



南方科技大学  
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

# Advanced Programming

## Lab 1, C/C++ environment configuration

廖琪梅, 王大兴, 于仕琪



# Topics

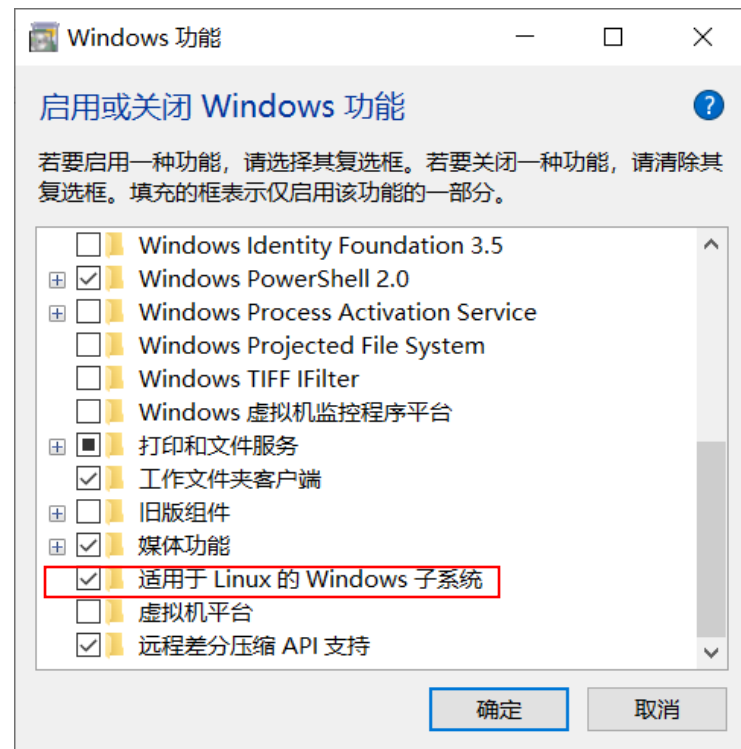
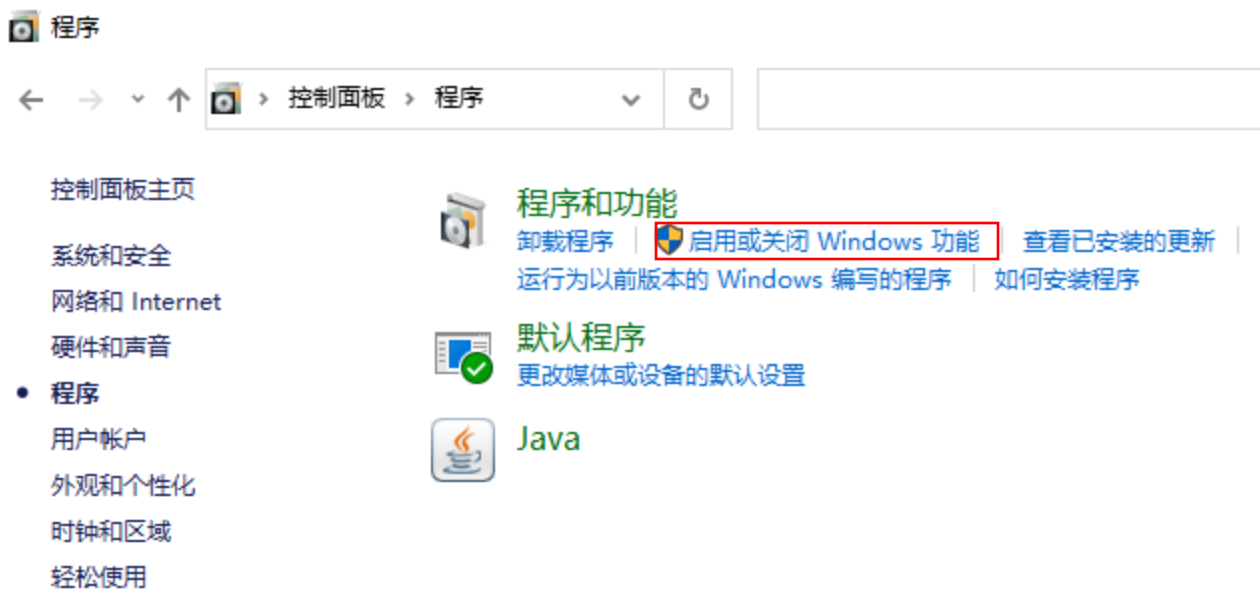
- 1. Download and install GCC on Windows 10 (Based on Windows Subsystem for Linux)
- 2. Download and install LLVM on macOS(Optional, based on the OS)
- 3. Download and install the editor (VSCode)
- 4. Compile, link and run C/C++ programs
- 5. Set output format
- 6. Practices



