창의적소프트웨어프로그래밍

Lab 1-1

TA. 정지나

snowgina00@hanyang.ac.kr

Instructor

TA. 정지나

이메일 snowgina00@hanyang.ac.kr

주소 공업센터 별관 503-2 (Software Engneering Lab.)

Today’s Contents

1. **Environment Setup**
   - **VS Code Installation**
   - **Compilier Setup**
   - **VS Code Setup**

How to install VS Code

Windows & macOS

Practice environment

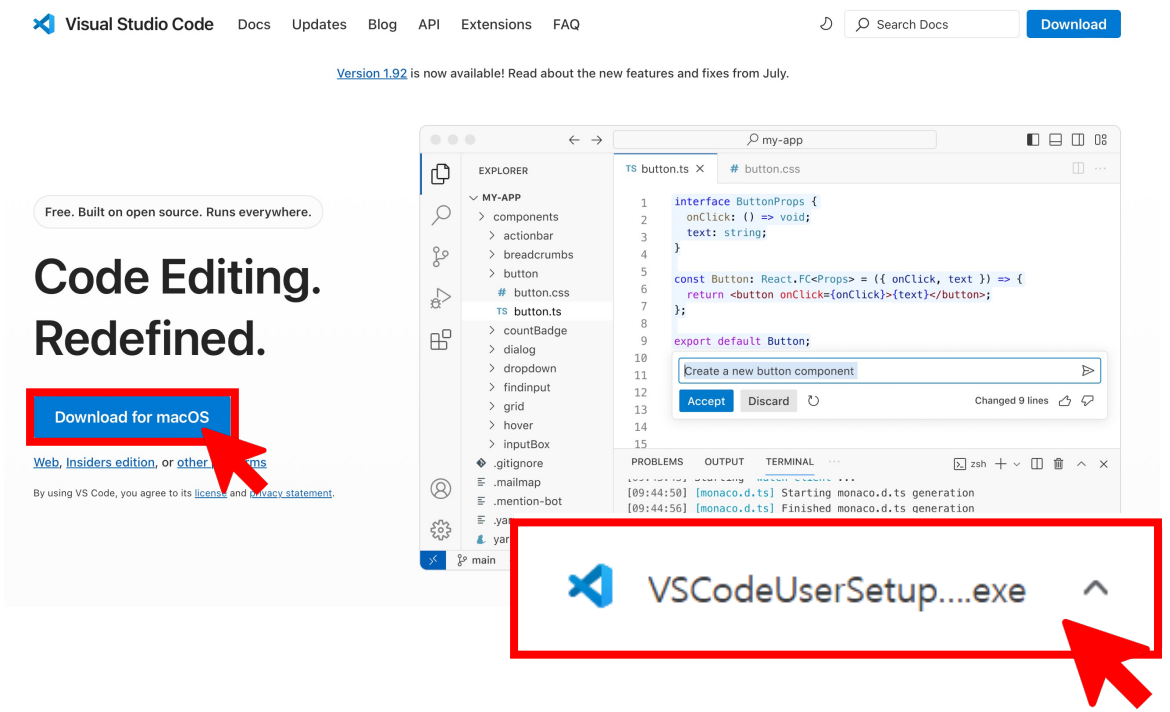
- **C++**

- **Visual Studio Code (VS Code)**
  - Code editor
    - a text editor program that is specifically optimized for writing and executing program codes
  - Supports Windows, macOS and Linux, ...
    - it can be used on almost any PC

VS Code Installation (Windows)

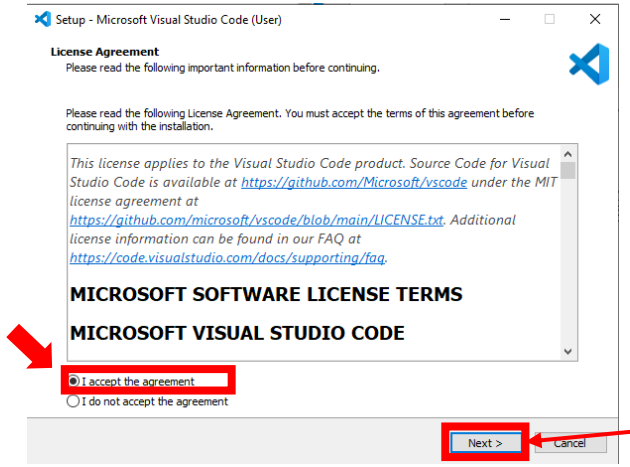
1. **Visit VSCode official website**
   - Link : https://code.visualstudio.com/

2. **Download and open .exe file**



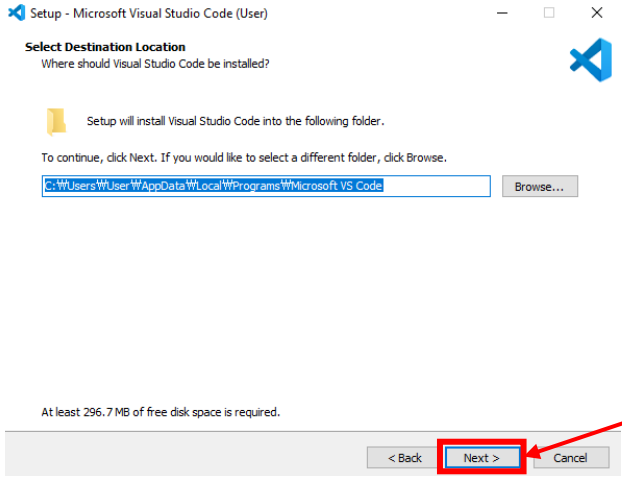
VS Code Installation (Windows)

