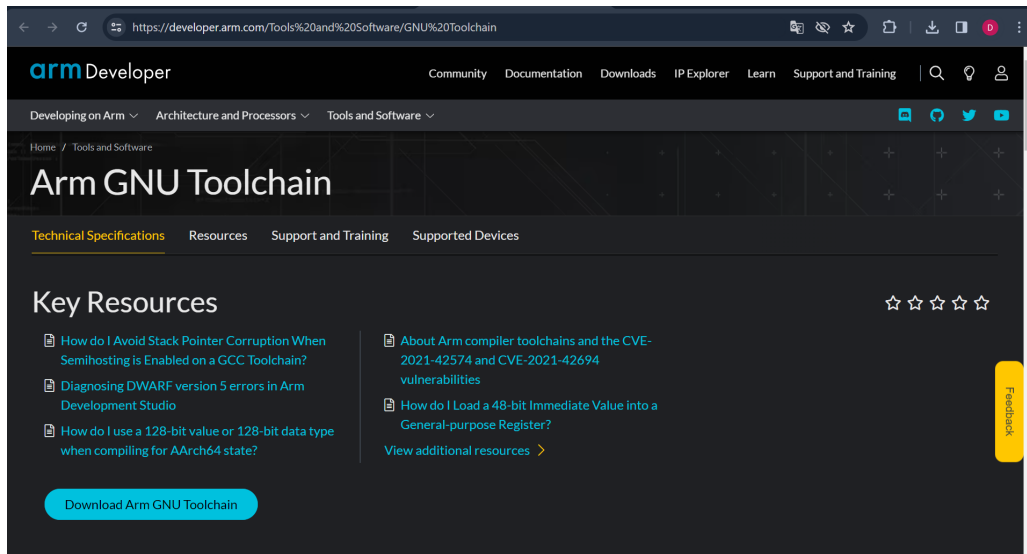
7. Click "Finish":



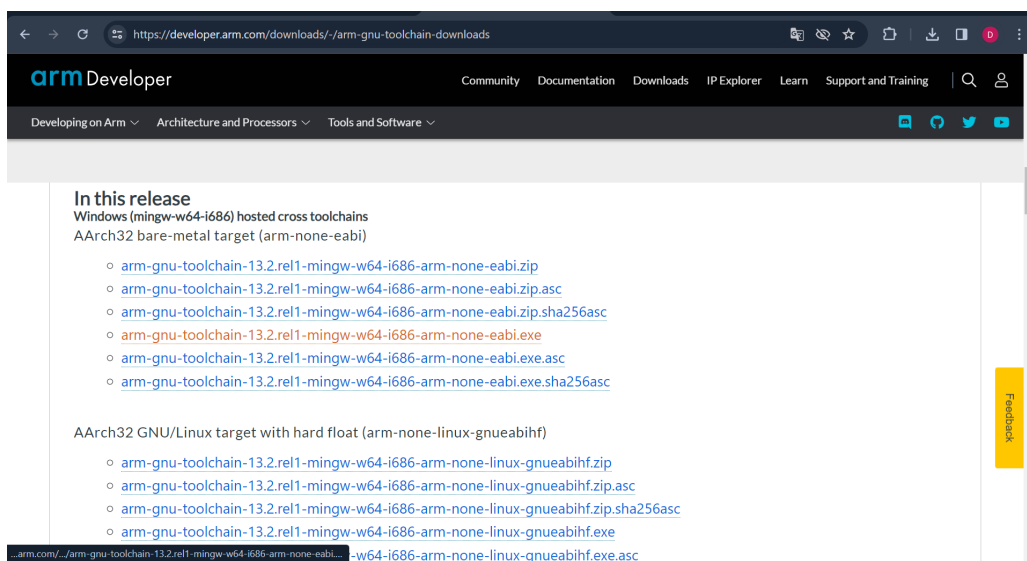
4 Arm GNU Toolchain: Download & Install

To download Arm GNU Toolchain open the follow link:

<https://developer.arm.com/Tools and Software/GNU Toolchain>



By clicking on the button "Download Arm GNU Toolchain" a new page will open. Scroll down until you find the section "Windows (mingw-w64-i686) hosted cross toolchain" and click on "arm-gnu-toolchain-13.2.rel1-mingw-w64-i686-arm-none-eabi.exe", immediately after the download will start.

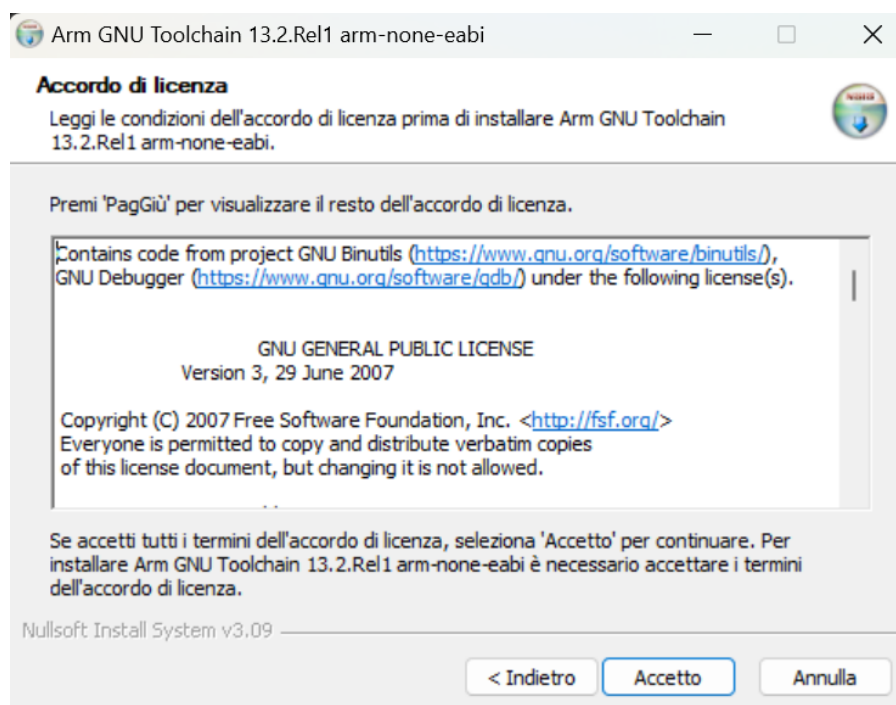


Now click on the installer and follow the procedure:

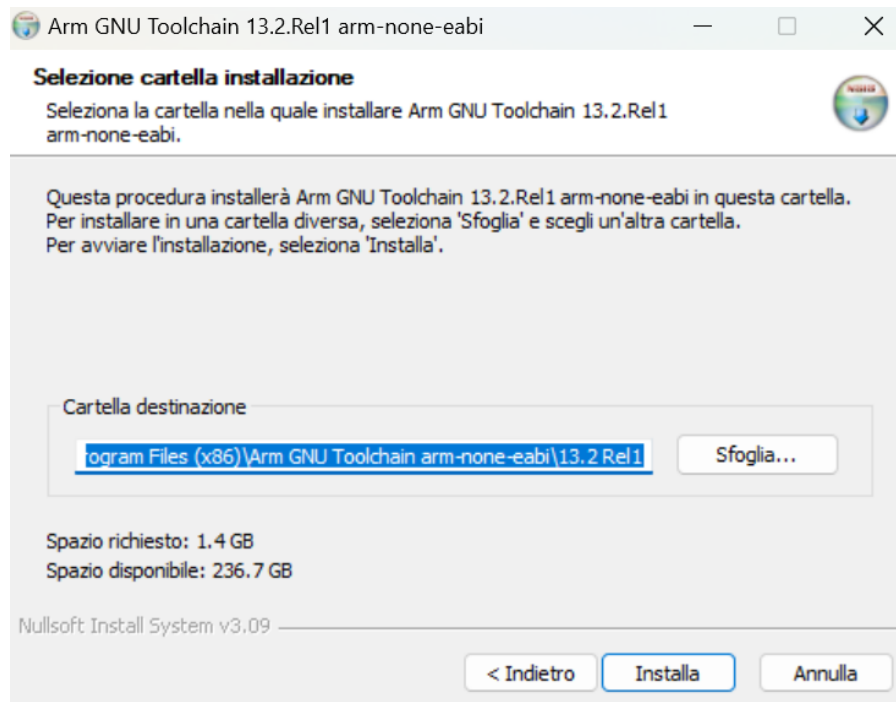
1. Click "Next":



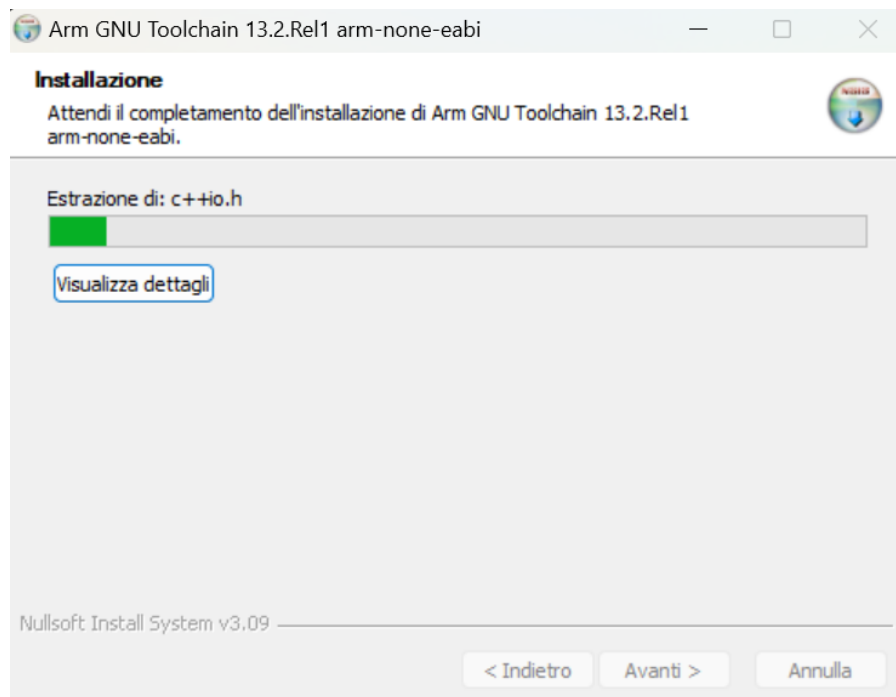
2. Click "Accept":