# 1 Download and install GCC on Windows 10 (Based on WSL)

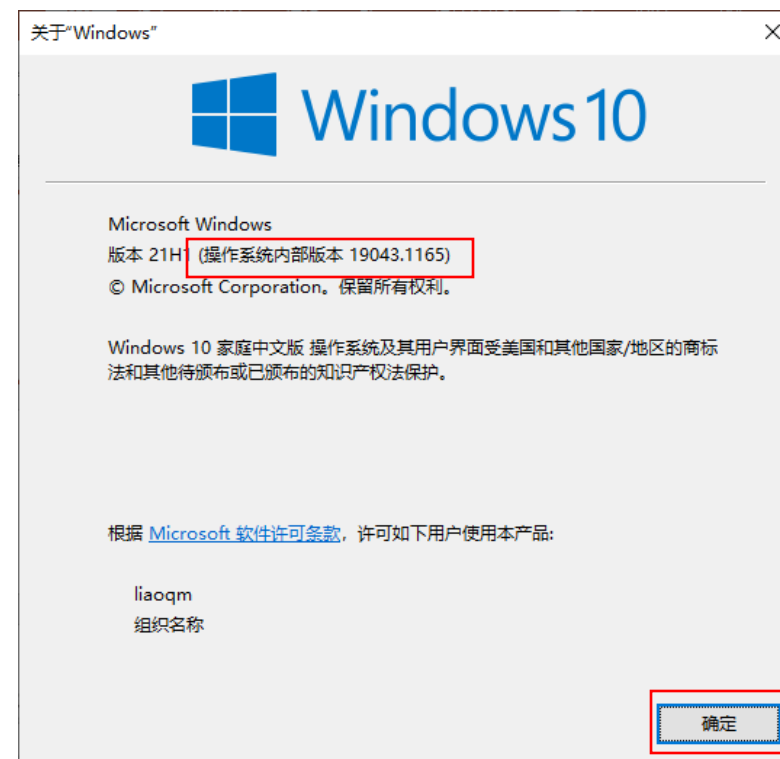
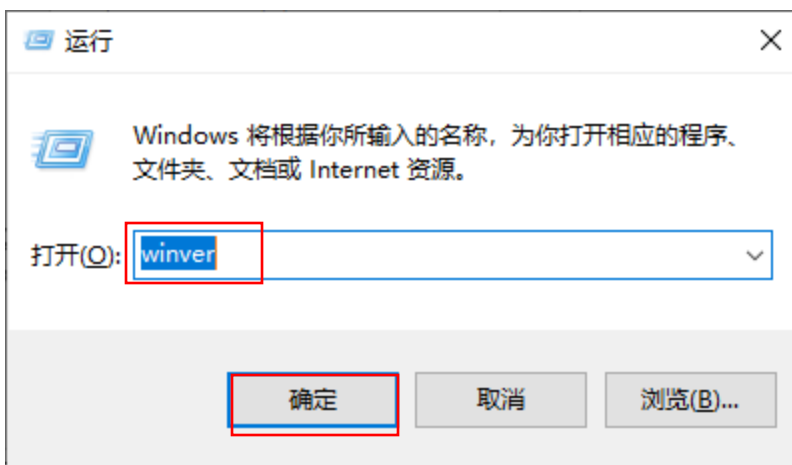
## 1.1 Install WSL on Windows system

- **Step one:** enable the **Windows Subsystem for Linux**
  - Open the Control Panel and set the Windows functions