3. Check “I accept the agreement”



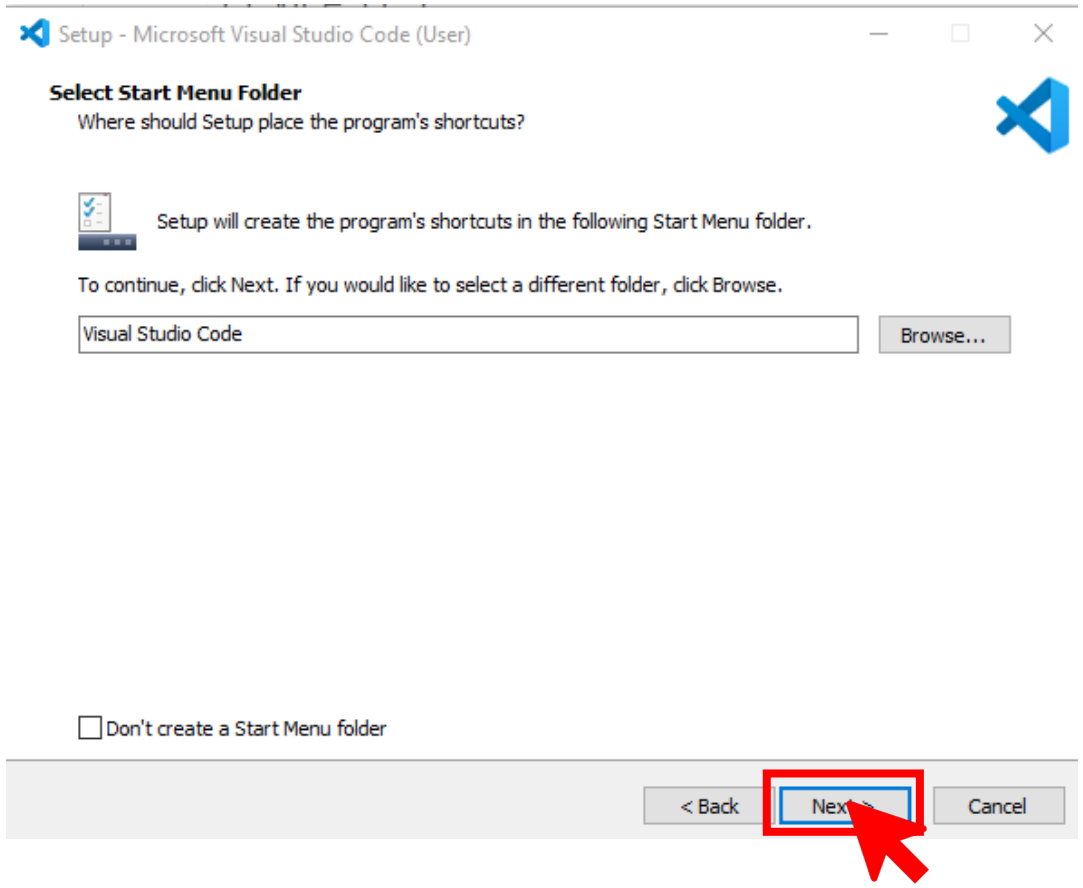
Click “Next” button

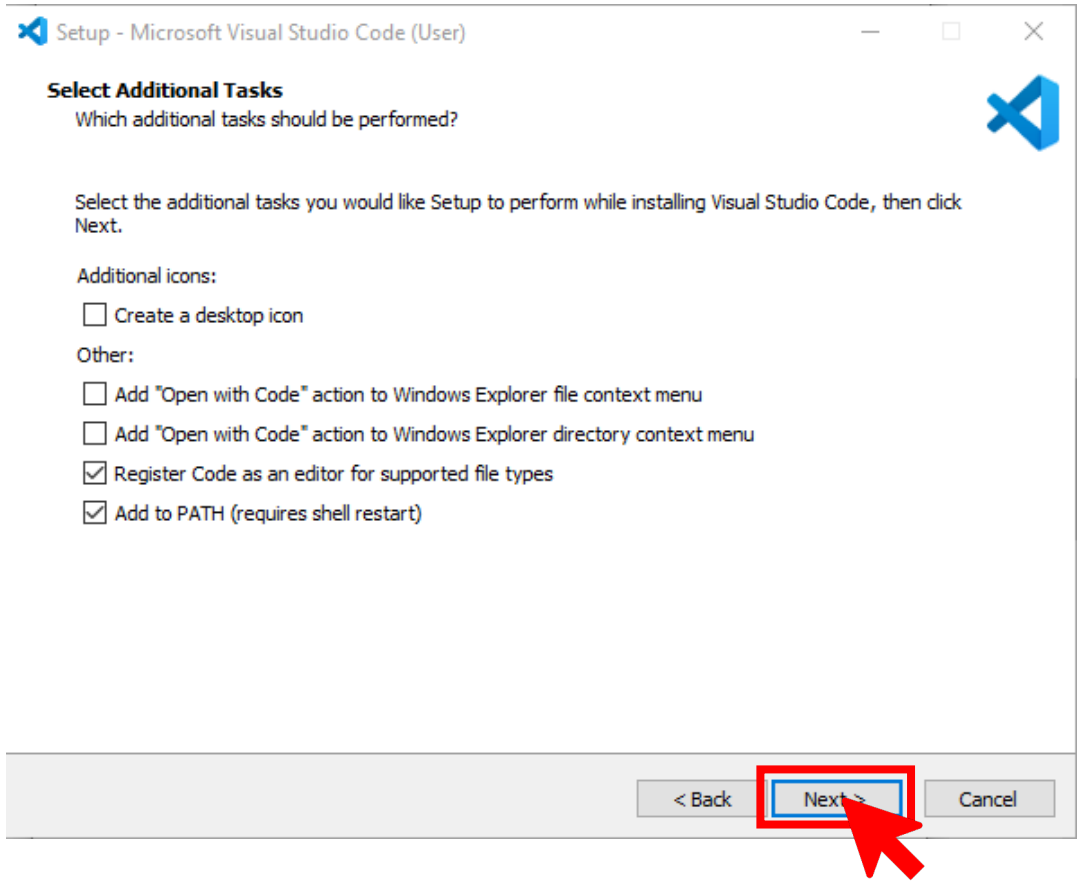
4. Check installation path



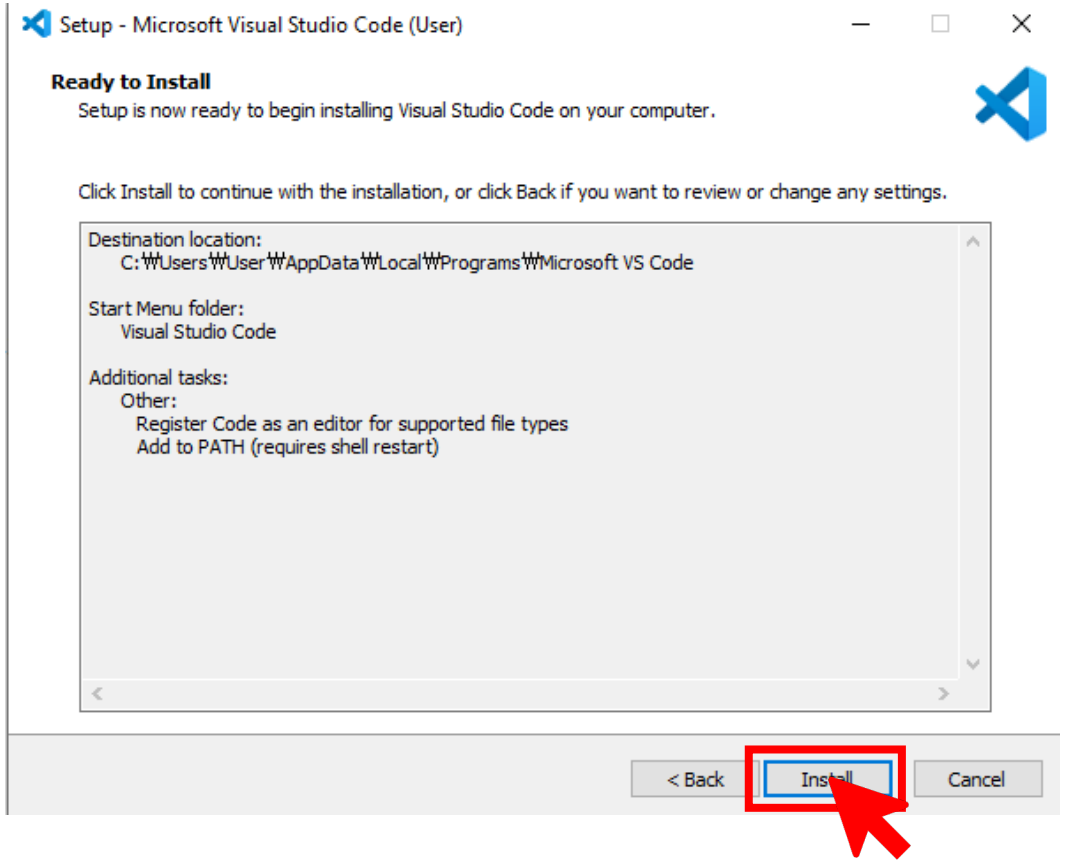
Click “Next” button

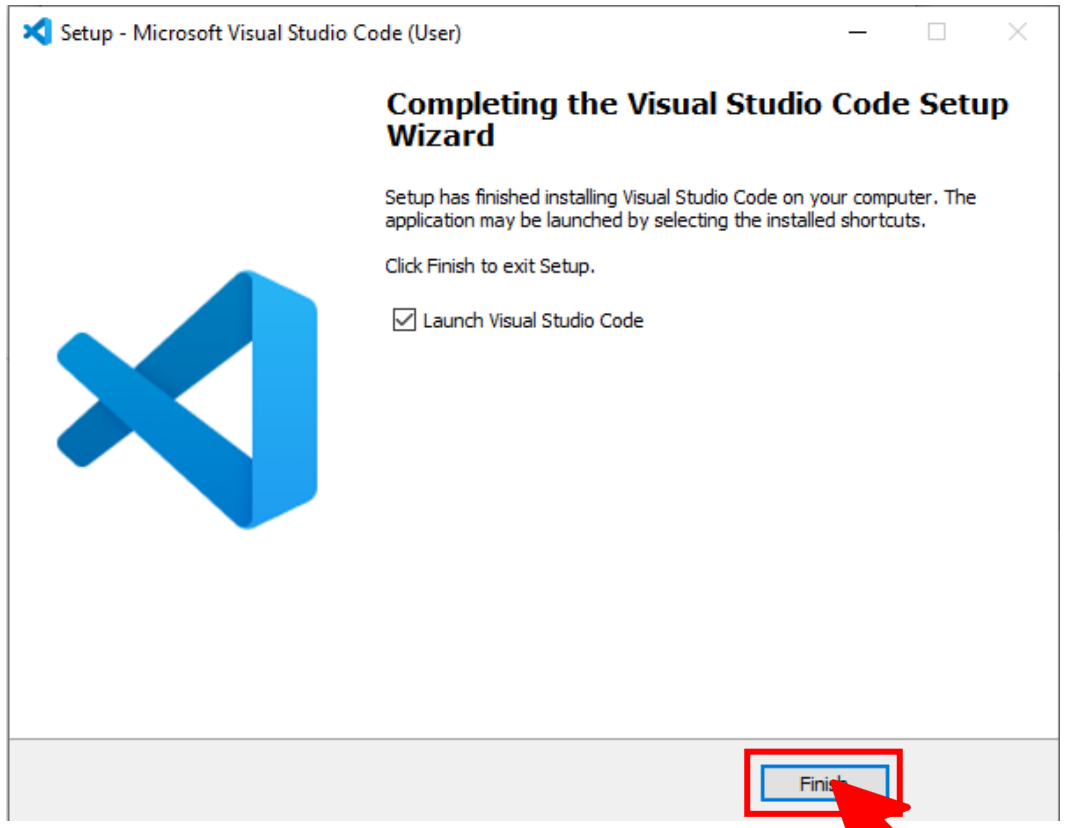
VS Code Installation (Windows)





VS Code Installation (Windows)

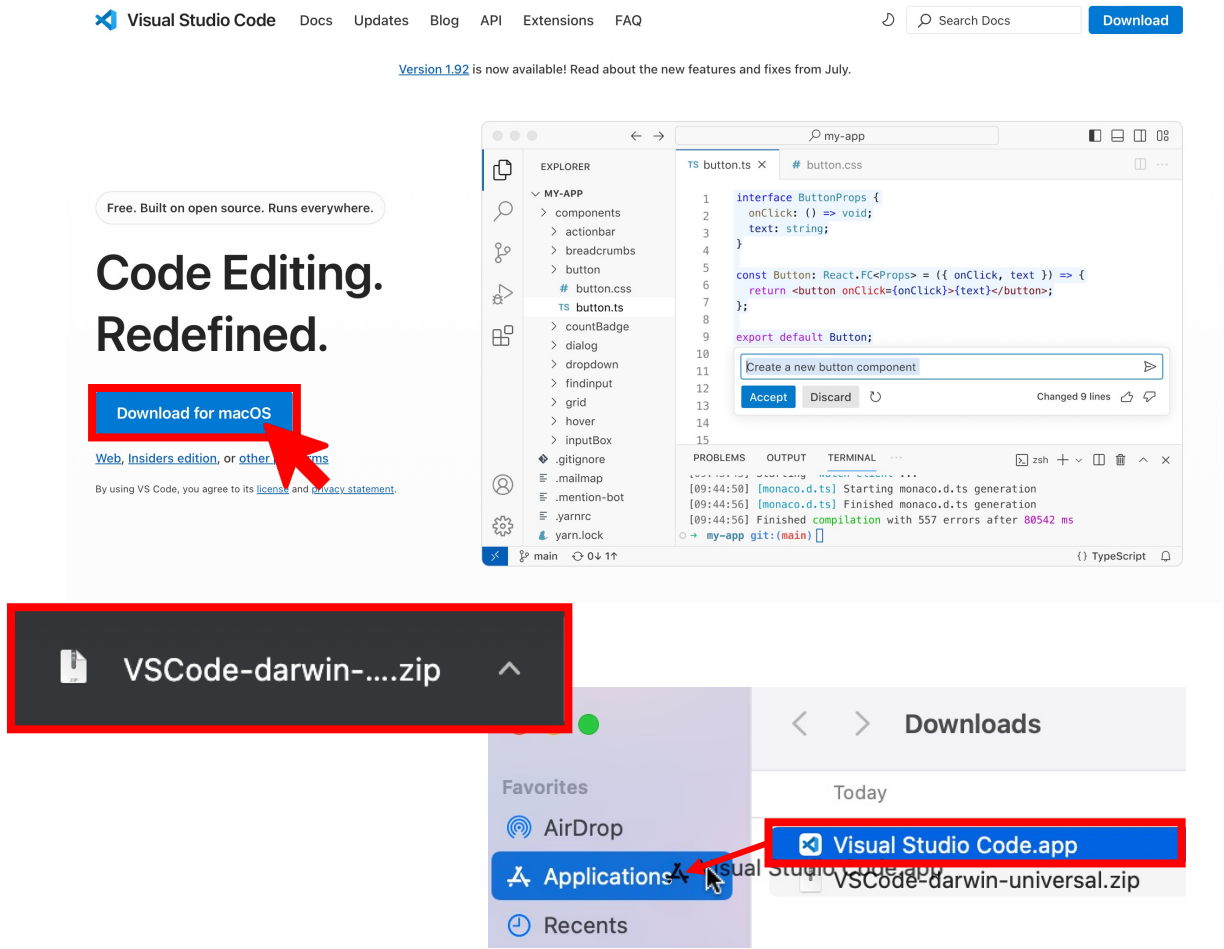




VS Code Installation (macOS)

1. **Visit VSCode official website**
   - Link : https://code.visualstudio.com/

2. **Unzip and move the VS Code to Applications folder**



How to install C/C++ Compiler

Windows

What is MSYS2?

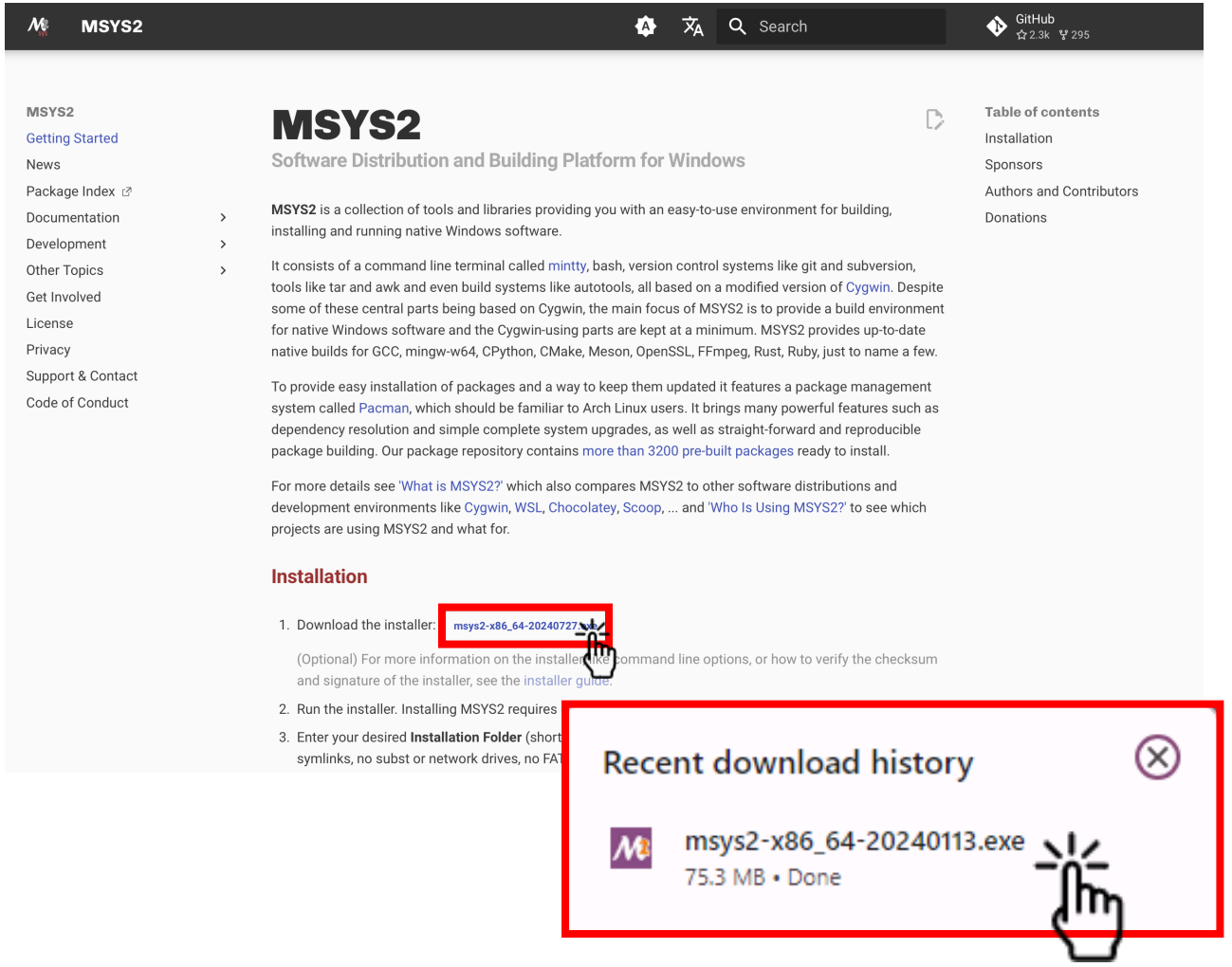
MSYS2

- Software distribution and a building platform for Windows.
- Provides a Unix-like environment, a command-line interface and a software
- Use a port of *pacman* for package management.

MSYS2 Installation

1. **Visit MSYS2 official website**
   - Link : https://www.msys2.org/

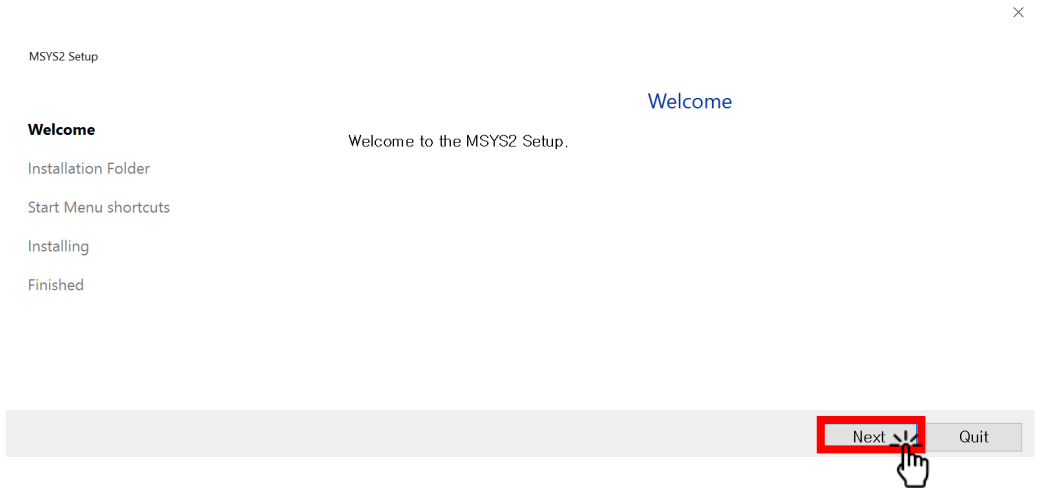
2. **Download and open installation file**

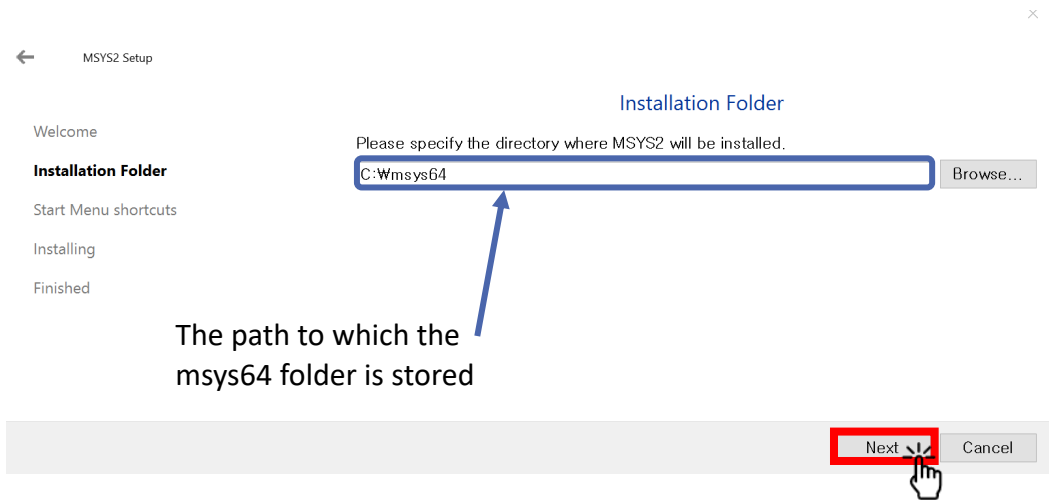


MSYS2 Installation

3. **Click “Next” button until “Installing” step**

4. **Click “Finish” button**
   - Run MSYS2 after the installation is completed.