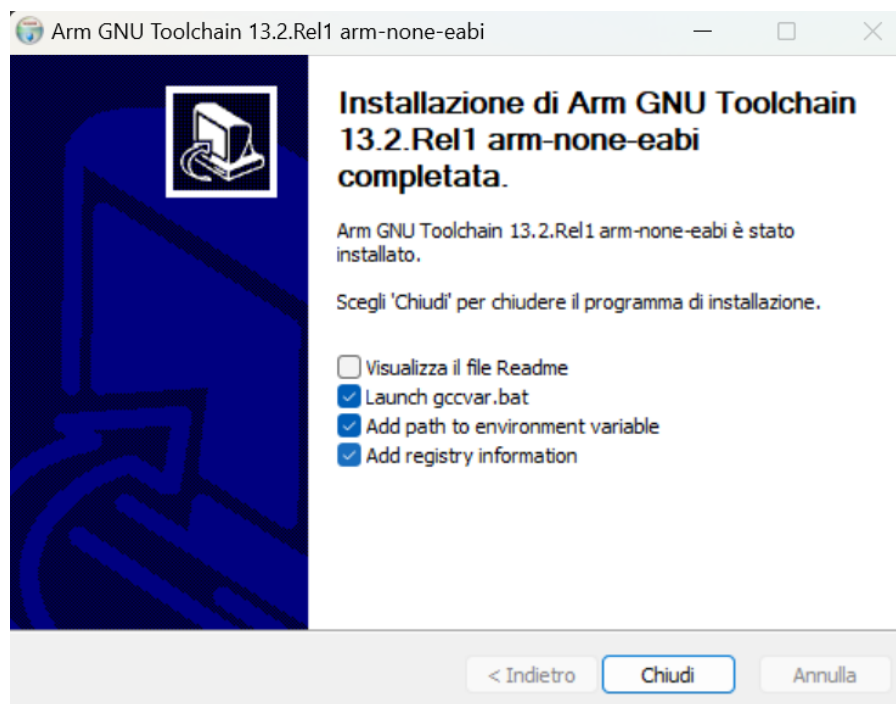
3. Chose an installation path and then click "Install":



4. Wait for the installation to finish:

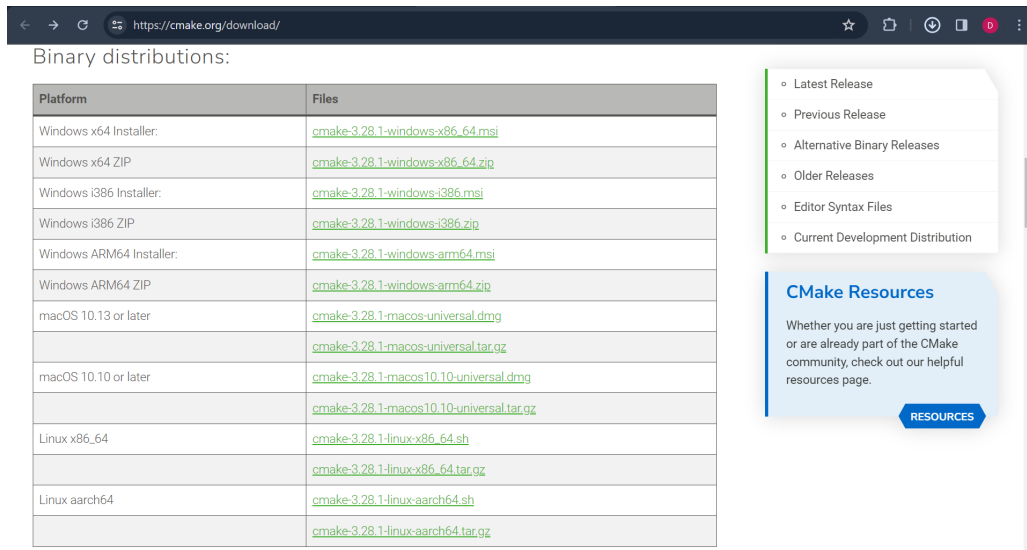


5. Tick the following boxes and then click "Close":



5 CMake: Download & Install

To download [CMake](#), go to the [download page](#) and scroll down the page to the section with *Binary Distributions*. Click on the most appropriate installer (**not** the ZIP) for your computer's processor.



Binary distributions:

Platform	Files
Windows x64 Installer:	cmake-3.28.1-windows-x86_64.msi
Windows x64 ZIP	cmake-3.28.1-windows-x86_64.zip
Windows i386 Installer:	cmake-3.28.1-windows-i386.msi
Windows i386 ZIP	cmake-3.28.1-windows-i386.zip
Windows ARM64 Installer:	cmake-3.28.1-windows-arm64.msi
Windows ARM64 ZIP	cmake-3.28.1-windows-arm64.zip
macOS 10.13 or later	cmake-3.28.1-macos-universal.dmg cmake-3.28.1-macos-universal.tar.gz
macOS 10.10 or later	cmake-3.28.1-macos10.10-universal.dmg cmake-3.28.1-macos10.10-universal.tar.gz
Linux x86_64	cmake-3.28.1-linux-x86_64.sh cmake-3.28.1-linux-x86_64.tar.gz
Linux aarch64	cmake-3.28.1-linux-aarch64.sh cmake-3.28.1-linux-aarch64.tar.gz

- Latest Release
- Previous Release
- Alternative Binary Releases
- Older Releases
- Editor Syntax Files
- Current Development Distribution

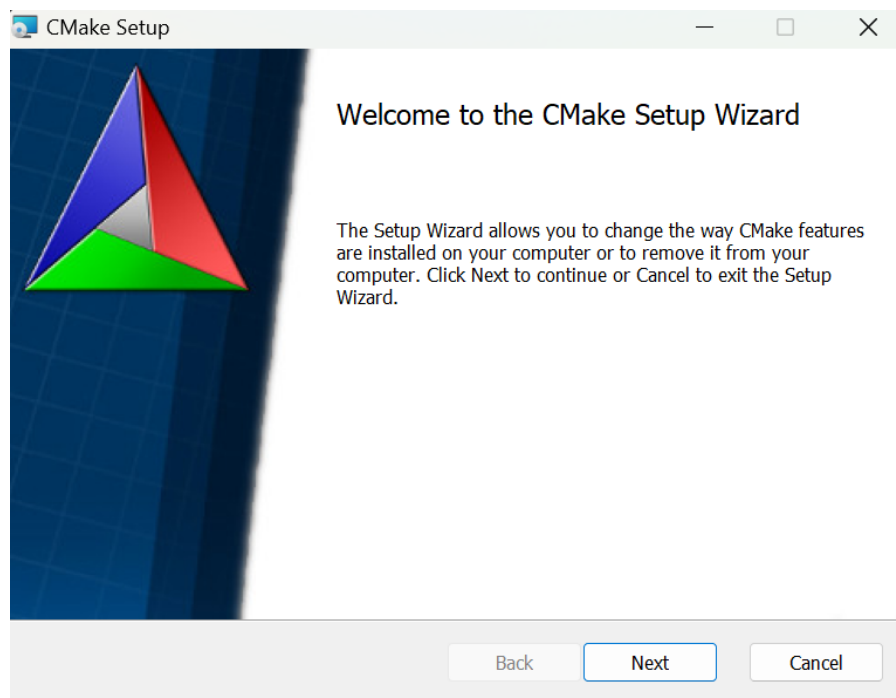
CMake Resources

Whether you are just getting started or are already part of the CMake community, check out our helpful resources page.

[RESOURCES](#)

Once the download is finished, click on the installer and follow the steps below to properly configure it:

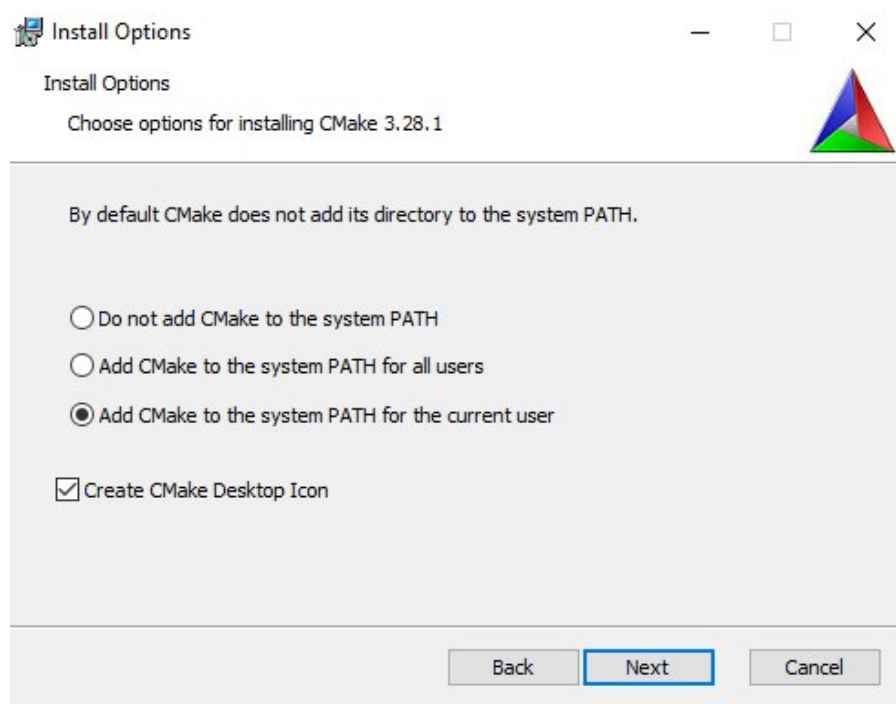
1. Click "Next":