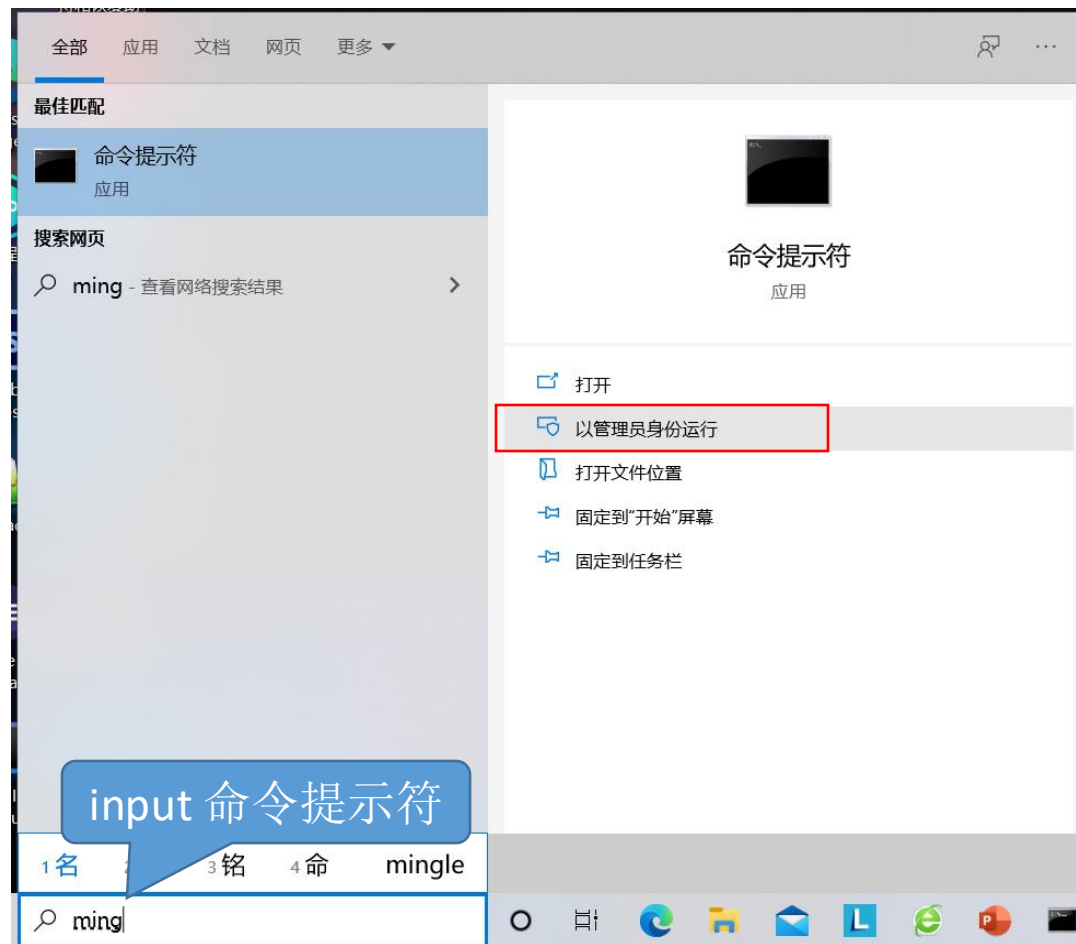
- **Step one:** enable the Windows Subsystem for Linux
  - Check Prerequisites: You must be running Windows 10 version 2004 and higher (Build 19041 and higher) or Windows 11.
  - To check your version and build number, select **Windows** logo key + **R**, type **winver**, select **OK**.



<https://learn.microsoft.com/en-us/windows/wsl/install>



- **Step one:** enable the Windows Subsystem for Linux
  - Open Powershell or Windows Command Prompt in administrator mode and enter the ***wsl --install*** command





```
Administrator: Command Promp
C:\Users\Craig>wsl --install
Installing: Virtual Machine Platform
Virtual Machine Platform has been installed.
Installing: Windows Subsystem for Linux
Windows Subsystem for Linux has been installed.
Downloading: Ubuntu
[===== ]
```

It usually downloads and installs Ubuntu in your computer.

```
C:\> 管理员: 命令提示符 - wsl --install
Microsoft Windows [版本 10.0.19045.3086]
(c) Microsoft Corporation。保留所有权利。

C:\WINDOWS\system32>wsl --install
正在安装: 虚拟机平台
已安装 虚拟机平台。
正在安装: 适用于 Linux 的 Windows 子系统
已安装 适用于 Linux 的 Windows 子系统。
正在安装: 适用于 Linux 的 Windows 子系统
```

If Ubuntu is not downloaded, terminate the current installation by **Ctrl+c** and restart your computer.