MSYS2 Setup

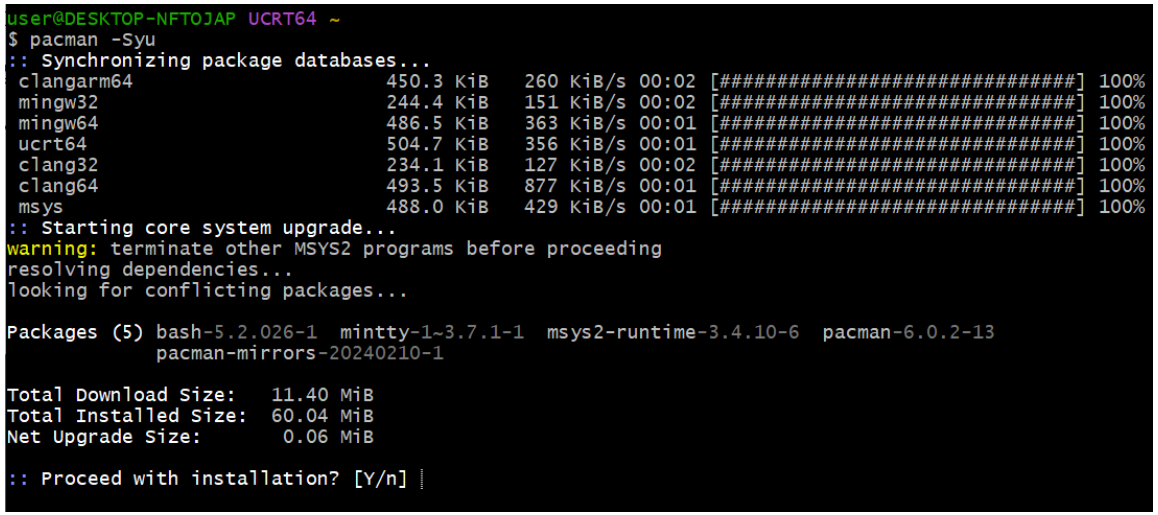
1. **Open MSYS2 Terminal**
   - Terminal : An interface on a computer system that allows users to enter commands and control the operation of the system
2. **Update the package database and base packages**
   - Write a command "**pacman –Syu**" in the terminal window.

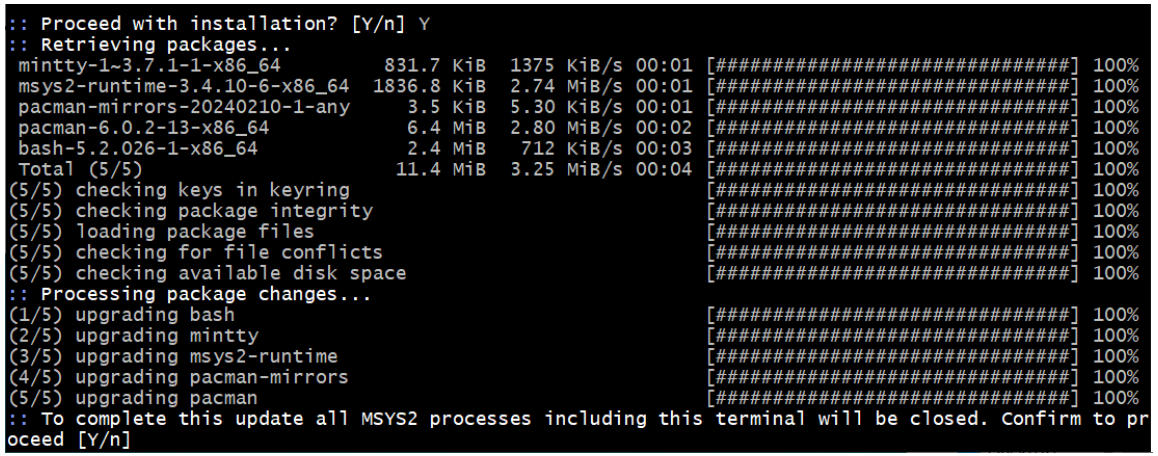
- ***pacman***(PACkage MANager)
  - Download, install, update, and uninstall **compiled binary** software packages on Arch **Linux** system.
- **-Syu** : pacman option
  - S : stand for sync
  - y : refresh(local cache)
  - u : system update

MSYS2 Setup

3. Write ‘Y’ and then press Enter

- If you see the message “To complete this update … Confirm to proceed[Y/N]”, write ‘Y’.



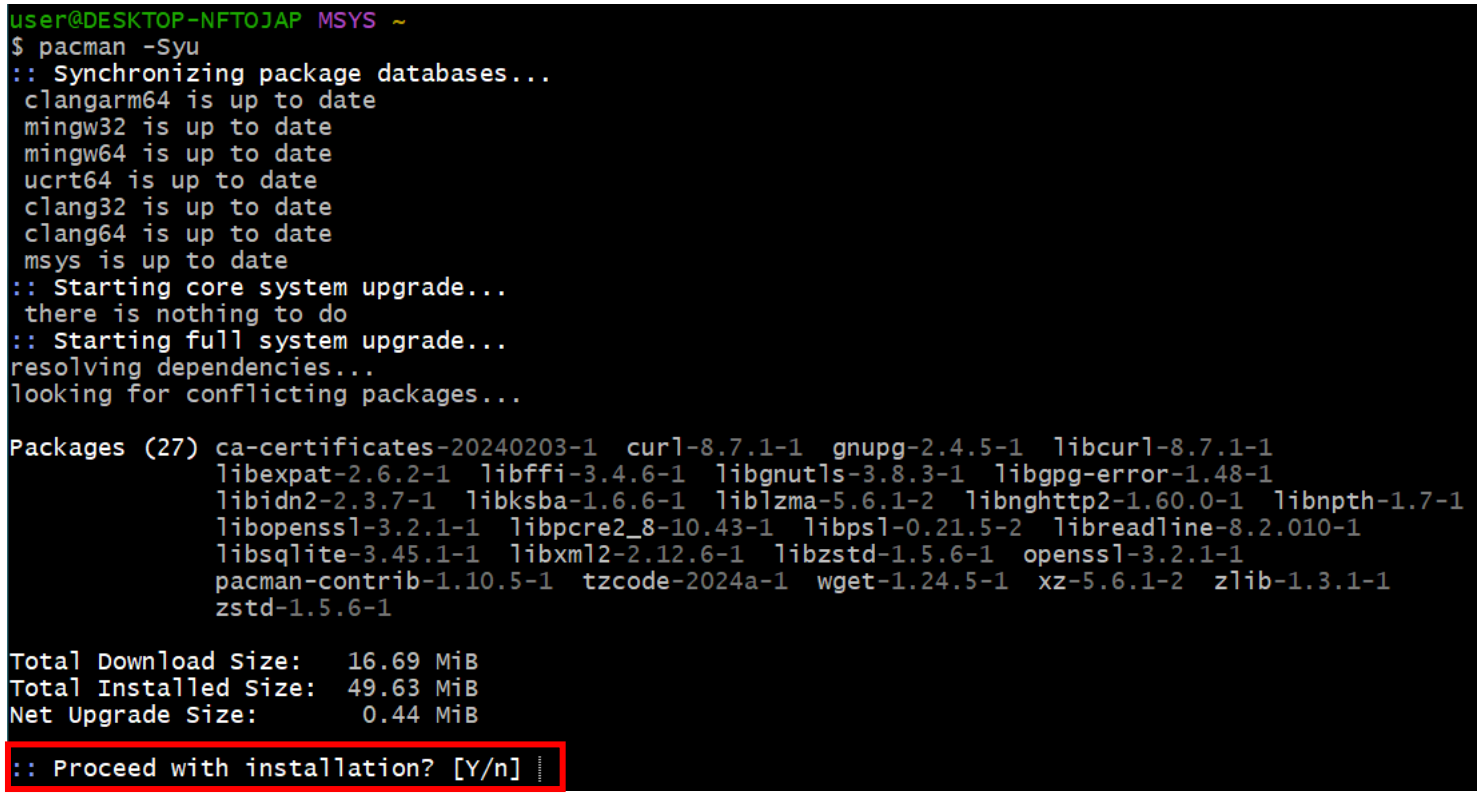


MSYS2 Setup

4. **Run ‘MSYS2 MSYS’ again and then write ‘pacman –Syu’ again.**
   - Update the rest of the base packages

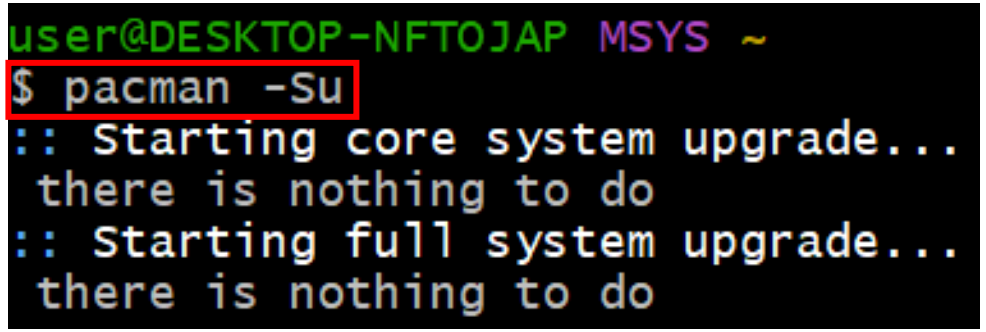
MSYS2 Setup

5. Write ‘Y’ and then press Enter



MSYS2 Setup

6. **Write the command “pacman –Su”**
   - If you see the message below, the basic installation and update have been completed.



MSYS2 Setup

7. Write the command to install clang package

- **pacman -S --needed base-devel mingw-w64-clang-x86_64-toolchain**

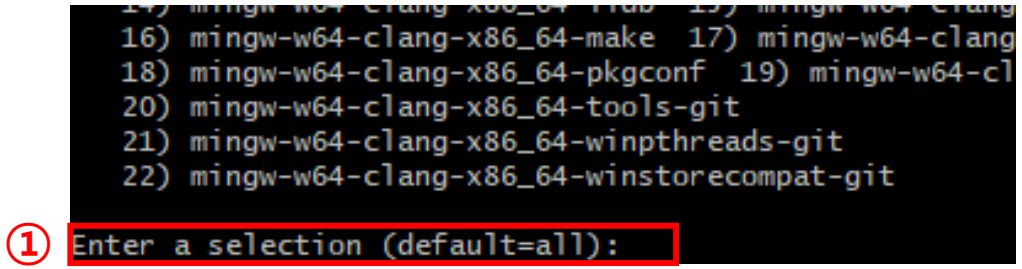
```
user@DESKTOP-NFTOJAP MSYS ~
$ pacman -S --needed base-devel mingw-w64-clang-x86_64-toolchain
:: There are 22 members in group mingw-w64-clang-x86_64-toolchain:
:: Repository clang64
   1) mingw-w64-clang-x86_64-clang  2) mingw-w64-clang-x86_64-clang-analyzer
   3) mingw-w64-clang-x86_64-clang-libs  4) mingw-w64-clang-x86_64-clang-tools-extra
   5) mingw-w64-clang-x86_64-compiler-rt  6) mingw-w64-clang-x86_64-crt-git
   7) mingw-w64-clang-x86_64-gcc-compat  8) mingw-w64-clang-x86_64-headers-git
   9) mingw-w64-clang-x86_64-libc++  10) mingw-w64-clang-x86_64-libmangle-git
   11) mingw-w64-clang-x86_64-libunwind  12) mingw-w64-clang-x86_64-libwinpthread-git
   13) mingw-w64-clang-x86_64-lld  14) mingw-w64-clang-x86_64-lldb  15) mingw-w64-clang-x86_64-llvm
   16) mingw-w64-clang-x86_64-llvm-libs  17) mingw-w64-clang-x86_64-make
   18) mingw-w64-clang-x86_64-openmp  19) mingw-w64-clang-x86_64-pkgconf
   20) mingw-w64-clang-x86_64-tools-git  21) mingw-w64-clang-x86_64-winpthreads-git
   22) mingw-w64-clang-x86_64-winstorecompat-git

Enter a selection (default=all): |
```

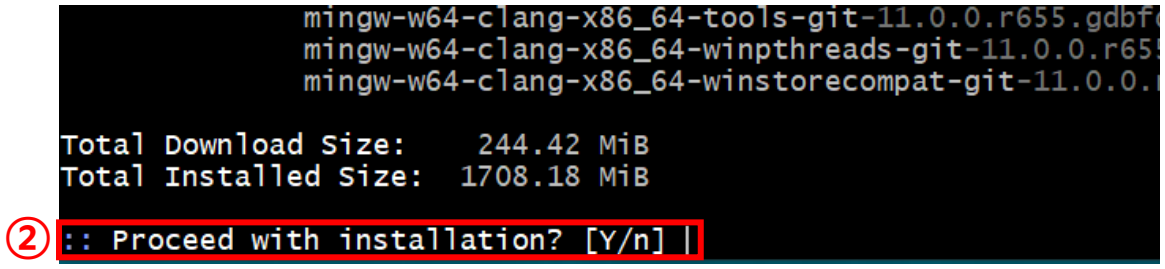
- pacman optional : -S –needed
  - -S : install package
  - --needed : only install it if is not installed.
- base-devel
  - MSYS2 package group name
- toolchain
  - Basically clang, llvm, etc… are built in.

MSYS2 Setup

8. **Press *Enter* for ① and write “Y” for ②.**

   - Download mingw-w64-x86-clang_64-toolchain package.