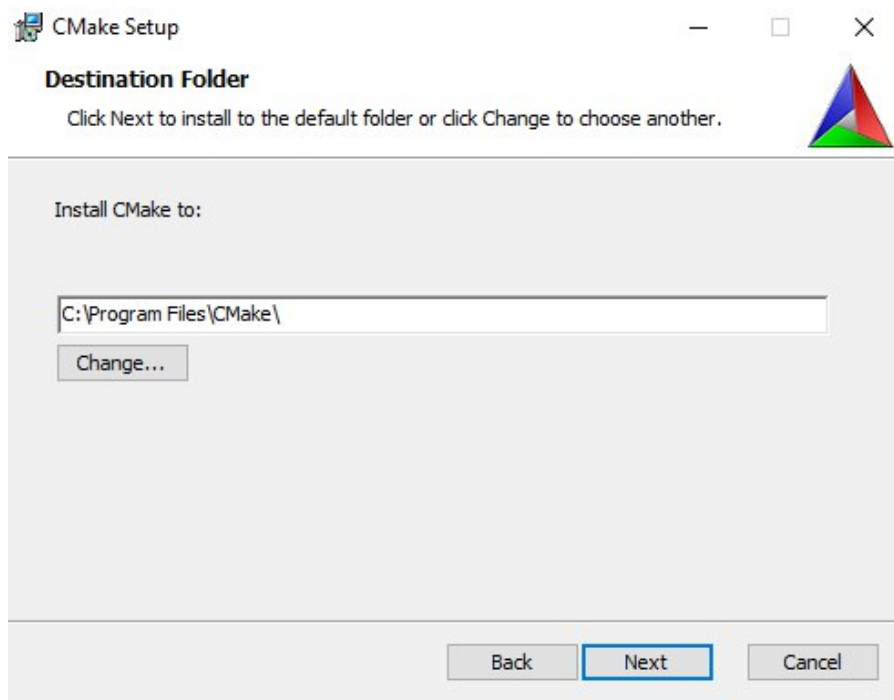
2. Tick the box and click "Next":



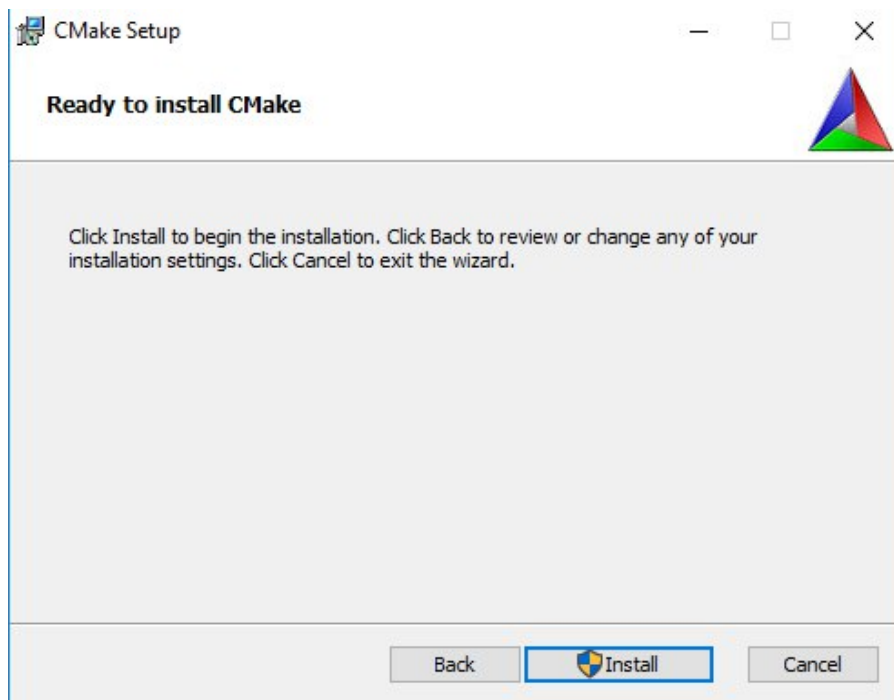
3. Select "Add CMake to the system PATH for the current user", then click "Next":



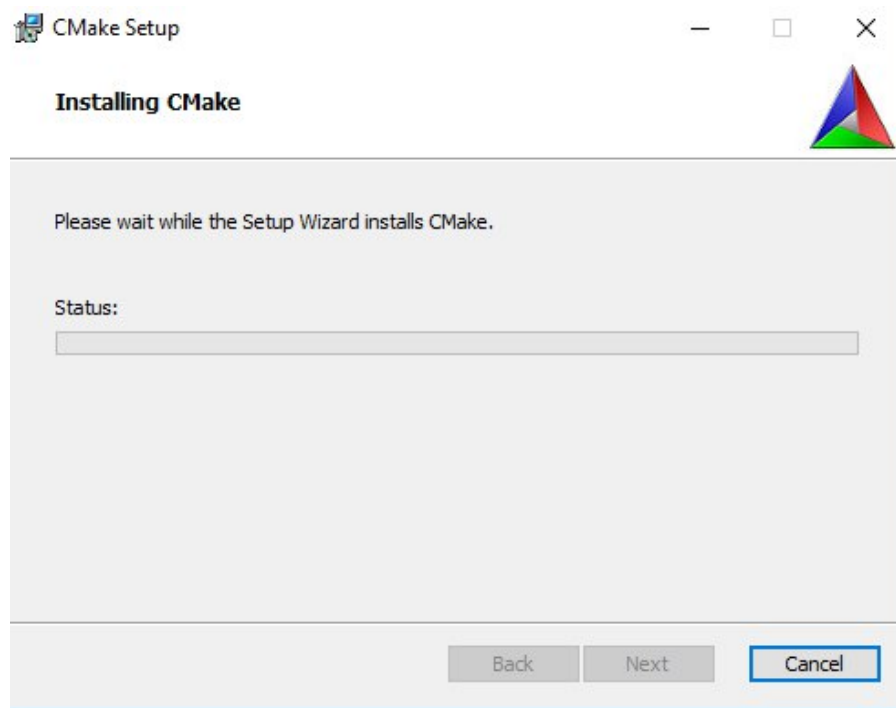
4. Chose an installation path and then click "Next":



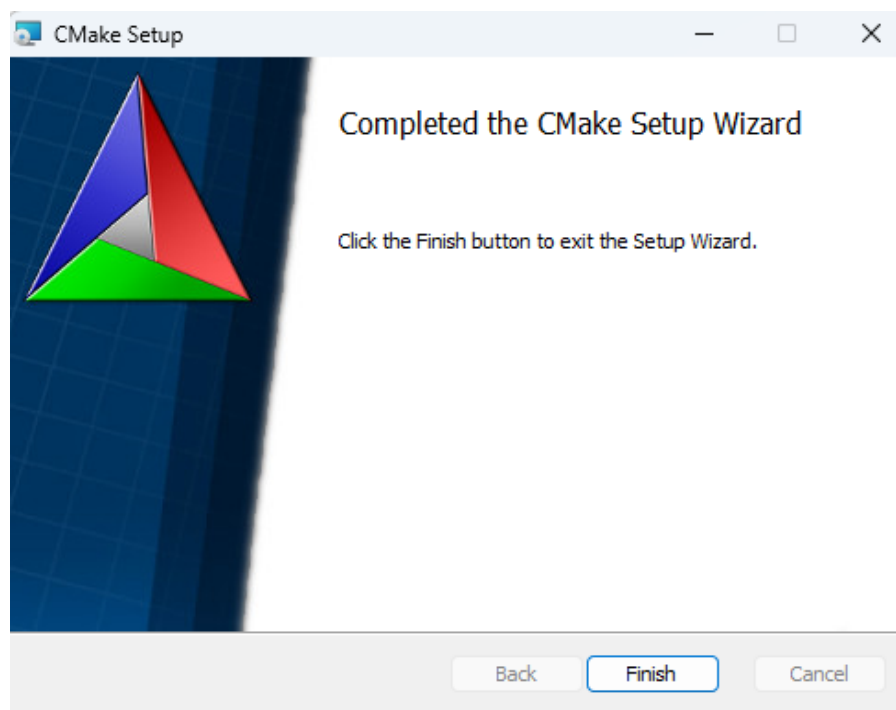
5. Click "Install", beware, **Admin** privileges are **required**:



6. Wait for the installation to end:

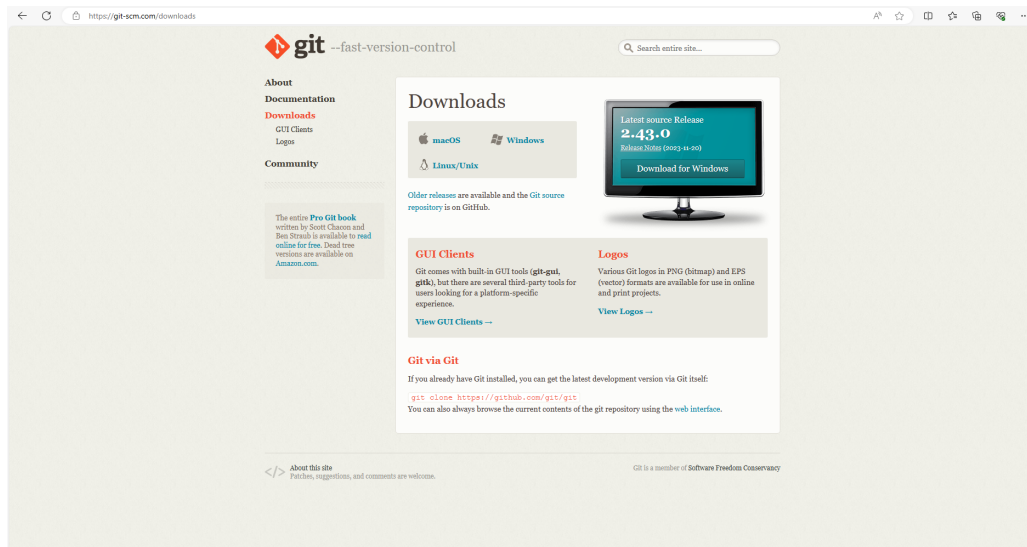


7. Click "Finish":

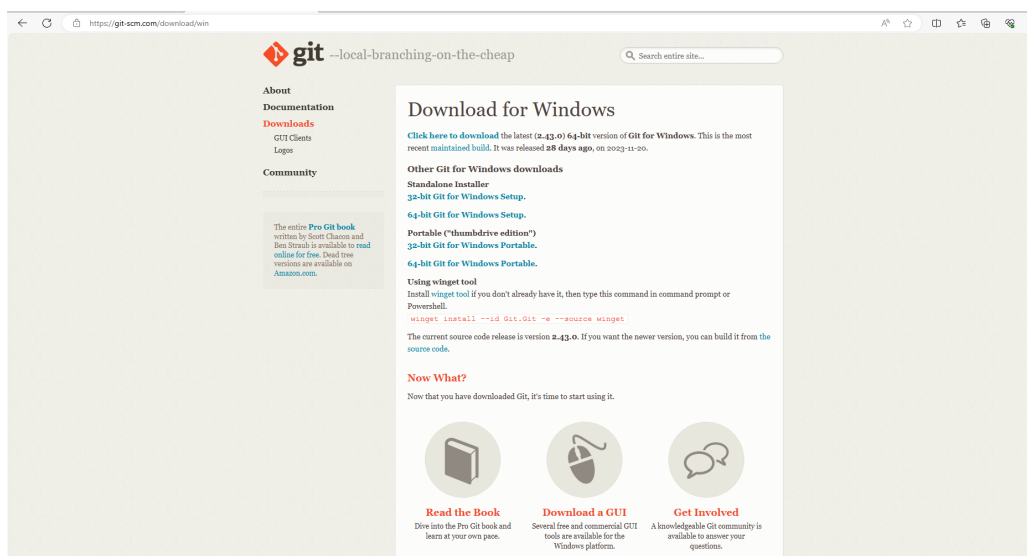


6 Git: Download & Install

To download [git](#), go to the [download](#) page and click on "*Windows*".

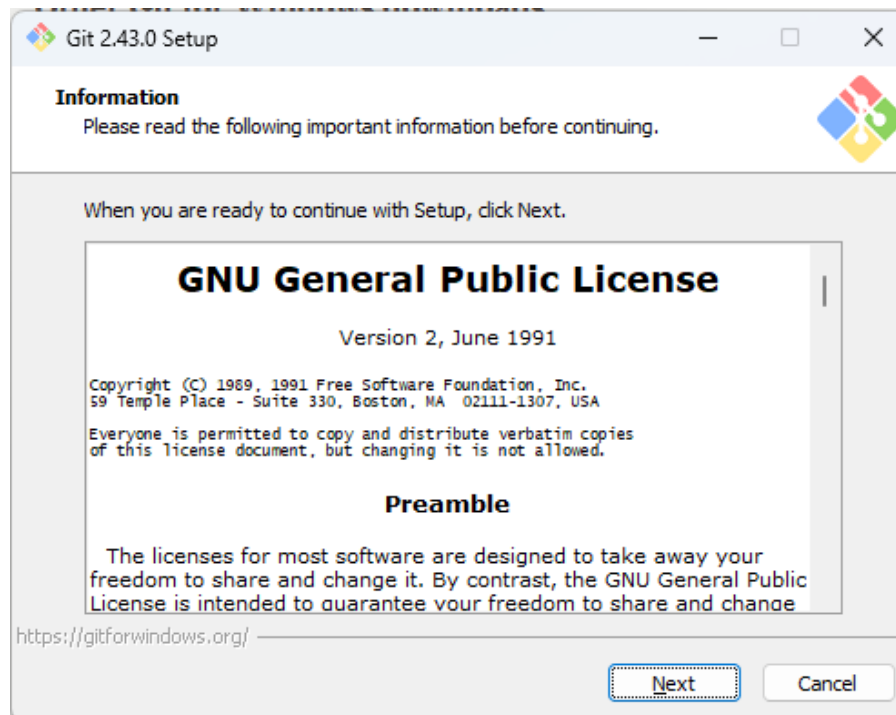


On the page that opens, select the appropriate *Standalone Installer* for your computer's processor.

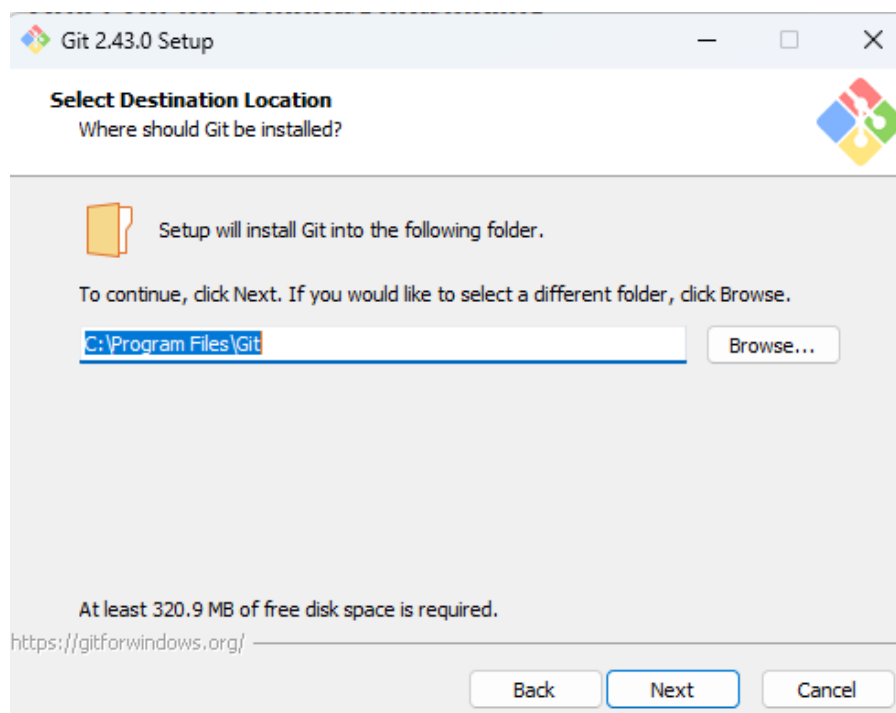


To install git open the download and follow the steps as shown in the figures.

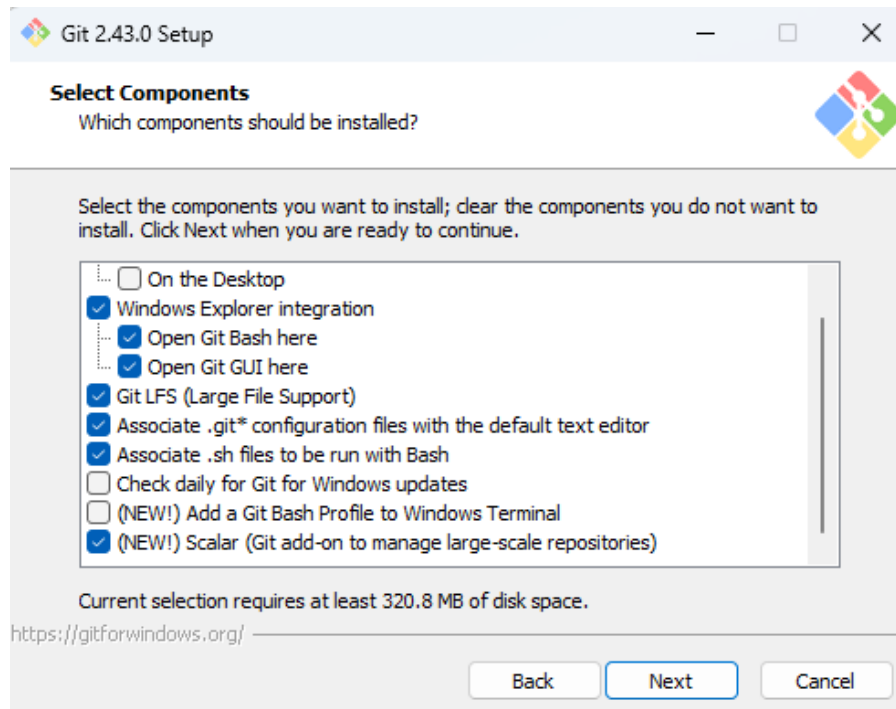
1. Accept the terms by clicking "Next":