Use the command `wsl -l -o` to check which version of Ubuntu is valid to your system. Then use the command `wsl --install -d` with the name of Ubuntu. For example, `wsl --install -d Ubuntu-20.04`.

The screenshot shows a Windows Command Prompt window titled "管理员: 命令提示符". The user enters the command `wsl -l -o` at the prompt `C:\WINDOWS\system32>`. Below the command, there is a list of Linux distributions and their friendly names. A callout bubble points to the command, stating "List the Linux distributions in your computer." Another callout bubble points to the list of distributions, stating "List the valid name and version of Linux in your system." The user then enters the command `wsl --install -d Ubuntu-20.04` at the prompt. A third callout bubble points to this command, stating "Type the installation command to install Ubuntu 20.04."

```
C:\WINDOWS\system32>wsl -l -o
以下是可安装的有效分发的列表。
请使用“wsl --install -d <分发>”安装。

NAME                                FRIENDLY NAME
Ubuntu                              Ubuntu
Debian                              Debian GNU/Linux
kali-linux                          Kali Linux Rolling
Ubuntu-18.04                        Ubuntu 18.04 LTS
Ubuntu-20.04                        Ubuntu 20.04 LTS
Ubuntu-22.04                        Ubuntu 22.04 LTS
OracleLinux_7_9                     Oracle Linux 7.9
OracleLinux_8_7                     Oracle Linux 8.7
OracleLinux_9_1                     Oracle Linux 9.1
openSUSE-Leap-15.5                  openSUSE Leap 15.5
SUSE-Linux-Enterprise-Server-15-SP4 SUSE Linux Enterprise Server 15 SP4
SUSE-Linux-Enterprise-15-SP5        SUSE Linux Enterprise 15 SP5
openSUSE-Tumbleweed                 openSUSE Tumbleweed

C:\WINDOWS\system32>wsl --install -d Ubuntu-20.04
```



If you use command ***wsl --install*** and return the command instruction, it means that the default Ubuntu is not fit to your system(If you input the wrong command, it shows you the same window).

```
C:\WINDOWS\system32\cmd.exe
C:\Users\liaomq>wsl --install
版权所有 (c) Microsoft Corporation。保留所有权利。

用法: wsl.exe [Argument] [Options...] [CommandLine]

运行 Linux 二进制文件的参数:

    如果未提供命令行, wsl.exe 将启动默认 shell。

    --exec, -e <CommandLine>
        在不使用默认 Linux Shell 的情况下执行指定的命令。

    --
        按原样传递其余命令行。

选项:

    --cd <Directory>
        将指定目录设置为当前工作目录。
        如果使用了 /, 则将使用 Linux 用户的主页路径。如果路径
        以 / 字符开头, 将被解释为绝对 Linux 路径。
        否则, 该值一定是绝对 Windows 路径。

    --distribution, -d <Distro>
        运行指定分发。

    --user, -u <UserName>
        以指定用户身份运行。

管理适用于 Linux 的 Windows 子系统的参数:

    --help
        显示用法信息。

    --install [选项]
        安装额外的适用于 Linux 的 Windows 子系统分发。
        要获得有效分发列表, 请使用 "wsl --list --online"。

        选项:
            --distribution, -d [参数]
                按名称下载并安装分发。

            参数:
                有效分发名称(不区分大小写)。

            示例:
                wsl --install -d Ubuntu
                wsl --install --distribution Debian

    --set-default-version <Version>
        更改新分发的默认安装版本。

    --shutdown
        立即终止所有运行的分发及 WSL 2
        轻型实用工具虚拟机。

    --status
```

At this time, you need open PowerShell as administrator and run:

**dism.exe /online /enable-feature  
/featurename:VirtualMachinePlatform /all /norestart**

```
管理员: 命令提示符
(c) Microsoft Corporation。保留所有权利。

C:\WINDOWS\system32>dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart

部署映像服务和管理工具
版本: 10.0.19041.844

映像版本: 10.0.19043.1165

启用一个或多个功能
[=====100.0%=====]
操作成功完成。

C:\WINDOWS\system32>
```