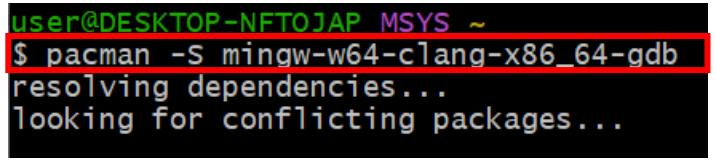




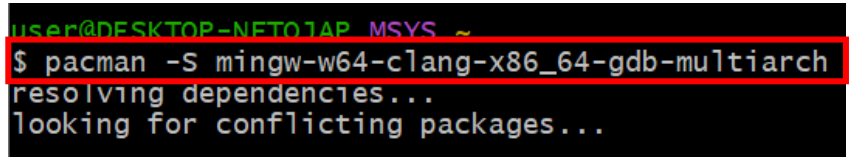
MSYS2 Setup

9. Write the command to install gdb package

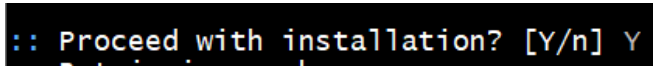
- **pacman -S mingw-w64-clang-x86_64-gdb**
  - GNU debugger(gdb) package



- **pacman -S mingw-w64-clang-x86_64-gdb-multiarch**
  - GNU debugger(gdb) support all targets package

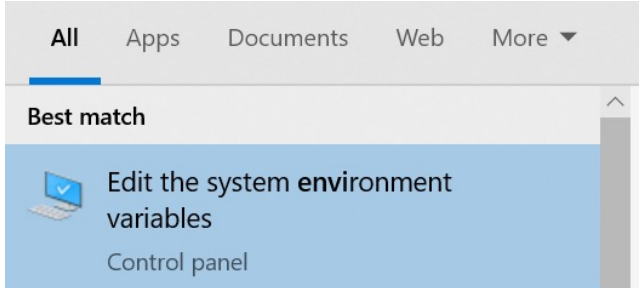


- Write ‘Y’ each in each installation process

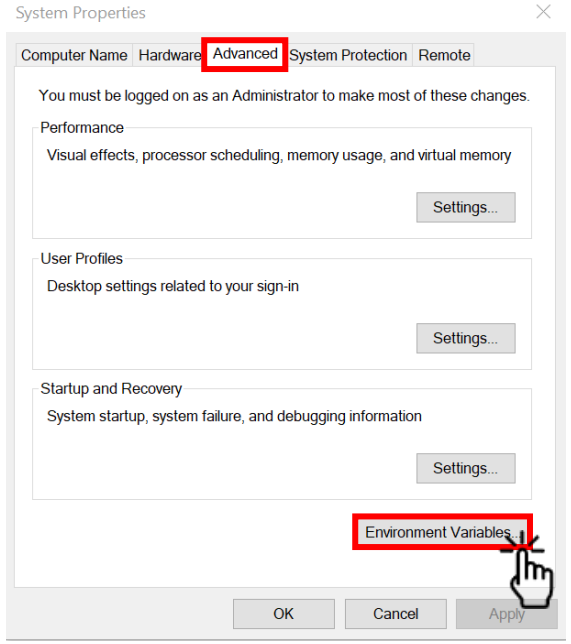


MSYS2 Setup

10. **Edit environment variables**
    - Select “Edit the system environment variables” section in the control panel.

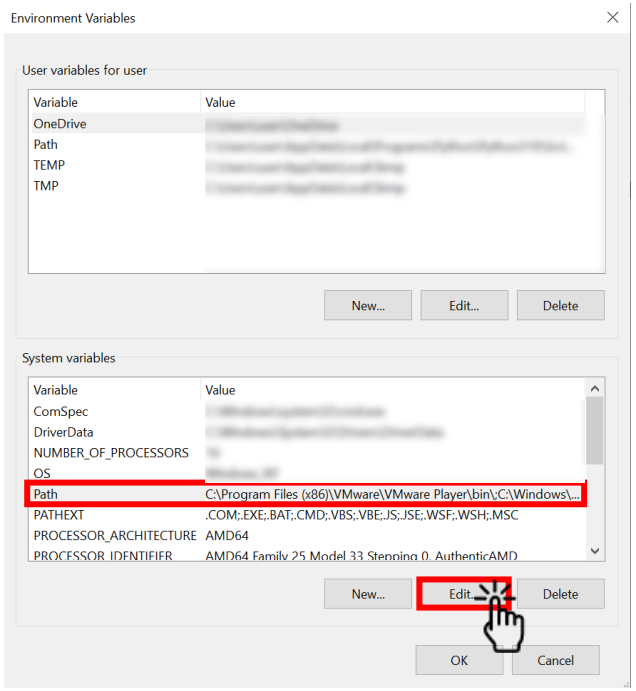


11. **Click “Environment Variables…” button in “Advanced” tab.**

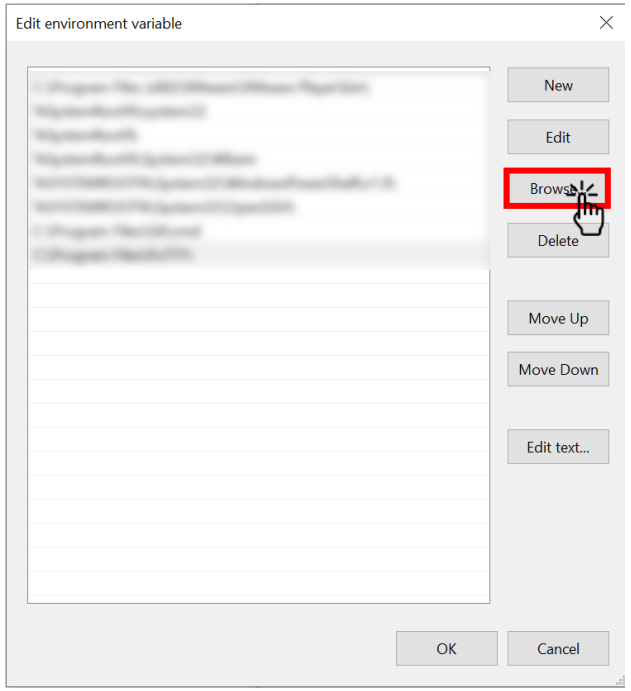


MSYS2 Setup

12. **Find and edit path of the system variable**
    - Find a variable “Path” in the system variables.
    - Click “Edit” button.

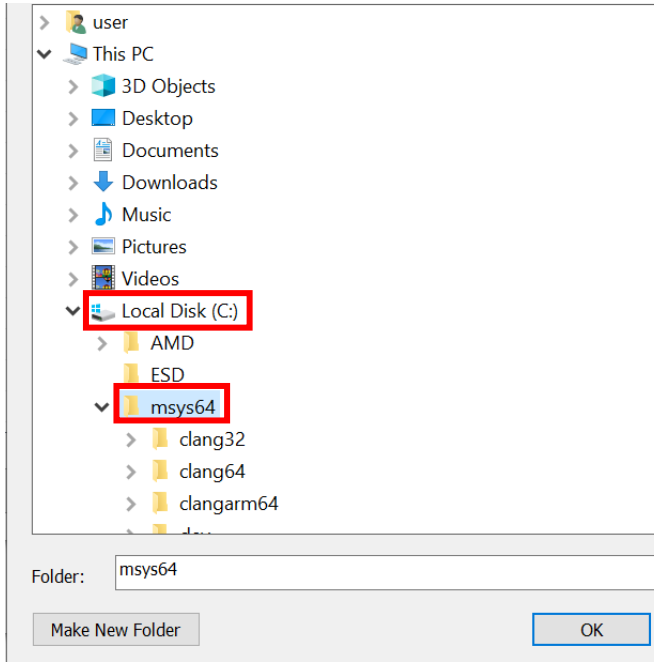


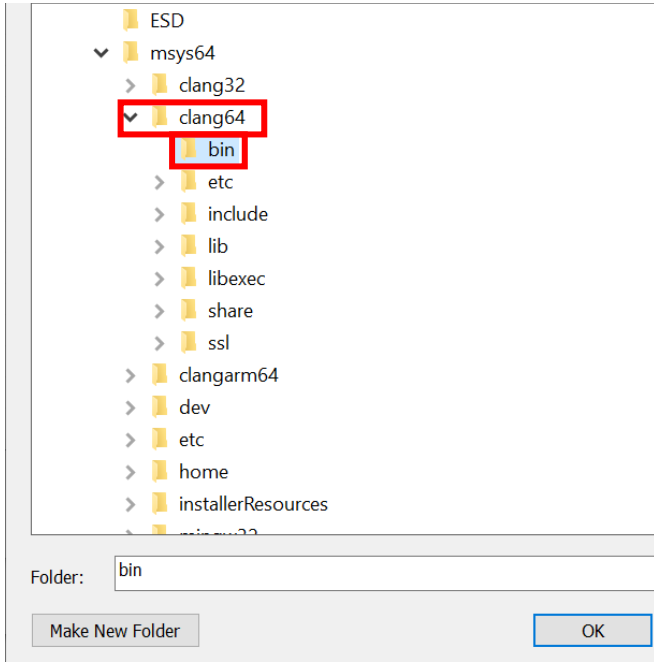
13. **Click “Browse” button.**



MSYS2 Setup

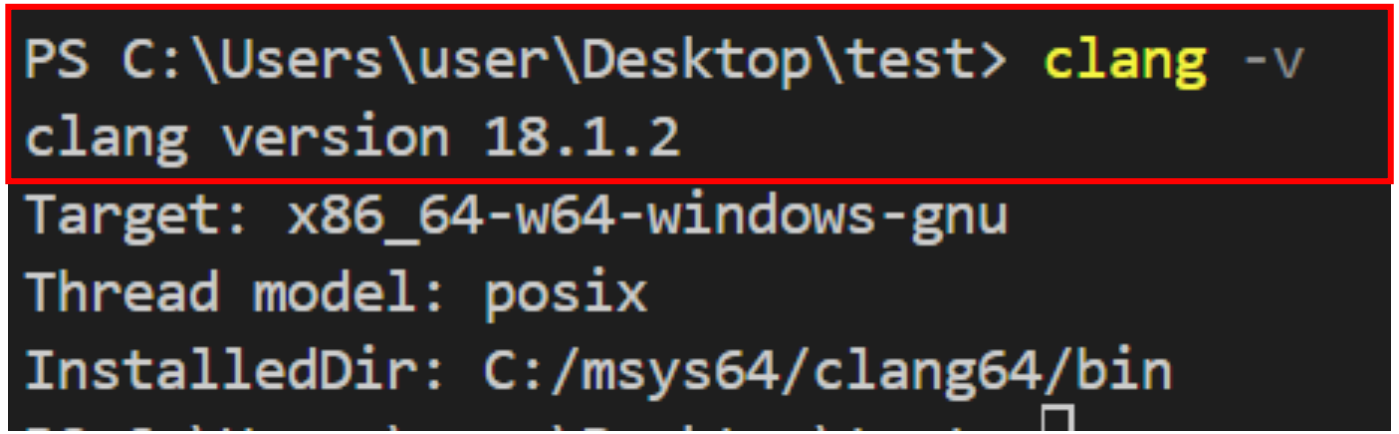
14. Select “C:\msys64\clang64\bin\” folder.





MSYS2 Setup

- **Check if your setups are valid**
  - Open **VSCode** and press " *Ctrl* + *Shift* + ` " to open new terminal window.
  - Write "**clang –v**" in the terminal below.
  - Your Clang version will be shown if your settings are valid.



How to install C/C++ Compiler

macOS

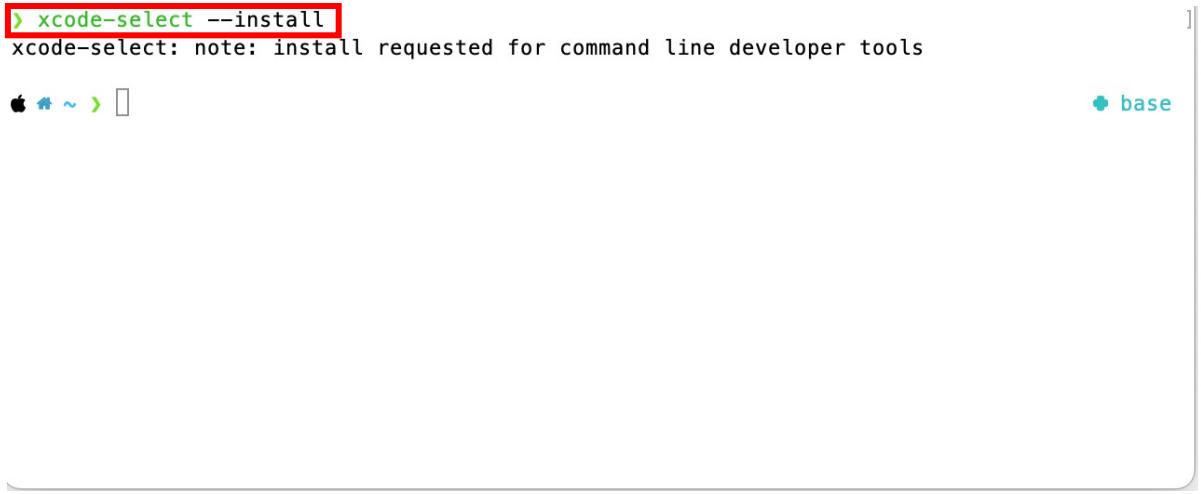
What is Xcode?

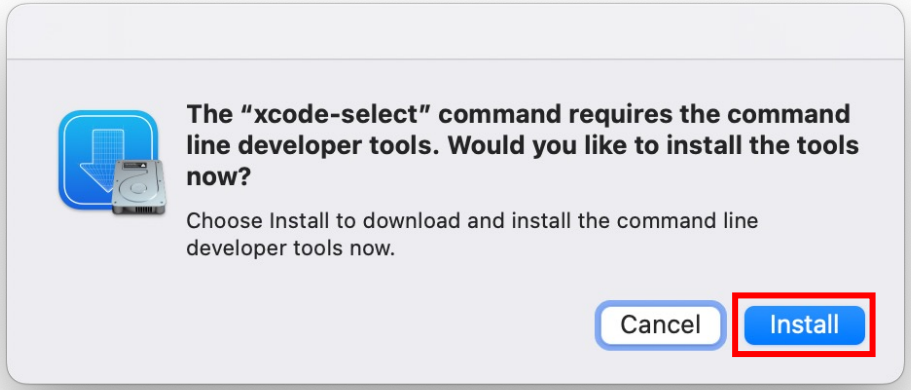
Xcode

- Apple’s IDE for software development.
- Includes various programming modules for source compilation.
- Includes the **Clang**/LLVM compilers and debugger.