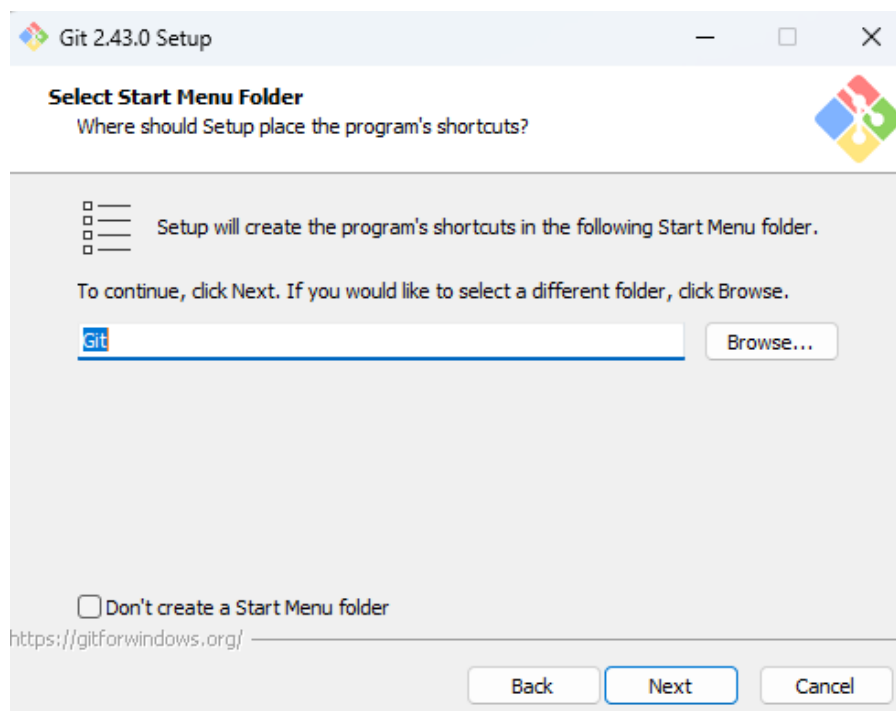
2. Chose an installation path and then click "Next":



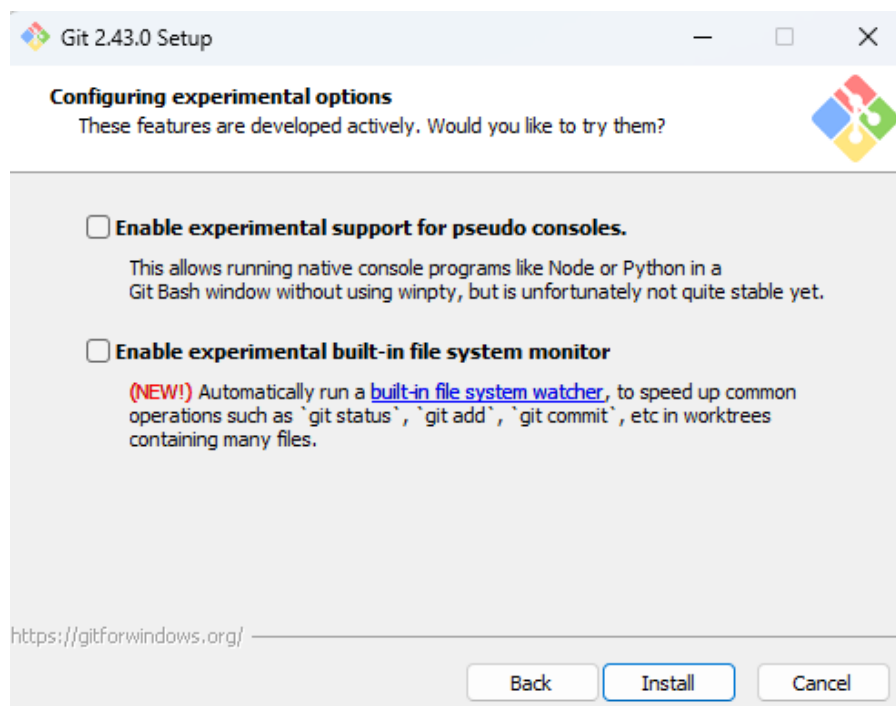
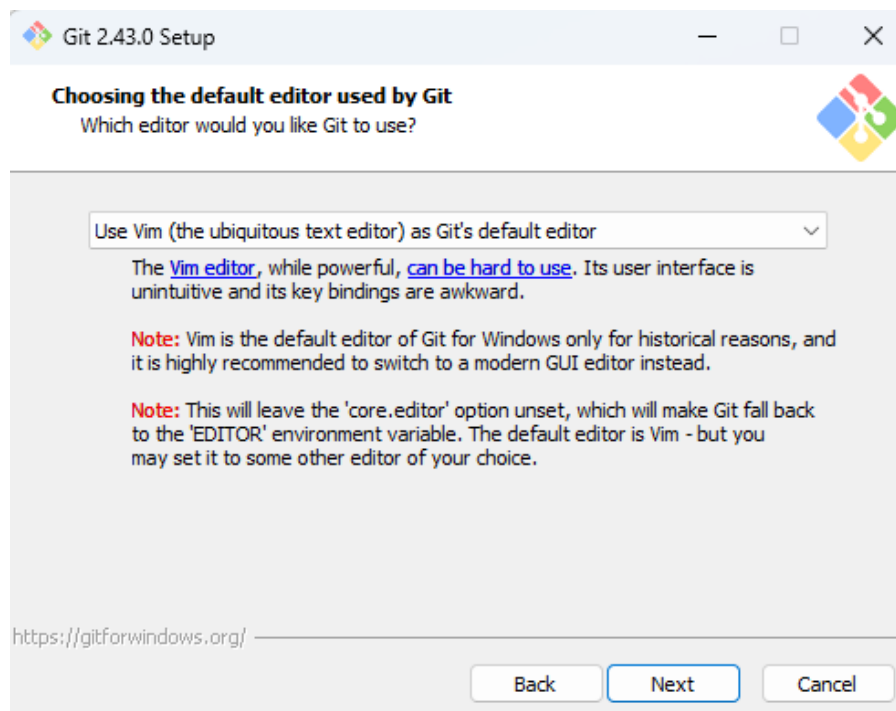
3. Tick the boxes as shown in the image below and then click "Next":



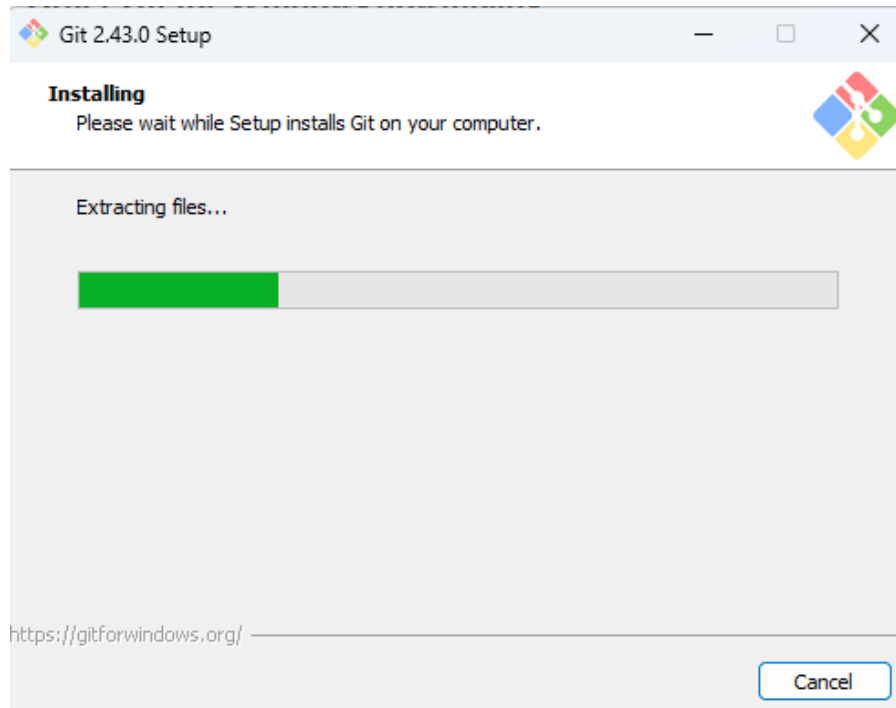
4. Optionally, you can create a *Start Menu Folder*:



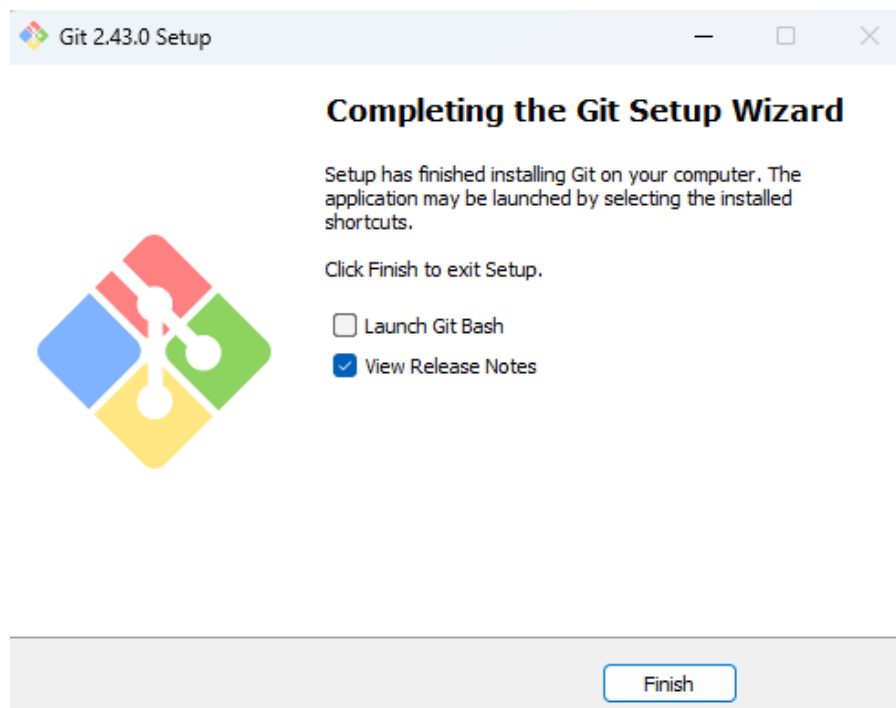
5. From now on do **not** modify the predefined settings and click "Next" until the "Install" button appears, click it to proceed with the installation:



6. Wait for the installation to finish:



7. Click "Finish":



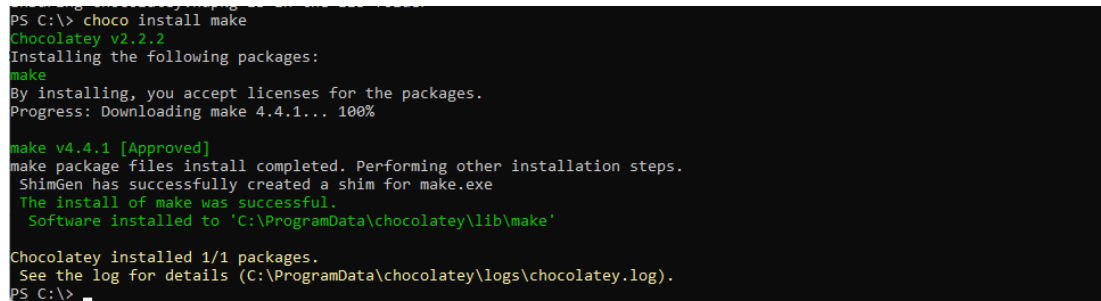
7 Make: Download & Install

Before installing the **Make** tool, you need to have on your computer **Chocolatey** which is a tool used for installing software via the Windows command line.