Copy the command and run

**Restart** your computer, open the **Microsoft Store**, search for your preferred Linux distribution (Ubuntu), get and install it in your computer according to the guidance.





```
liao@DESKTOP-OOC4F37: ~  
Installing, this may take a few minutes...  
Please create a default UNIX user account. The username does not need to match your Windows username.  
For more information visit: https://aka.ms  
Enter new UNIX username: liao  
New password:  
Retype new password:  
passwd: password updated successfully  
Installation successful!  
适用于 Linux 的 Windows 子系统现已在 Microsoft Store 中可用:  
你可以通过运行“wsl.exe --update”或通过访问 https://aka.ms/wslstorepage 进行升级  
从 Microsoft Store 安装 WSL 将可以更快地获取最新的 WSL 更新。  
有关详细信息, 请访问 https://aka.ms/wslstoreinfo  
  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 4.4.0-19041-Microsoft x86_64)  
  
* Documentation: https://help.ubuntu.com  
* Management: https://landscape.canonical.com  
* Support: https://ubuntu.com/advantage  
  
System information as of Mon Sep  4 13:13:33 CST 2023  
  
System load:   0.52      Processes:            7  
Usage of /home: unknown  Users logged in:      0  
Memory usage:  36%      IPv4 address for eth0: 10.16.75.223  
Swap usage:    0%       IPv6 address for eth0: 2001:da8:201d:1107::1239  
  
Expanded Security Maintenance for Applications is not enabled.  
  
0 updates can be applied immediately.  
  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update  
  
This message is shown once a day. To disable it please create the  
/home/liao/.hushlogin file.  
liao@DESKTOP-OOC4F37: $
```

Input new UNIX username and new password.  
Remember your username and password.  
**Note:** the password is not displayed on the screen.