Xcode Installation & Setup

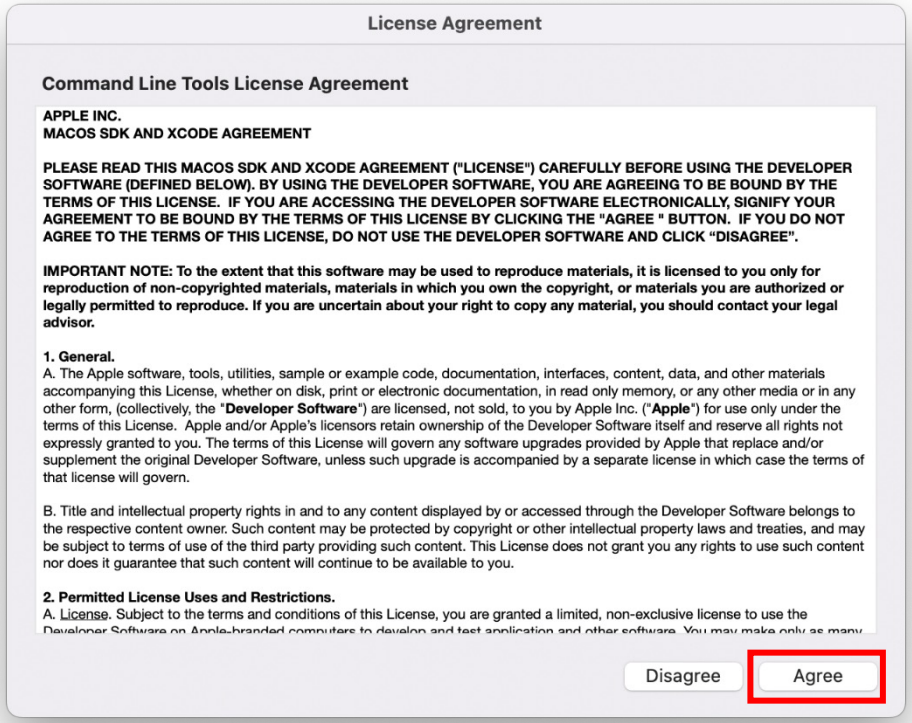
- **Install with Xcode Command Line Tools**
  - Open the Terminal app and write “**xcode-select --install**”.
  - Select install

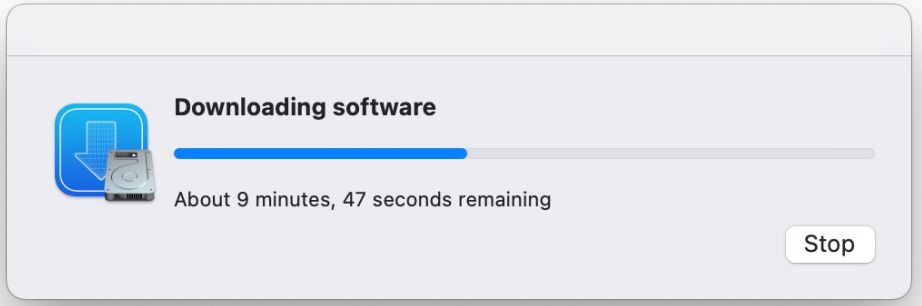


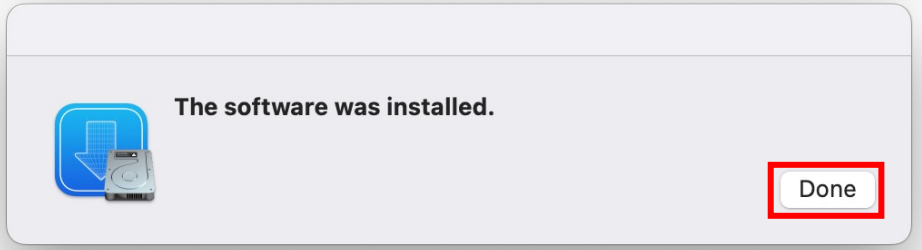


Xcode Installation & Setup

- **Install with Xcode Command Line Tools**
  - Click “Agree” button for the license agreement and wait for finishing installation.







Xcode Installation & Setup

- **Check if you have valid clang installed**
  - Enter “clang -v” to check the clang is available.
  - Outputs may look different depending on your mac’s CPU (Apple silicon / Intel).

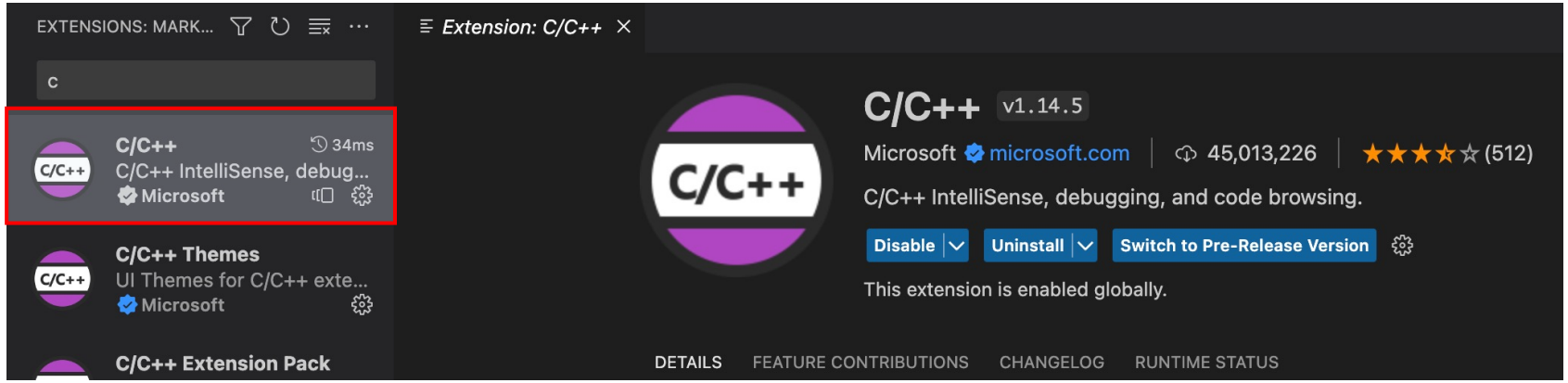
```
❯ clang -v
Apple clang version 15.0.0 (clang-1500.3.9.4)
Target: arm64-apple-darwin23.3.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
```

VS Code Setup

Windows & macOS

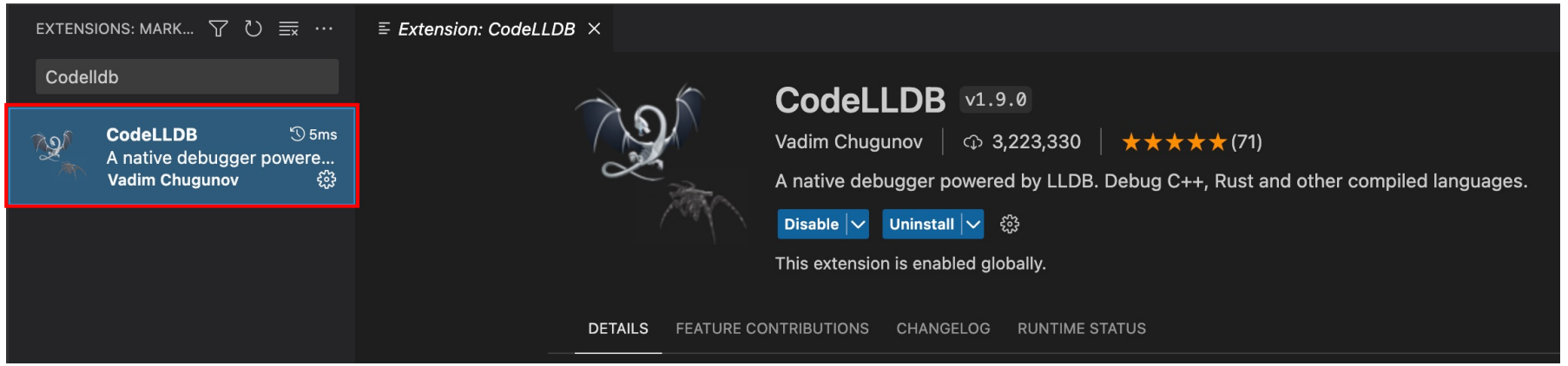
VS Code Setup

- **Install C/C++ extension**
  - Tap extensions menu and search ‘C’ in the bar.
  - Select the first “C/C++” extension and click “Install” button.



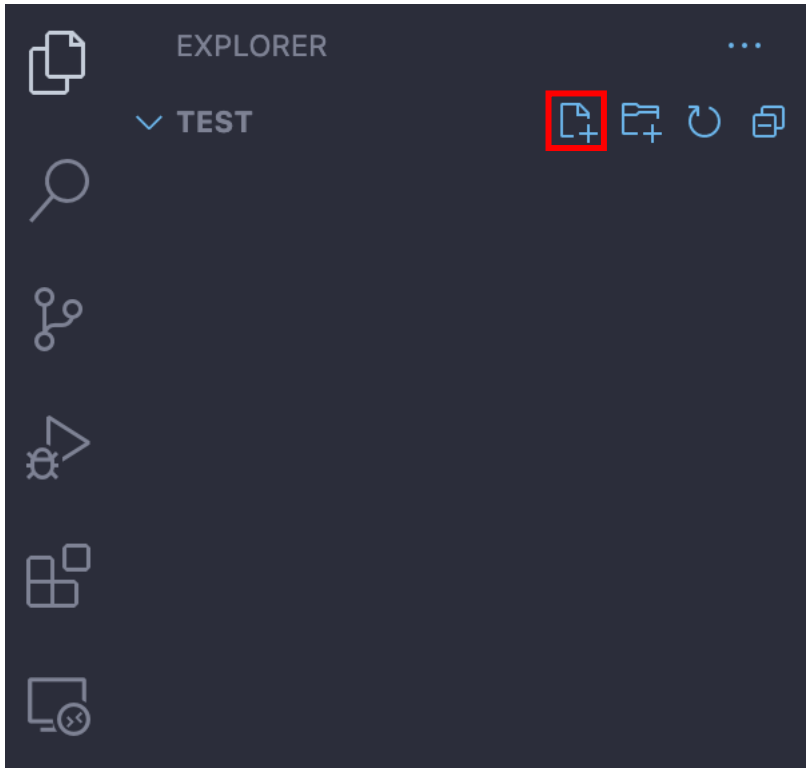
VS Code Setup

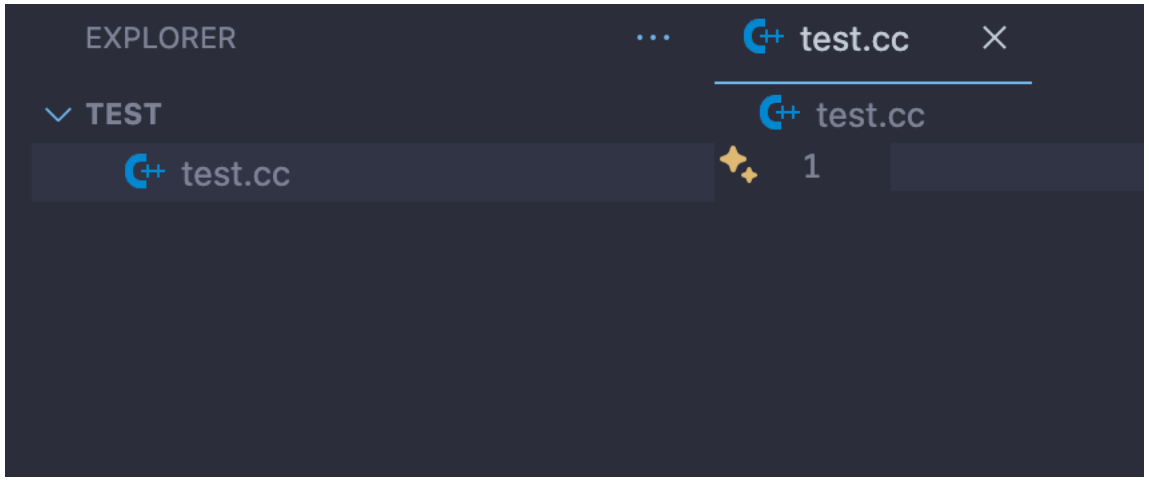
- **Install ‘CodeLLDB’ extension – Only MacOS**
  - Tap extensions menu and search ‘CodeLLDB’ in the bar.
  - Select the “CodeLLDB” extension and click “Install” button.



