**GCC C-MAKE FRAMEWORK**

Author : Long Dao | <https://louisvn.com>

1. **Introduce**

* GCC C-Make Framework is designed to manage C/C++ software projects.
* Allows users to create, delete, compile, run, and export test reports for any project.
* To manage a project, I need you to install some of the following software to use the framework.
  + [**Toolchain GCC**](https://winlibs.com/). This is a compiler that takes care of the process of translating from C code to machine code so that the program you write runs on your PC. Download **GCC version 13.2.0 (with POSIX threads - UCRT) 32bit** for environmental consistency.
  + Makefile. This is a replacement for typing complex commands into simple commands on your PC's Terminal control window.
  + [**Cygwin**](https://www.cygwin.com/install.html). If you are a Windows user, you need to install this tool to emulate a Linux environment to make Makefile and other tools work stably.
  + [**VSCode**](https://code.visualstudio.com/download). This is extremely popular word processing software when you write code. It helps you display colors, command prompt when writing, supports debugging configuration, ...
  + [**Python**](https://www.python.org/downloads/). In Python, [**Gcovr**](https://gcovr.com/en/stable/) tool allows you to generate code coverage reports.

1. **Install GCC**

* To install GCC, go to the following URL (Github): [**Click here**](https://github.com/brechtsanders/winlibs_mingw/releases/download/13.2.0-16.0.6-11.0.0-ucrt-r1/winlibs-i686-posix-dwarf-gcc-13.2.0-llvm-16.0.6-mingw-w64ucrt-11.0.0-r1.zip)
* After the download is complete, press **Window + E**. At the File Manager window, go to drive **C**, create a folder called "**Toolchain**”.
* Then extract the downloaded file to the newly created folder. After extracting, you will have a path:“**C:\Toolchain\mingw32**”.
* Click **on Window** and search for "**environment**". You'll see the following app:



* Tap select the app. Select the **Advanced** tab > **Environment variables…**
* A new window appears. At the **System variables** table> **Path** > **Edit…**
* A new window appears. Click **New** to add the following paths:
  + **C:\Toolchain\mingw32\bin**
  + **C:\Toolchain\mingw32\i686-w64-mingw32\bin**
* Once the addition is complete, press **OK** one by one to exit.
* Finished! To check if you have installed successfully, press **Window** and search for "**cmd**”



* Open the application, type **"gcc -v**" in the window. If you see a message like this, done.



1. **Install VSCode**

* To install VSCode, go to the following URL: [**Click here**](https://code.visualstudio.com/download)
* After downloading, click open setup file. Choose “**I accept the agreement**” > “**Create a desktop icon**” > **Next** > **Install**
* Wait for the installation to finish and it is successful!
* Next, you need to open the newly installed VSCode application to install the necessary tool packages.
* Press Ctrl  **+ Shift + X**. An "Extensions" tab pops up.
* Type in the search bar and download the following packages in turn:
  + **C/C++** ; **C/C++ Themes** ; **C/C++ Extension Pack**
  + **Code Runner**
  + **Doxygen Documentation Generator**

A screenshot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

* After the installation is complete, press the key combination [ **Ctrl + ,** ] to open the "Settings” tab.
* Type "**Code-runner: Run In Terminal**" into the search bar. Then click on the square according to the picture below:



1. **Install Cygwin**

* To install Cygwin, go to the following URL: [**Click her**](https://www.cygwin.com/setup-x86_64.exe)**e**
* Same step 2 of GCC installation, Go to drive **C**, create a folder that is “**Cygwin**”.
* Next, copy the downloaded setup file into the folder you just created: **C:\Cygwin**
* Click open the copied app. Choose **Next** > **Install from Internet**
* In the new window, enter **"C:\Cygwin**" in "**Root Directory**". At the bottom select **All Users** > **Next**
* In the new window, enter “**C:\Cygwin**” to “**Local Package Directory**” > **Next**
* Select Use System Proxy **Settings** > **Next**
* Click on any URL to download the data. (Should choose the 4th URL) > **Next**
* Wait a while until a "**Select Packages**" window appears. At **View** select **Full**
* Search for the following packages and select the latest versions : **make**, **zip**, **unzip**, **sed**, **cygrunsrv, bc, tree**

Note, click on the square with the down arrow to select the version. For example, the following figure is shown:

A screenshot of a computer

Description automatically generated with medium confidence

* After selecting the installation version for all the packages listed above, click  **Next**  > **Next** to proceed with the installation.
* Wait until the installation is complete, click **Finish**.
* Then, you open "**Environments**" similar to step 4+ of the GCC installation and add the following paths:
  + **C:\Cygwin\bin**
  + **C:\Cygwin\sbin**
  + **C:\Cygwin\usr\sbin**
* After adding Environments, **right-click** on the newly installed application icon. Select **"Run as administrator**" and wait for the application to show up.
* Type in "**cygserver-config**" > **Enter** >... Type "**yes**". Wait for the run to finish and close the window.
* **Note** : If you have Git installed on your PC. Please move Cygwin's paths above Git's paths in Enviroments.
* Restart your PC - required.

1. **Install Python**

* To install Python, go to the following URL: [**Click her**](https://www.python.org/downloads/)**e**
* Click the button "**Download Python ...**" to download the app.
* Press **Window + E**. In the File Manager window, go to drive **C**, create nested folders as follows: "**Toolchain > Python > Python311**". (Replace "311" as your version of Python).
* After downloading, Click open the application. At the settings window, select "**Customize installation**". Next, click **all** the options in "**Optional Features**", especially "**Pip**". Then click "**Next**".
* In the "**Advanced Options**" window, also click all options. Then in the "**Customize install location**" section, enter the path to the previously created folder **"C:\Toolchain\Python\Python311**" and select "**Install**".
* Wait for the installation to complete and click “**Close**”.
* After the installation is complete, open **"Path**" under "**Eviroments**". If you don't see the same 2 paths placed at the top (as shown below), copy the exact path on your PC and add it (move up to the top).