Linux command prompt



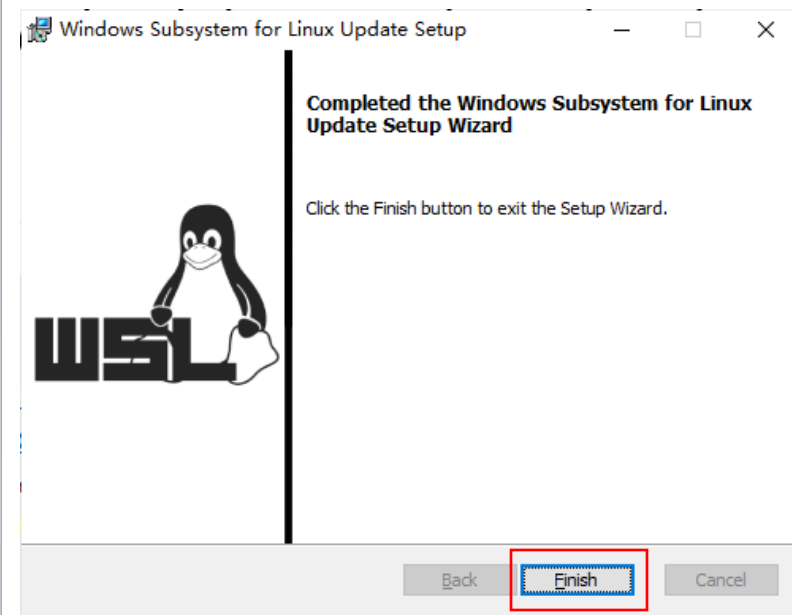
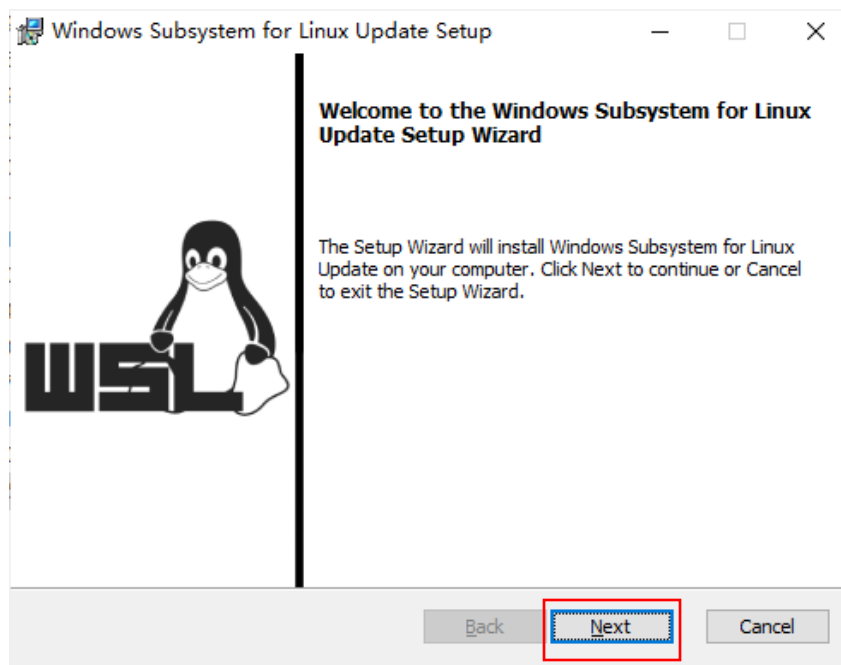
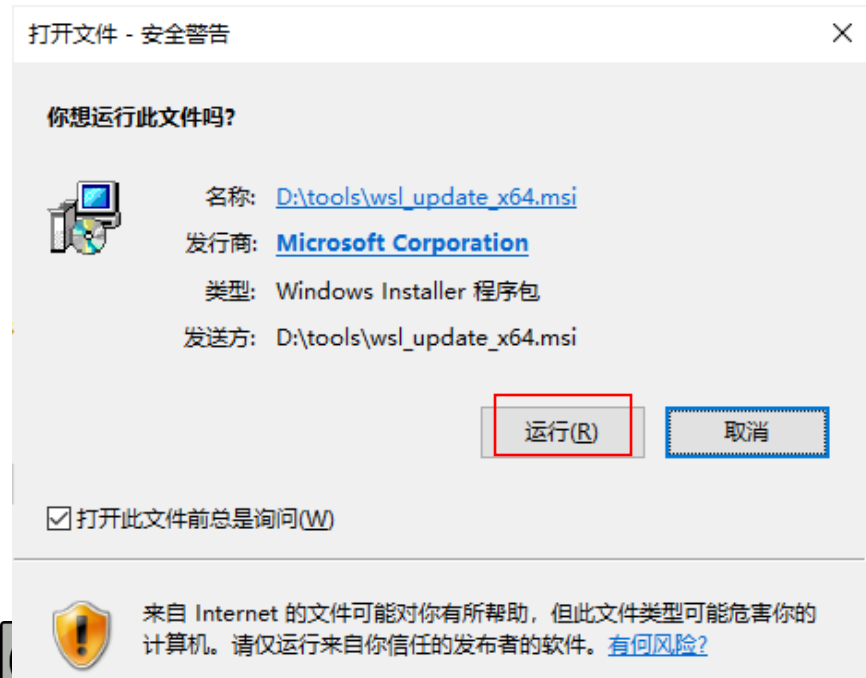
# 1.1 Install WSL on Windows 10(cont.)

- **Step two:** Update WSL kernel
- Download the latest package and run the update package

[https://wslstorestorage.blob.core.windows.net/wslblob/wsl\\_update\\_x64.msi](https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi)

If you're using an ARM64 machine, please download the **ARM64 package** instead.

[https://wslstorestorage.blob.core.windows.net/wslblob/wsl\\_update\\_arm64.msi](https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_arm64.msi)