VS Code Setup

- **Create '*test.cc*' file**





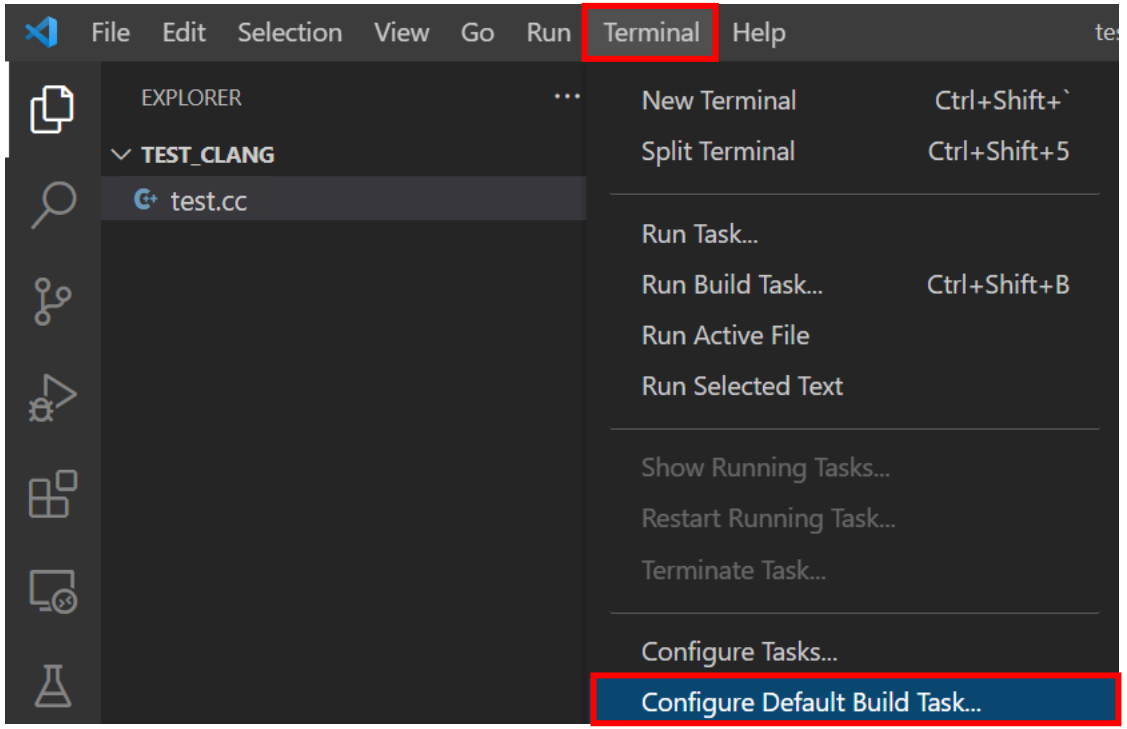
VS Code Setup

- **Write a simple C++ codes in the file**


```cpp
#include <iostream>

int main() {
    std::cout << "Hello!" << std::endl;
    return 0;
}
```

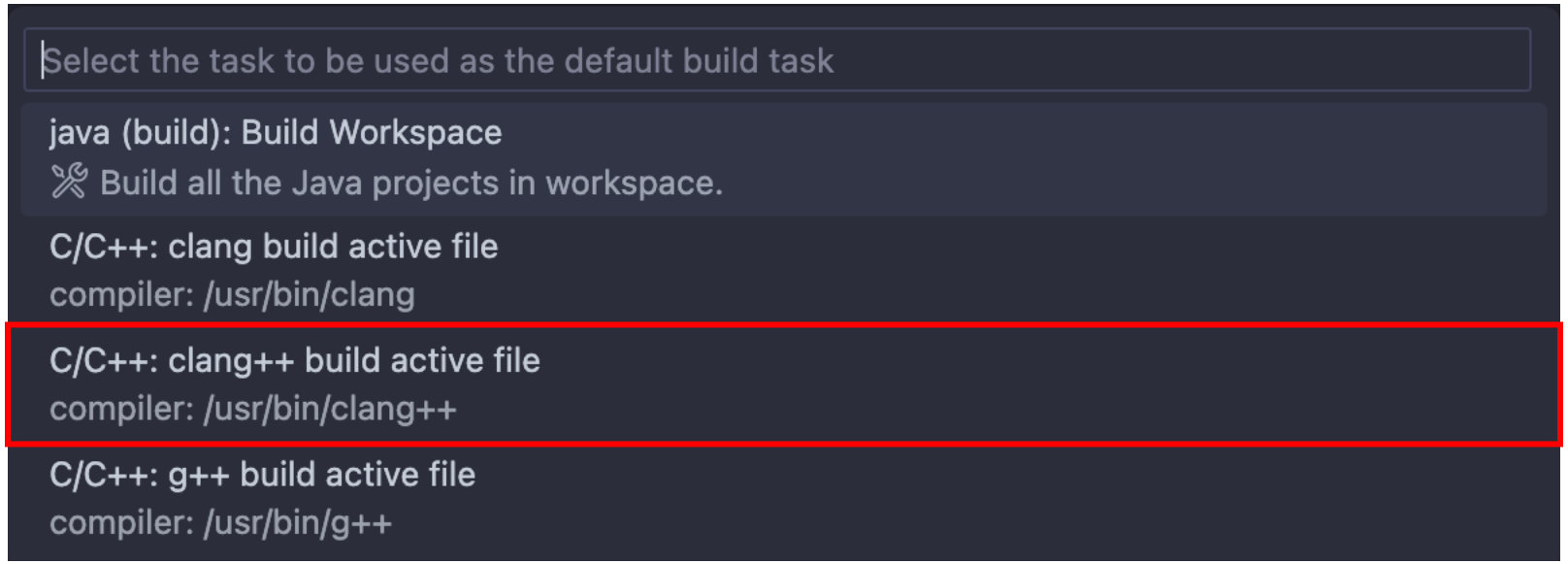
VS Code Setup

- **Click ‘Terminal > Configure Default Build Task’**



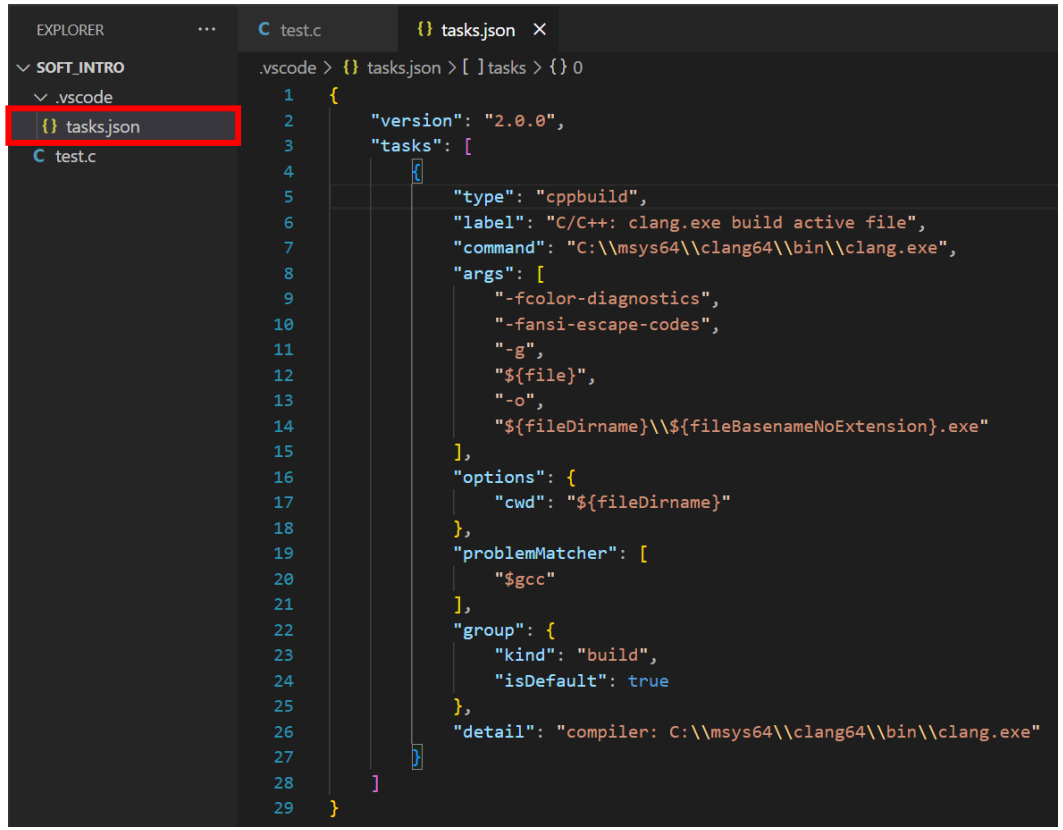
VS Code Setup – Build Source

- **Select “C/C++: clang++ build active file”**



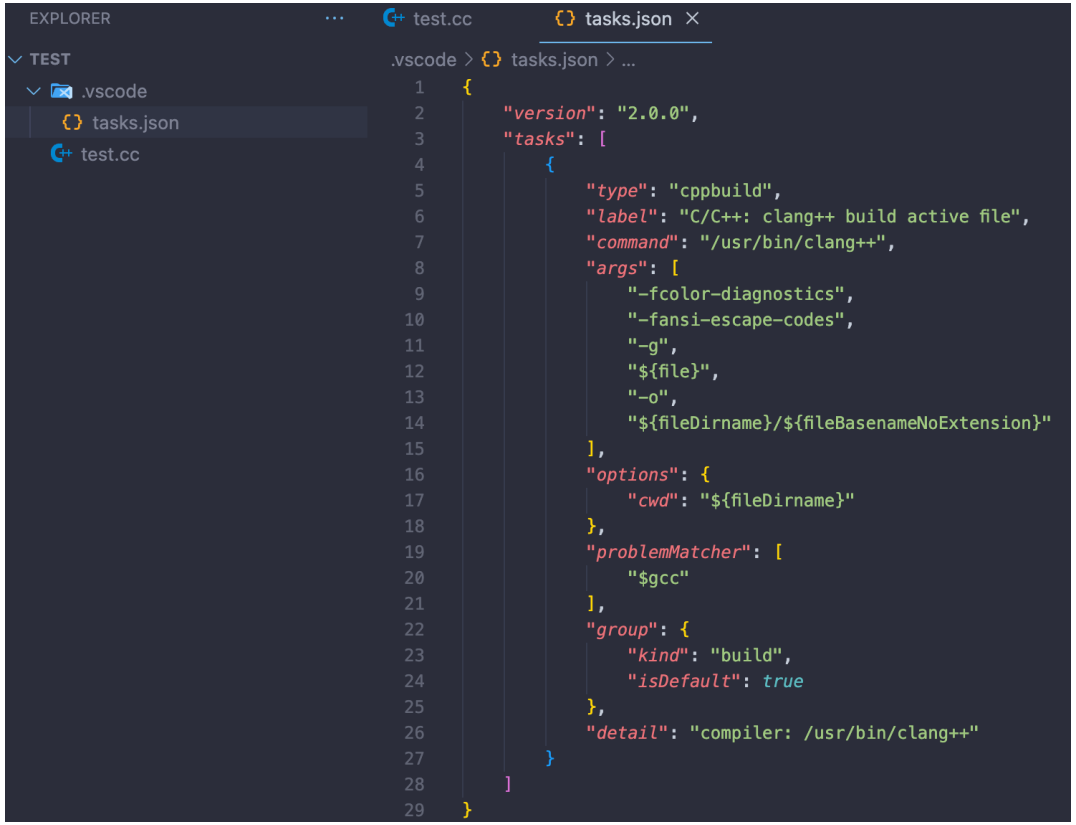
VS Code Setup – Build Source (Windows)

- Create '*tasks.json*' file in '*.vscode*' folder



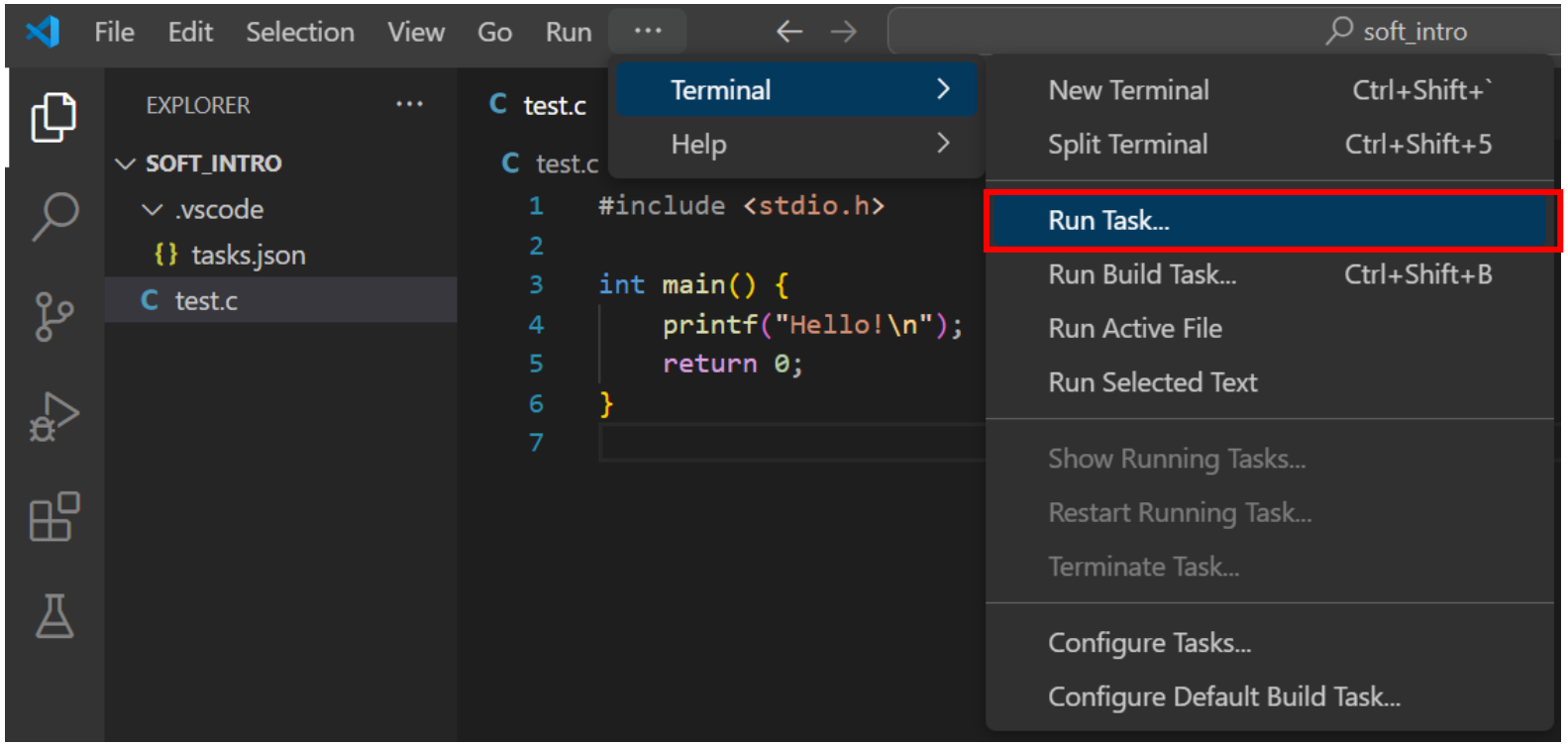
VS Code Setup – Build Source (macOS)