To install it, follow this steps:

1. Open Windows's **Powershell** with **Admin** privileges;
2. Type: `"Set-ExecutionPolicy AllSigned";`
3. When prompted, type: `"Y"`
4. Type all of this before pressing enter:
`"Set-ExecutionPolicy Bypass -Scope Process -Force;
[System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol
-bor 3072;
iex((New-Object System.Net.WebClient). DownloadString('https://chocolatey.org/install.ps1'))";`
5. Wait for the installation of Chocolatey to complete;

Finally, to install make, write the command `"choco install make"` in the Powershell and wait for the installation to finish.



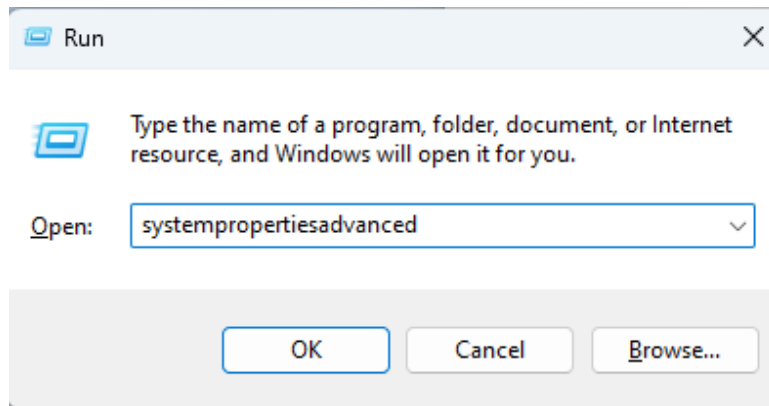
```
PS C:\> choco install make
Chocolatey v2.2.2
Installing the following packages:
make
By installing, you accept licenses for the packages.
Progress: Downloading make 4.4.1... 100%
make v4.4.1 [Approved]
make package files install completed. Performing other installation steps.
ShimGen has successfully created a shim for make.exe
The install of make was successful.
Software installed to 'C:\ProgramData\chocolatey\lib\make'

Chocolatey installed 1/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
PS C:\>
```

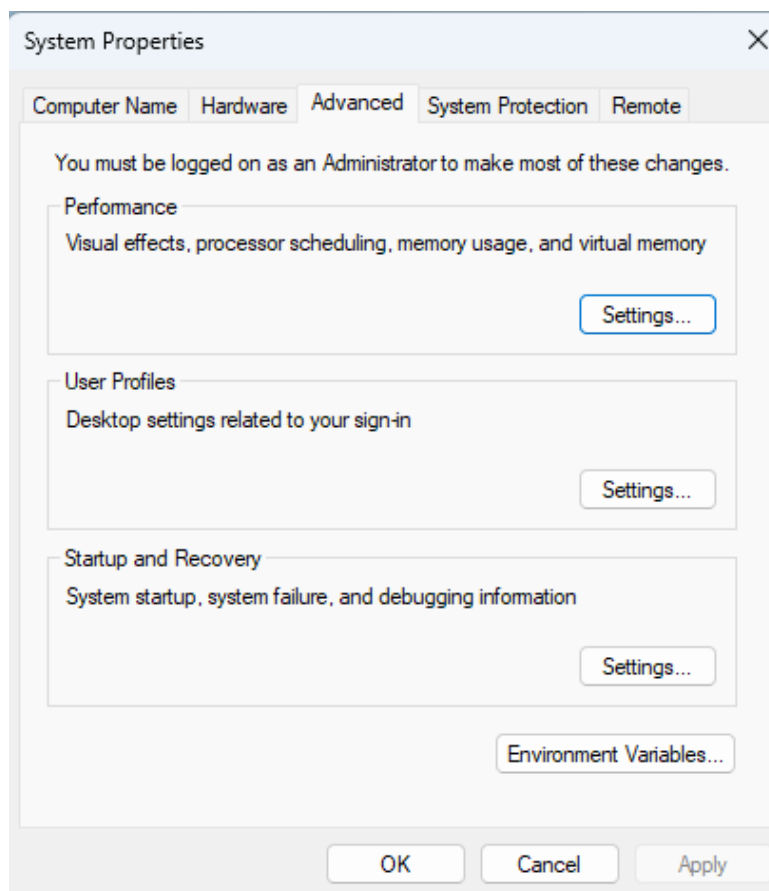
8 Add the Previously Installed Programs to the User Path Environment

If you have followed the installations of the programs correctly, then *make* and the *Arm GNU Toolchain* paths should have already been added to the User Path Environment. On the other hand, *Qemu* and *CMake* should have to be added manually. To do this:

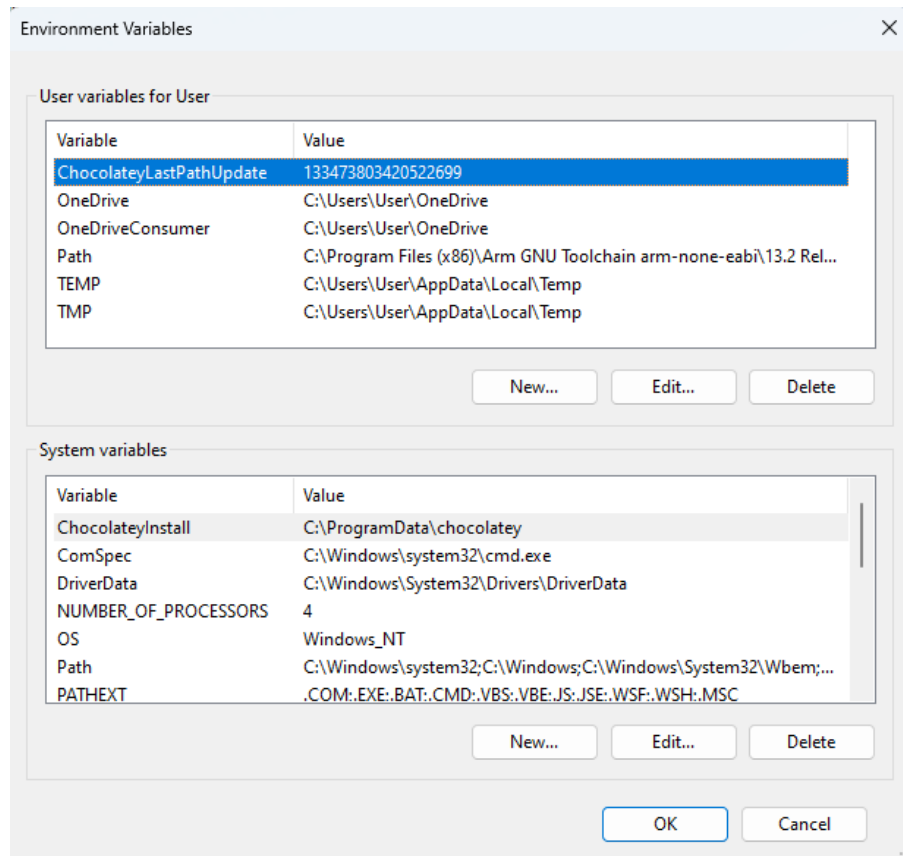
1. Open Windows's [Run Command](#) with the combination Windows + R. Then type "*systempropertiesadvanced*" and click "OK";



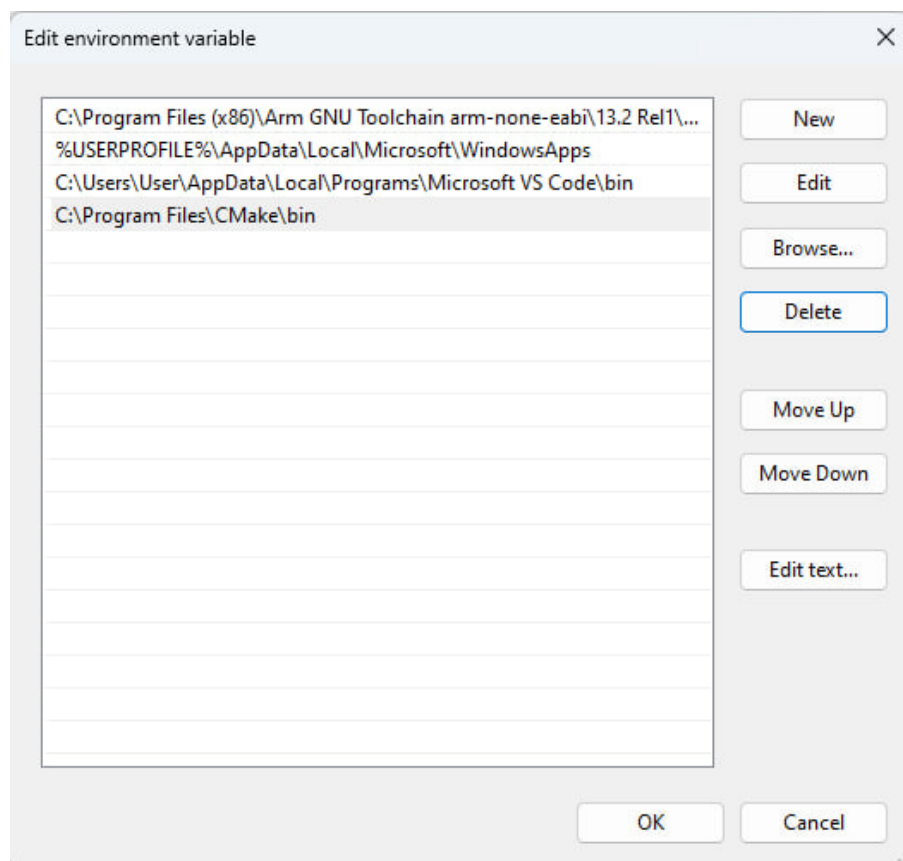
2. The "System Properties" window should appear on screen:



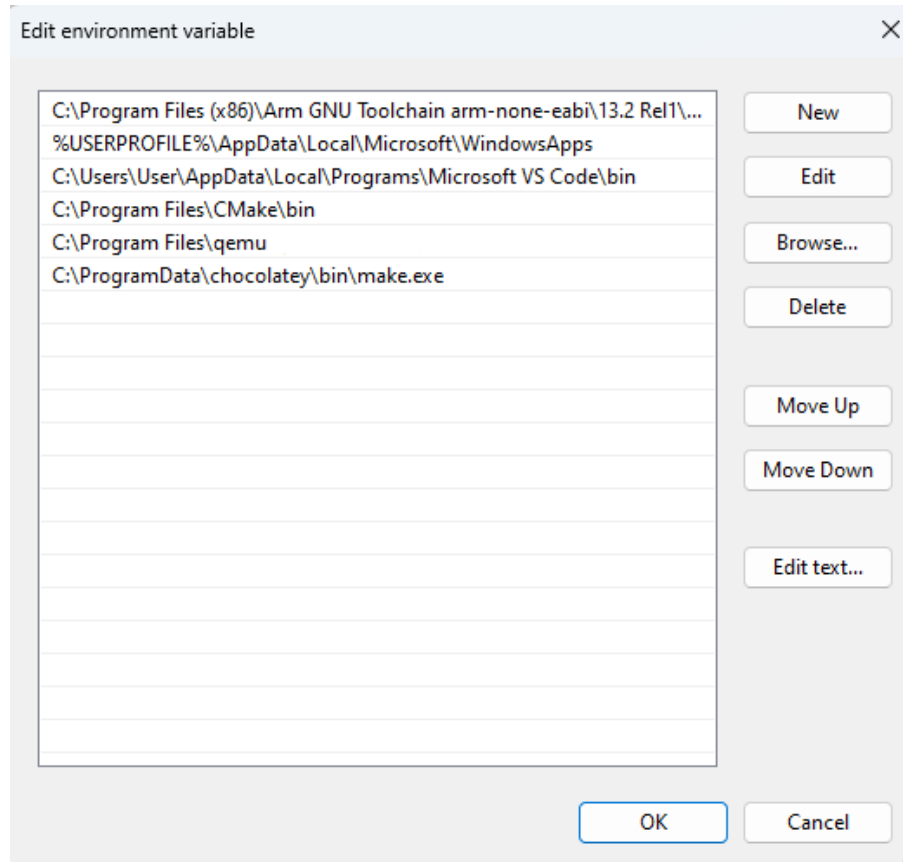
3. Click on "Environment Variables", the following window should appear:



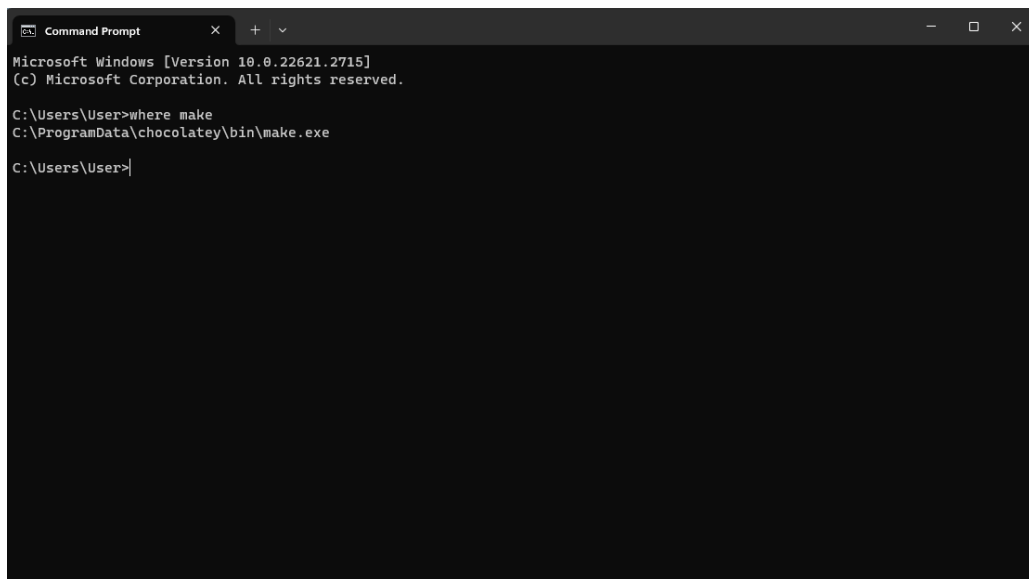
4. Under the "User variables for User", select the "Path" variable and then click "Edit".



5. A new window should open, as shown in the image above. Here you can add the paths of **all** the missing tools, such as *CMake* and *Qemu* by clicking on the "New" button and typing in. Once finished, the Path environment should look something like this. Please **note**, the order in which the variable appear is arbitrary.



If you are having troubles finding the path of some tool, open Windows's [Command Prompt](#) and type "*Where ...*" where ... is the name of the tool. As output, the Prompt should return to you the path of the program.

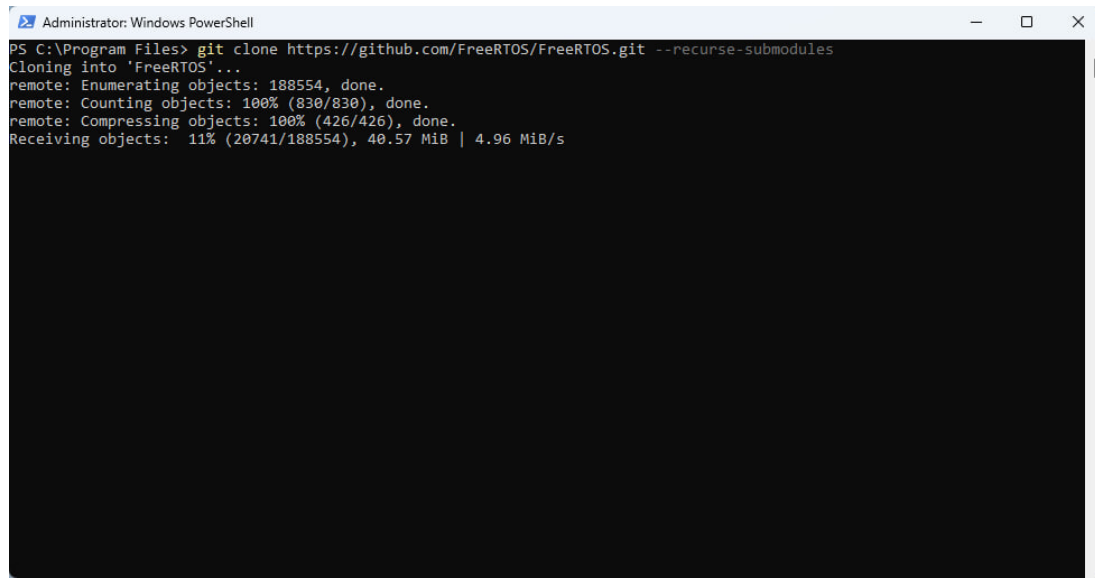


Do **not** worry if in your User Path Environment there are more more or less variables,, the important thing is that the paths of *Qemu*, *Arm GNU Toolchain*, *CMake* and *Make* are present.

9 Cloning the FreeRTOS Embedded OS from Git

Now you can finally begin the installation process of the **FreeRTOS** embedded OS. **Beware**, you are going to need to use the Git tool.

1. Open the Powershell with **Admin** privileges;
2. Use the [Windows Shell commands](#) to navigate to the folder where you want to save FreeRTOS;
3. Type: "`git clone https://github.com/FreeRTOS/FreeRTOS.git --recurse-submodules`";
4. Press enter and wait for the installation to terminate;



```
Administrator: Windows PowerShell
PS C:\Program Files> git clone https://github.com/FreeRTOS/FreeRTOS.git --recurse-submodules
Cloning into 'FreeRTOS'...
remote: Enumerating objects: 188554, done.
remote: Counting objects: 100% (830/830), done.
remote: Compressing objects: 100% (426/426), done.
Receiving objects: 11% (20741/188554), 40.57 MiB | 4.96 MiB/s
```

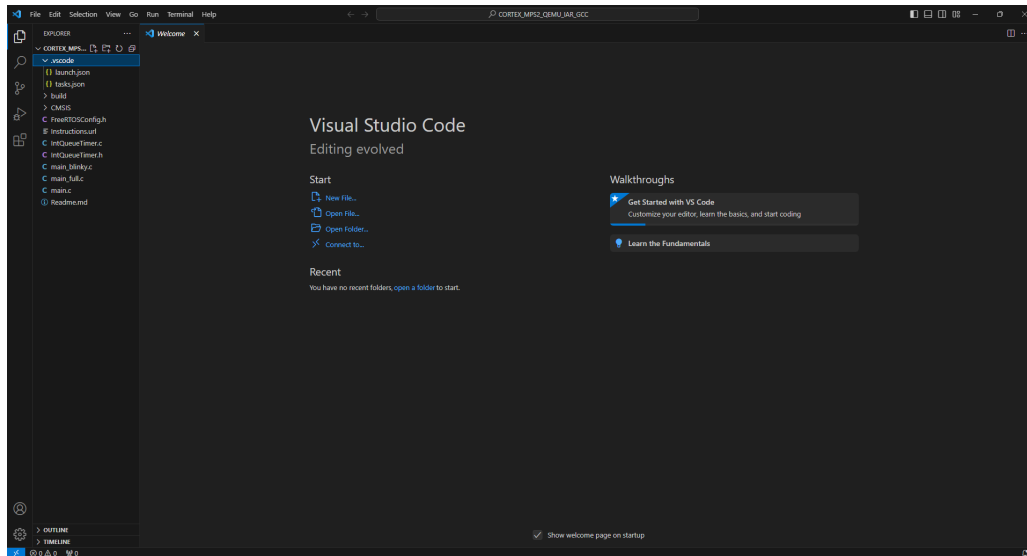
10 Coding Environment Set-up for FreeRTOS

The Git repository we have found contains many demo projects which vary in configuration to adapt to the various embedded environment FreeRTOS can run on.