A screenshot of a computer program

Description automatically generated

* Then, open "**cmd**" and type **"python --version**" to check the version of Python. If you have version information printed, you have successfully installed it.

A black text on a white background

Description automatically generated

* Next, at the "**cmd**" window, enter **"pip install gcovr" to install** the Gcovr **tool**. Wait until the tool is successfully installed, you will see the text "Successfully installed gcovr-...". Type **"gcovr --version**" to check its version.



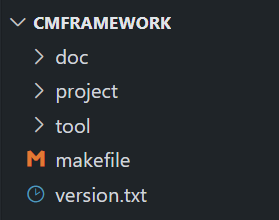
* Python version **3.11.4** and **Gcovr version 6.0** are running stably on this framework.

1. **Using the Framework**

* You need to download and unzip the Framework before working with it.
* To use the Framework, first open the **VSCode** editor.
* Để thêm Framework vào trình soạn thảo, nhấn liên tục 2 tổ hợp **Ctrl + K** và **Ctrl + O**.
* An "**Open Folder" window** will appear, select the path to the previously **extracted "CMFramework**" folder. Then select "**Select Folder**”.
* If a window similar to the one below appears,  **click** on the box and click "**Yes**”.



* You will see in the "**Explorer**" tab showing all the newly added frameworks.
* **THE STRUCTURE OF THE FRAMEWORK**



* + **doc** : Store documents for common use for every project. This "**Readme.en.docx**" file is also located in this folder.
  + **project** : Manage all your projects. Note, I have provided a sample project "**~temp**" as a foundation for creating other projects. You force not to delete it at all costs.
  + **tool** : Includes common libraries and test report generation tools, …
  + **makefile** : this is a file that manages all the backbone features to ensure the Framework works. You also **cannot** customize this file except for the values of the variables inside the "**Settings"** section.
  + **version.txt** : this file records the history of the current and version versions of the Framework.
* **THE STRUCTURE OF THE PROJECT IN THE FRAMEWORK**



Take the example on the template project "**~temp**" provided. Note that you **may not** edit or delete this project. You need to create a new project and execute on it.

* + **doc** : store of documents of this project.
  + **inc** : contains project header files (**.h**)
  + **dev** : contains project source files developed to test code coverage (**.c .cpp**)
  + **src** : contains the source files of the project (**.c .cpp .o**)
  + **user\_cfg.mk** : A makefile allows users to configurations for each specific project.
* **USE MAKE COMMANDS**

First, open a Terminal window on VSCode. Go to **Terminal** > **New Terminal**

A screenshot of a phone

Description automatically generated with medium confidence

At the **TERMINAL tab**, enter your requirements. Note, the root path on the terminal should be the path to "**CMFramework**". If this is not the case, use the command **"cd**" to move to.

* + **make setup** : initialize the tools, this command should run 1 time when you first use this framework. It will create executables inside the "**tool/bin"** directory.
  + **make** | **make all** : These 2 commands allow running {**make clean build run**}, see below.
  + **make clean** : Used to delete the output folder, where object files and executable files are stored. It also deletes test reports (if any).
  + **make build** : used to compile files (.**c**) to object files (**.o**), then link it to an executable file (**.exe**).
  + **make merge** : Used to merge object files into a single object file.
  + **make run** : used to run executable files (**.exe**) on Terminal.
  + **make report** : used to generate test reports if you use the "**utest.h**" library (available in the **tool**) in the C program to write test cases.
  + **make pack** : used to package the project, allowing you can share it with anyone.
  + **make vsinit** : used to create configuration files in VSCode, helping the software correctly link file paths on the display interface. When you move to another project, this command is **automatically** run. You just need to run this command again when you **update** the path in makefile (**user\_cfg.mk**).
  + **make move** : used to move to any project. Automatically add new ones if the project doesn't exist.
    - For example, **make move.proj1** : will mean moving to the project "proj1”.
    - Or: **make move.Group1: proj1**: move to project "Group1/proj1". You can nest in multiple groups, using the "**:**" to separate.
  + **make remove** : used to delete any project. If you delete the current project, it will automatically switch back to the "**~temp"** template project. The usage is the same as the **"move**" command.
  + **make import** : used to add a new project from a zip file shared fromsomeone else's "pack" command. The usage is the same as the **"move**" command. However, the variable "ZIP" will indicate the path to the zip file.
    - Example : **make** **import.proj1 ZIP=path/to/file.zip**
    - Note: the variable "ZIP" must be written adjacent to the "=" and path. Don't use white space in the middle.
  + **make list** : used to list all the files and folders that are in the project.
  + **make print** : used to print out the values of variables used inside makefile.
    - Example : **make print.VAR1.VAR2** : Print out the value of 2 variables : **VAR1** and **VAR2**.
    - Similarly, if printed multiple variables will be separated by dots.
    - This command is only for use by Admins or Framework developers.
* **NOTES**
  + All paths and file names must be written continuously. That is, there is no white space in the middle. You should be mindful when naming any one file or folder.
  + It is not allowed to name source files or header files that coincide with the name of the project. This may cause conflicts when creating output files.
  + In a project, each filename is unique across all directories. You are not allowed to compile 2 identical filenames as it may also have a conflict at the output.

===================== **END** =====================

Author : Long Dao | <https://louisvn.com>