- **Create '*tasks.json*' file in '*.vscode*' folder**



VS Code Setup – Build Source

- **Click Terminal > Run Task…**



VS Code Setup – Build Source

- **Check build result in the terminal window**

Windows

```
 *  Executing task: C/C++: clang.exe build active file

Starting build...
cmd /c chcp 65001>nul && C:\msys64\clang64\bin\clang.exe -fcolor-diagnostics -fansi-escape-codes -g C:\Users\user\Desktop\t
est\soft_intro\test.c -o C:\Users\user\Desktop\test\soft_intro\test.exe

Build finished successfully.
 *  Terminal will be reused by tasks, press any key to close it.
```

macOS

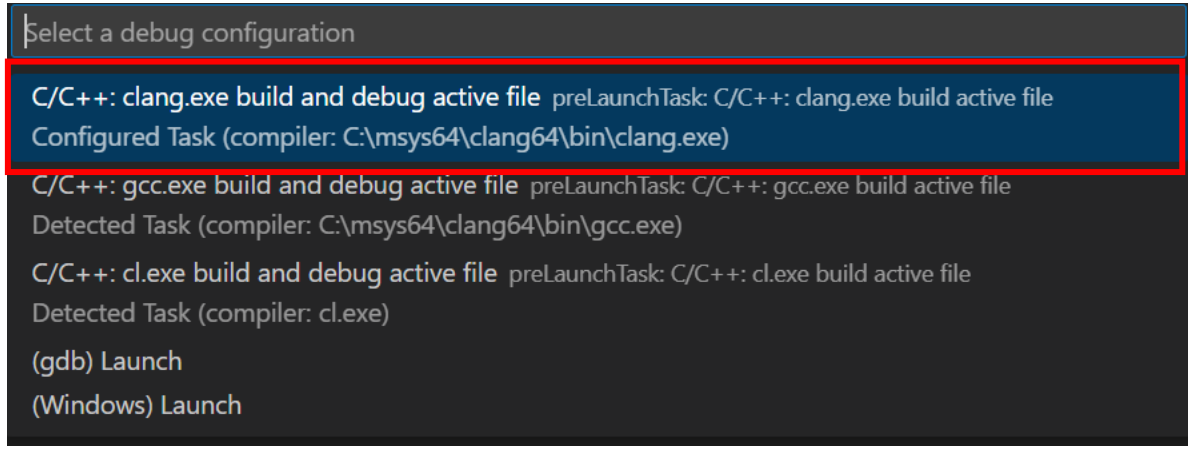
```
 *  Executing task: C/C++: clang++ build active file

Starting build...
/usr/bin/clang++ -std=gnu++14 -fcolor-diagnostics -fansi-escape-codes -g /Users/kevinbang/Desktop/test.cc -o /Users/kevinbang/Desktop/test

Build finished successfully.
 *  Terminal will be reused by tasks, press any key to close it.
```

VS Code Setup – Execution & Debugging

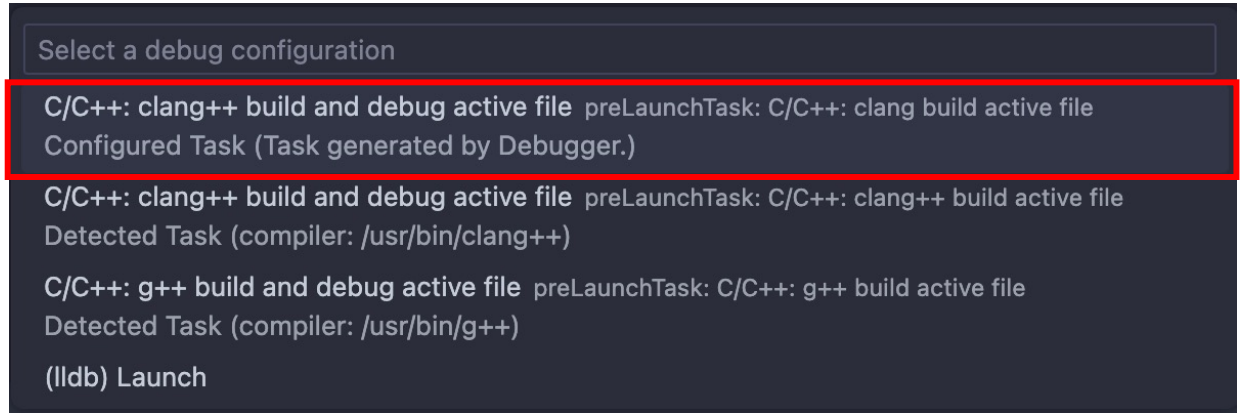
- **Open ‘*test.cc*’ file and Click ‘Add Debug configuration’ icon**

VS Code Setup – Execution & Debugging

- **Select a configuration**
  - **“C/C++: clang++(.exe) build and debug active file ….”**



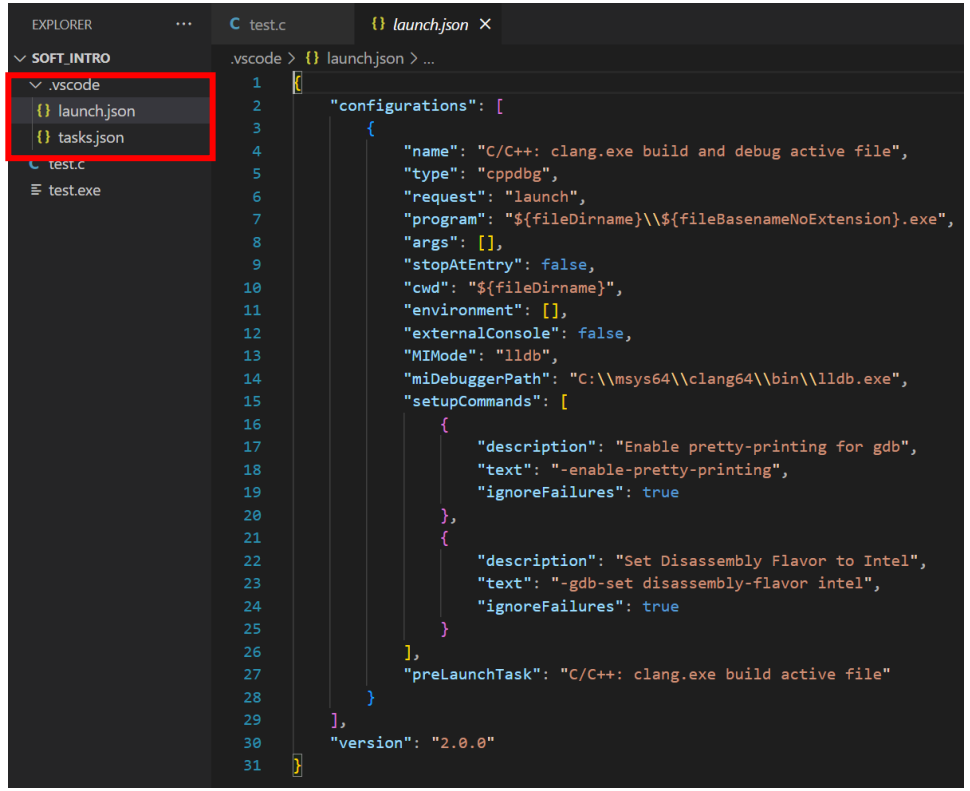
Windows



macOS

VS Code Setup – Execution & Debugging (Windows)

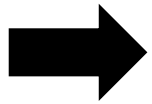
- Create '*launch.json*' file in '*.vscode*' folder



VS Code Setup – Execution & Debugging (Windows)

- **Open the '*launch.json*' file**
  - **replace the '*lldb*' with '*gbd*' in the '*launch.json*' file**
  - **If it is written as '*gdb*', there is no need to change it.**

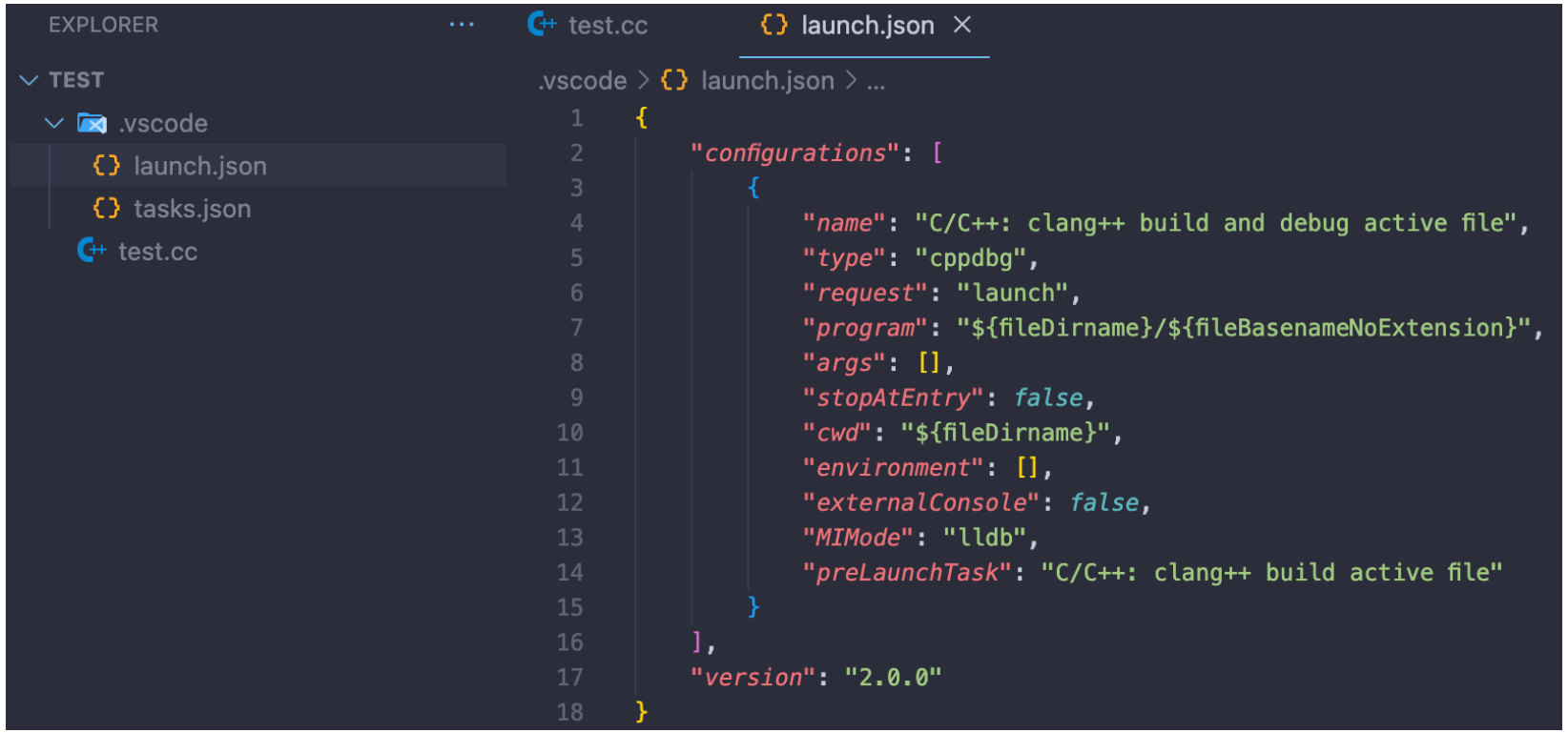
```
{
    "configurations": [
        {
            "name": "C/C++: clang.exe build and debug active file",
            "type": "cppdbg",
            "request": "launch",
          ...
            "MIMode": “lldb",
            "miDebuggerPath": "C:\\msys64\\clang64\\bin\\lldb.exe",
            "setupCommands": [
                ...
            ],
            "preLaunchTask": "C/C++: clang.exe build active file"
        }
    ],
    "version": "2.0.0"
}
```



```
{
    "configurations": [
        {
            "name": "C/C++: clang.exe build and debug active file",
            "type": "cppdbg",
            "request": "launch",
          ...
            "MIMode": "gdb",
            "miDebuggerPath": "C:\\msys64\\clang64\\bin\\gdb.exe",
            "setupCommands": [
                ...
            ],
            "preLaunchTask": "C/C++: clang.exe build active file"
        }
    ],
    "version": "2.0.0"
}
```

VS Code Setup – Execution & Debugging (macOS)

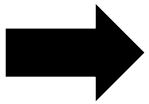
- Create '*launch.json*' file in '*.vscode*' folder



VS Code Setup – Execution & Debugging (macOS)

- **Open the '*launch.json*' file**
  - **Replace the '*cppdbg*' with '*lldb*' in the '*launch.json*' file**
  - **Delete '*stopAtEntry*' , '*externalConsole*', '*environment*' and '*MIMode*'**

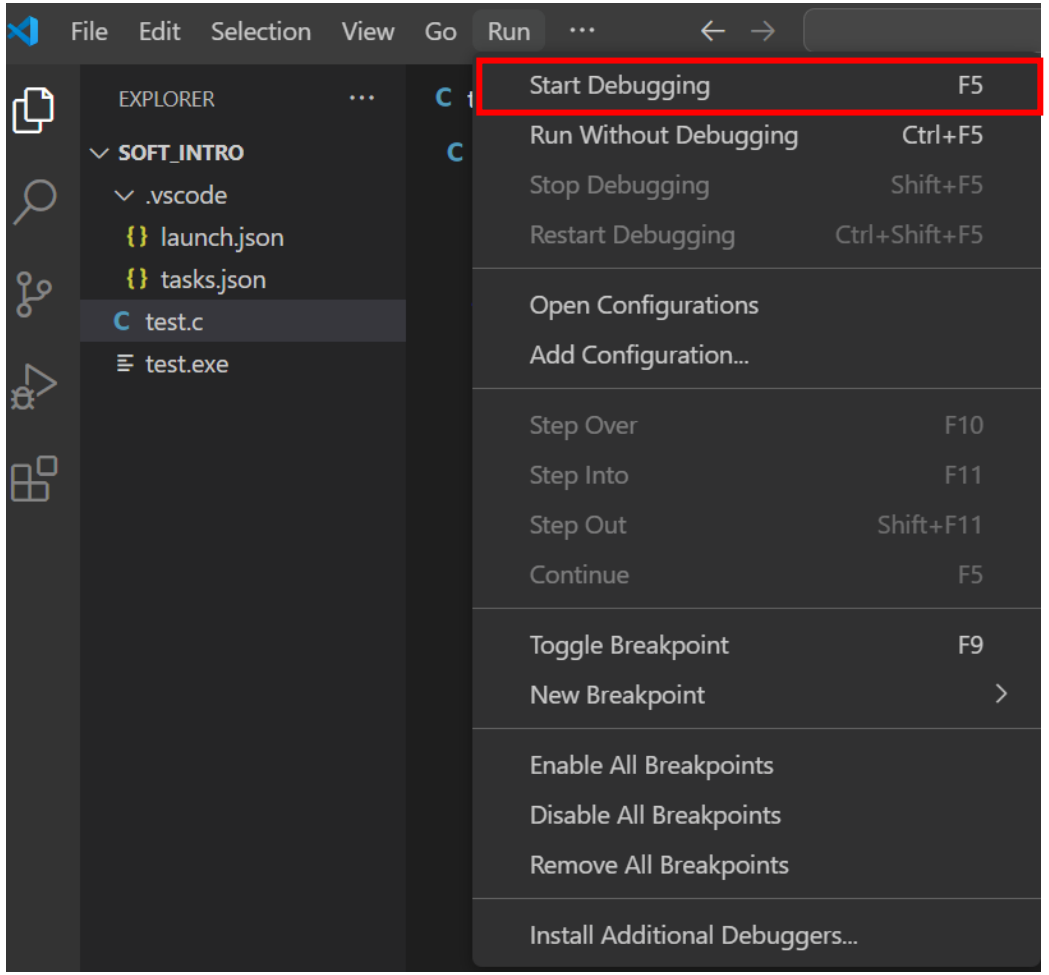
```
{
    "configurations": [
        {
            "name": "C/C++: clang++ build and debug active file",
            "type": "cppdbg",
            "request": "launch",
            "program": "${fileDirname}/${fileBasenameNoExtension}",
            "args": [],
            "stopAtEntry": false,
            "cwd": "${fileDirname}",
            "environment": [],
            "externalConsole": false,
            "MIMode": "lldb",
            "preLaunchTask": "C/C++: clang++ build active file"
        }
    ],
    "version": "2.0.0"
}
```



```
{
    "configurations": [
        {
            "name": "C/C++: clang++ build and debug active file",
            "type": "lldb",
            "request": "launch",
            "program": "${fileDirname}/${fileBasenameNoExtension}",
            "args": [],
            "cwd": "${fileDirname}",
            "preLaunchTask": "C/C++: clang++ build active file",
        }
    ],
    "version": "2.0.0"
}
```