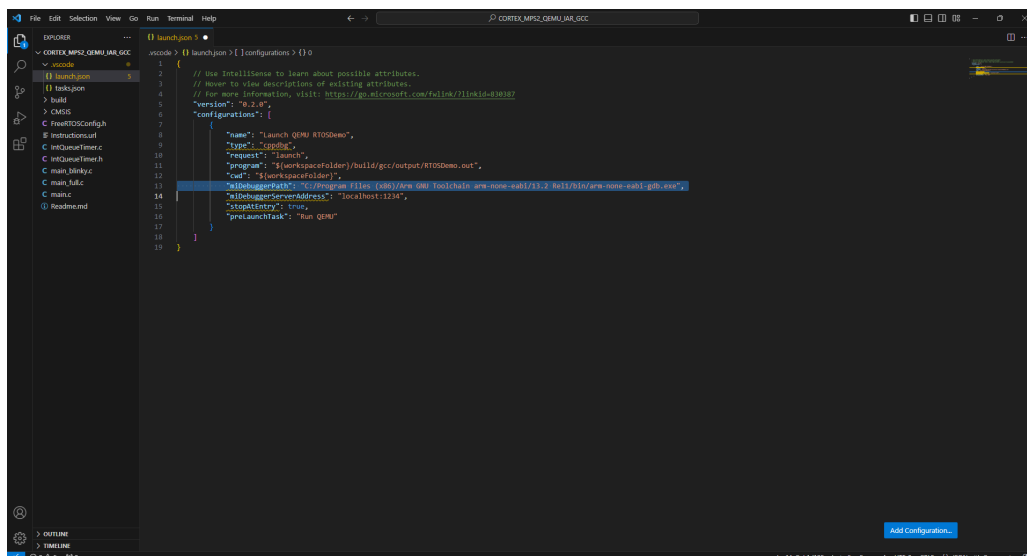
Since we are **not** using a physical board but QEMU, which is an **emulator** of an Instruction Set Architecture, we decided to use the **CORTEX_MPS2_QEMU_IAR_GCC** demo project. This project can then be edited to create some examples to show the functionalities offered by FreeRTOS or to implement some more complex programs such as new schedulers or memory management tasks.

To make this demo project run you need to:

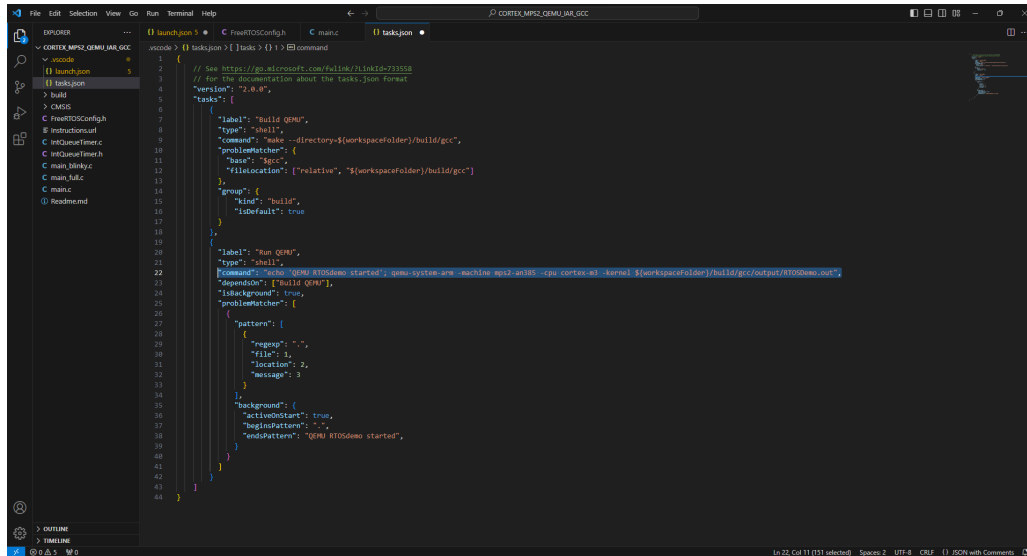
1. Open Visual Studio Code, go to 'File' -> 'Open Folder' and look for the path where you cloned FreeRTOS. Select this subfolder: `"../FreeRTOS/FreeRTOS/Demo/CORTEX_MPS2_QEMU_IAR_GCC"`;



2. In the top left, click on `".vscode"` and then double-click on `"launch.json"` to open the JSON file. As shown in the image, go to the highlighted line starting which starts with `"miDebuggerPath"`. Change the path to the path `"arm-none-eabi-gdb.exe"`, it should be located in the folder where you installed the ARM GNU Toolchain;



3. In the "task.json", go to line 22 and **delete** everything that comes **after** "echo 'QEMU RTOSdemo started'; qemu-system-arm -machine mps2-an385 -cpu cortex-m3 -kernel \${workspaceFolder}/build/gcc/output/RTOSDemo.out " as shown in the figure below. This will let us use Qemu the as the output of the *printf* function calls.

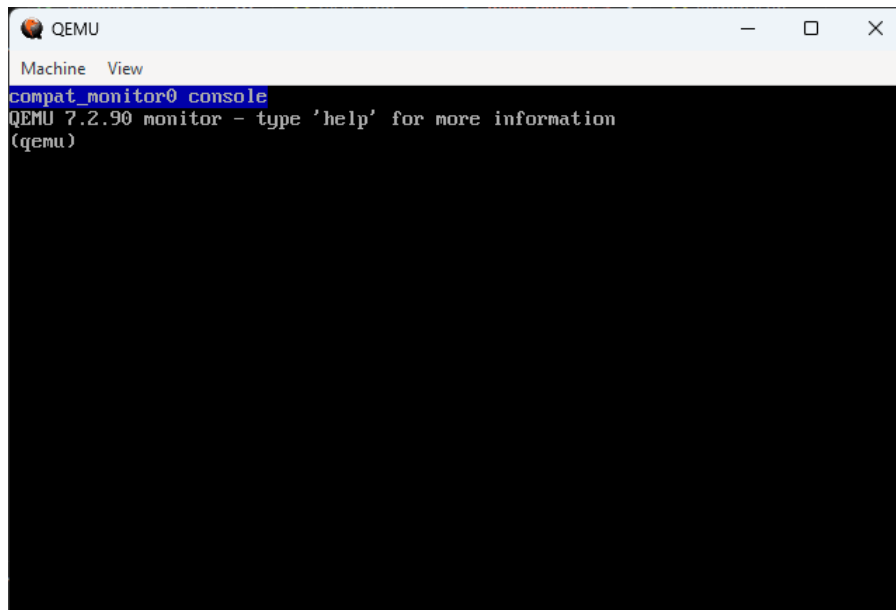


```
{
  // See https://go.microsoft.com/fwlink/?linkid=733558
  // for the documentation about the tasks.json format
  "version": "2.0.0",
  "tasks": [
    {
      "label": "Build QEMU",
      "type": "shell",
      "command": "make --directory=${workspaceFolder}/build/gcc",
      "problemMatcher": {
        "base": "gcc",
        "fileLocation": [ "relative", "${workspaceFolder}/build/gcc" ]
      },
      "group": {
        "kind": "build",
        "isDefault": true
      }
    },
    {
      "label": "Run QEMU",
      "type": "shell",
      "command": "echo 'QEMU RTOSdemo started'; qemu-system-arm -machine mps2-an385 -cpu cortex-m3 -kernel ${workspaceFolder}/build/gcc/output/RTOSDemo.out",
      "dependencies": [ "Build QEMU" ],
      "isBackground": true,
      "problemMatcher": [
        {
          "pattern": [
            {
              "regexp": ".*",
              "file": 1,
              "location": 2,
              "message": 3
            }
          ],
          "background": {
            "activeOnStart": true,
            "beginPattern": ".*",
            "endPattern": "QEMU RTOSdemo started",
            "isBackground": true
          }
        }
      ]
    }
  ]
}
```


11 Run the FreeRTOS Demo Project with QEMU

To actually run the project's code, you need to:

1. On the left bar of Visual Studio Code, look for the "Run and Debug" button and click it;
2. On the upper part of the new bar, look for the green arrow while "*Launch QEMU RTOSDemo*" is selected, press the arrow to start running the program. **Note**, any breakpoint has been added to the code, this would start the debugging of the program;
3. Wait for the project to be built by Visual Studio Code. Once the project has been built, QEMU will open on your screen, as shown in the figure below:



4. To view the printouts of the various tasks, click on the view button at the top, then click **serial0**;