VS Code Setup – Execution & Debugging

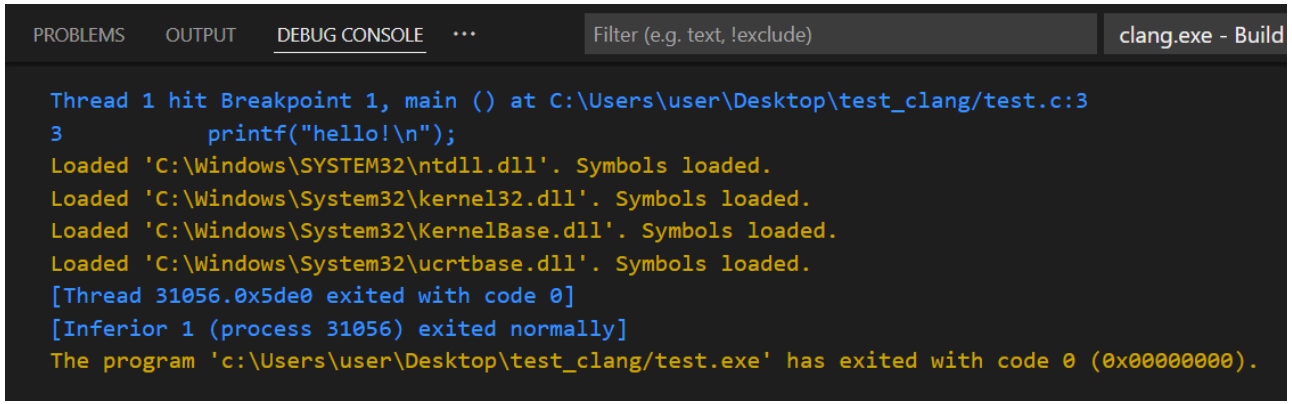
- **Open *test.c* file and press '*F5*' or click 'Run > Start Debugging'**



VS Code Setup – Execution & Debugging (Windows)

- **Complete debugging *test.cc* file**

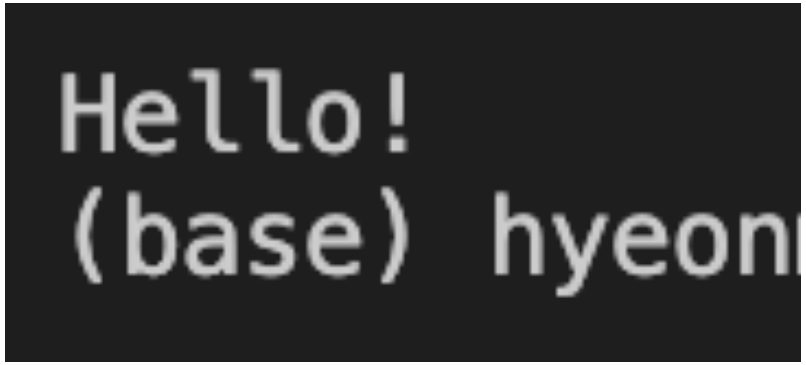
```
PS C:\Users\user\Desktop\test\soft_intro>  & 'c:\Users\user\.vscode\extensions\ms-vscode.cpptools-1.19.9-win32-x64\debugAda
pters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-c1az4lmx.5pp' '--stdout=Microsoft-MIEngine-Out-i3tte2cg.
t53' '--stderr=Microsoft-MIEngine-Error-rvk4p2fj.syq' '--pid=Microsoft-MIEngine-Pid-jmgjq5kg.w24' '--dbgExe=C:\msys64\clang
64\bin\gdb.exe' '--interpreter=mi'
Hello!
```



VS Code Setup – Execution & Debugging (macOS)

- **Complete debugging *test.cc* file**

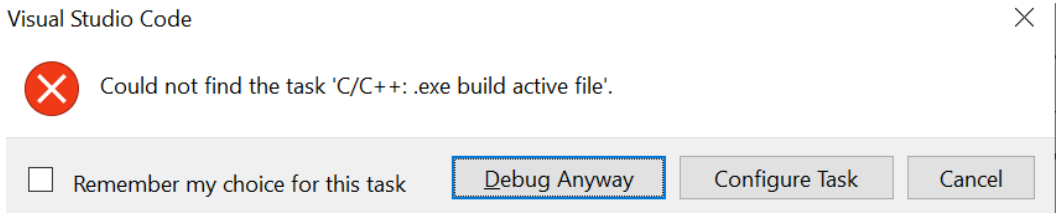
```
 *  Executing task: C/C++: clang++ build active file

Starting build...
/usr/bin/clang++ -std=gnu++14 -fcolor-diagnostics -fansi-escape-codes -g /Users/kevinbang/Desktop/test/test.cc -o /Users/kevinbang/Desktop/test/test

Build finished successfully.
 *  Terminal will be reused by tasks, press any key to close it.
```



Resolve the environment set up errors

Error related to “could not find file”

- **If you get an error like the picture below when you execute C code file, follow the solution below.**



- **Solution**
  - Check the ‘***preLaunchTask***’ part of the ‘***launch.json***’ file.
  - If ‘***preLaunchTask***’ is not the same as the ‘***label***’ in the ‘***task.json***’ file, change it to be the same.

Launch.json

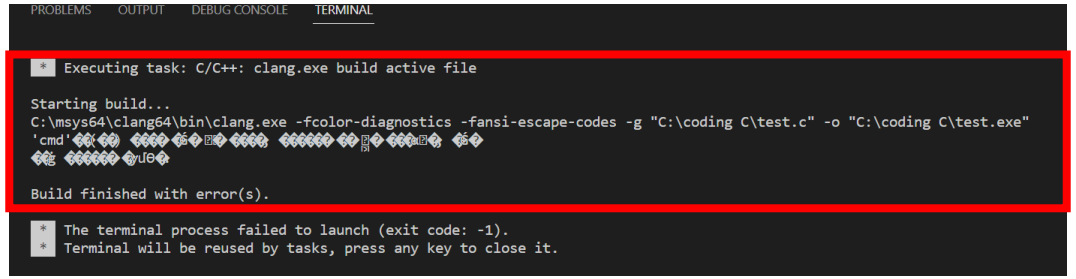
```
{
    "configurations": [
        {
            "name": "C/C++: clang.exe build and debug active file",
            "type": "cppdbg",
            "request": "launch",
          ...
            "preLaunchTask": "C/C++: clang.exe build active file"
        }
    ],
    "version": "2.0.0"
}
```

task.json

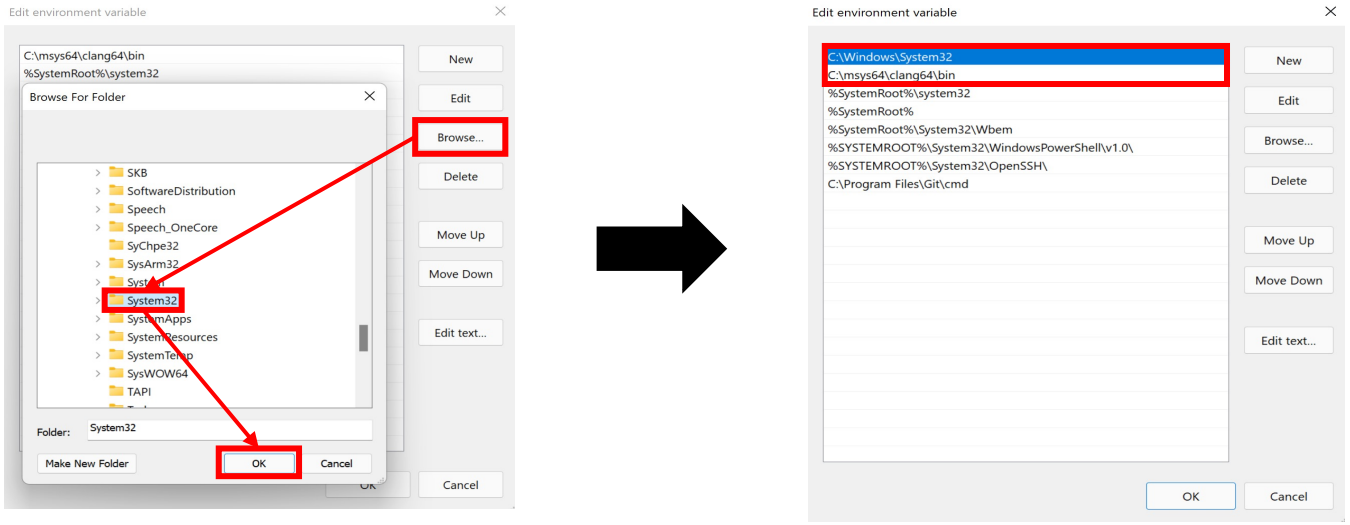
```
{
    "version": "2.0.0",
    "tasks": [
        {
            "type": "cppbuild",
            "label": "C/C++: clang.exe build active file",
            "command": "C:\\msys64\\clang64\\bin\\clang.exe",
            "args": [
            ...
    ]
}
```

Error related to “cmd”

- **If you get an error like the picture below when you execute ‘run task’, follow the solution below.**



- **Solution**
  - Add a path “**C:\Windows\System32**” to the PATH environment variable.



Error related to “unable to establish a connection to LLDB”

- **When you execute ‘start debugging’(= Click ‘F5’) an endless error as shown in the ‘Figure 1’ and then you press ‘Ctrl +C’ on the terminal an error message as shown in the Figure 2.**

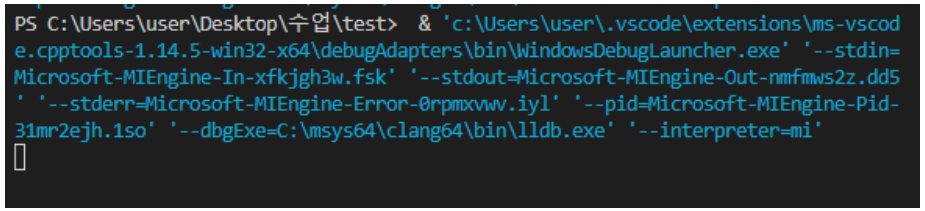


Figure 1

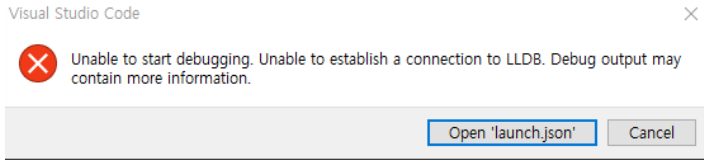
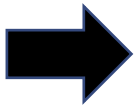


Figure 2

- **Solution**
  - Replace the ‘lldb’ with ‘gdb’ in the launch.json file.

```
//launch.json file
"configurations": [
        {
            ...
            "MIMode": "lldb",
            "miDebuggerPath": "C:\\msys64\\clang64\\bin\\lldb.exe",
            ....
            "preLaunchTask": "C/C++: clang.exe build active file"
        }
    ],..}
```

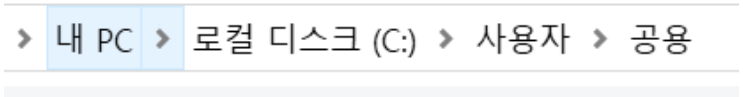


```
//launch.json file
"configurations": [
        {
            ...
            "MIMode": “gdb",
            "miDebuggerPath": "C:\\msys64\\clang64\\bin\\gdb.exe",
            ....
            "preLaunchTask": "C/C++: clang.exe build active file"
        }
    ],..}
```

Error related to “illegal byte sequential”

- **If you get an error like the picture below when you execute ‘start debugging’(press ‘F5’), follow the solution below.**

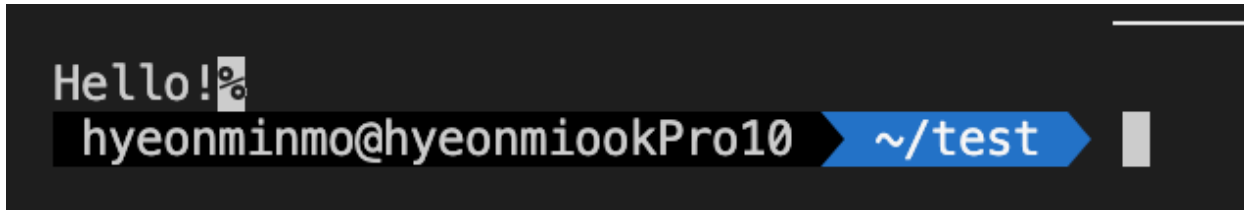
- **Solution**
  - Please make sure that there is no any letter which is not an English including whitespace
  - Move your current project folder to another path without Korean language.
    - Tip. Path of the following picture is equivalent to ‘C:\Users\Public’



FAQ

How to remove “%” at the end of a terminal output

- **If you the result of executing the code is ‘%’ as shown in the picture below**



- **Solution**
  - **Command +Shift+p (or view> command pallete)**
  - **search for “*Terminal: default Profile*”**
  - **select “*bash*”**