# 1.1 Install WSL on Windows 10(cont.)

- **Step three:** Set WSL version as 2
- Open PowerShell or Windows command prompt and run this command to set WSL version as 2 : `wsl --set-version Ubuntu-20.04 2`

```
管理员: 命令提示符
Microsoft Windows [版本 10.0.19045.3208]
(c) Microsoft Corporation。保留所有权利。

C:\WINDOWS\system32>wsl -l -v
  NAME                STATE      VERSION
* Ubuntu-20.04        Stopped    1

C:\WINDOWS\system32>wsl --set-version Ubuntu-20.04 2
正在进行转换, 这可能需要几分钟时间...
有关与 WSL 2 的主要区别的信息, 请访问 https://aka.ms/ws12
转换完成。

C:\WINDOWS\system32>wsl -l -v
  NAME                STATE      VERSION
* Ubuntu-20.04        Stopped    2

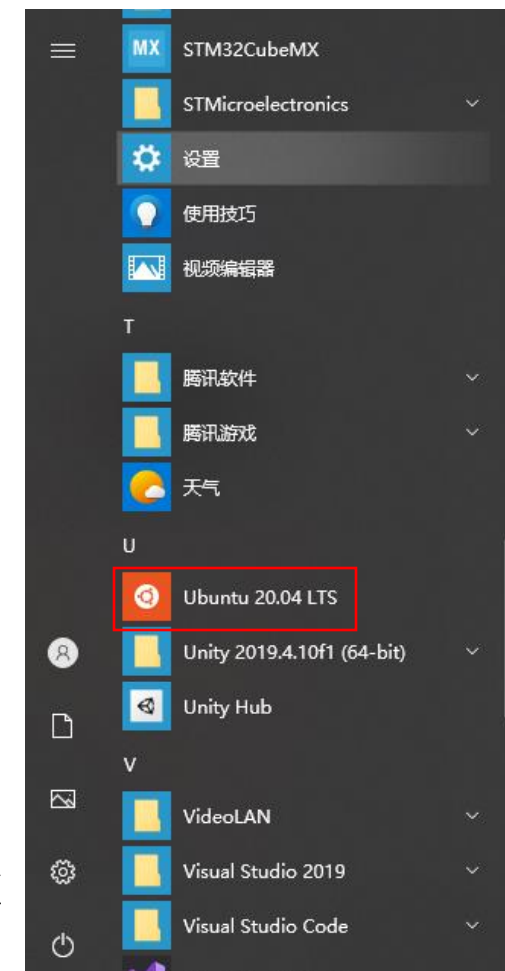
C:\WINDOWS\system32>
```

Check the version of WSL.



## 1.2 Install GCC on WSL

- Once you finished the installation of Ubuntu 20.04 LTS, you can find it on your start menu.
- Open it and you will see a Terminal for Linux
- You can set username and password for your system (Please remember this password as you need it to switch to root user later)
- Use the two commands below to install GNU: (If you are using any Linux distribution based on debian you can use below to install, too)
  - **sudo apt update**      this command will update your apt library (apt: Advanced Packaging Tools)
  - **sudo apt install g++ -y**      this command will install g++ and its independence





```
liao@DESKTOP-OOC4F37: ~  
适用于 Linux 的 Windows 子系统现已在 Microsoft Store 中可用！  
你可以通过运行“wsl.exe --update”或通过访问 https://aka.ms/wslstorepage 进行升级  
从 Microsoft Store 安装 WSL 将可以更快地获取最新的 WSL 更新。  
有关详细信息，请访问 https://aka.ms/wslstoreinfo  
  
To run a command as administrator (user “root”), use “sudo <command>”.  
See “man sudo_root” for details.  
  
liao@DESKTOP-OOC4F37:~$ sudo apt update  
[sudo] password for liao:  
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]  
Hit:2 http://archive.ubuntu.com/ubuntu focal InRelease  
Get:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]  
Get:4 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [2401 kB]  
Get:5 http://archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]  
Get:6 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [8628 kB]  
Get:7 http://archive.ubuntu.com/ubuntu focal/universe Translation-en [5124 kB]  
Get:8 http://archive.ubuntu.com/ubuntu focal/universe amd64 c-n-f Metadata [265 kB]  
Get:9 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 Packages [144 kB]  
Get:10 http://archive.ubuntu.com/ubuntu focal/multiverse Translation-en [104 kB]  
Get:11 http://archive.ubuntu.com/ubuntu focal/multiverse amd64 c-n-f Metadata [9136 B]  
Get:12 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [2788 kB]  
Get:13 http://archive.ubuntu.com/ubuntu focal-updates/main Translation-en [461 kB]  
Get:14 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metadata [17.0 kB]  
Get:15 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [2243 kB]  
Get:16 http://archive.ubuntu.com/ubuntu focal-updates/restricted Translation-en [313 kB]  
Get:17 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 c-n-f Metadata [576 B]  
Get:18 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [1105 kB]  
Get:19 http://archive.ubuntu.com/ubuntu focal-updates/universe Translation-en [264 kB]  
Get:20 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 c-n-f Metadata [25.4 kB]
```

input the command and  
your password



```
liao@DESKTOP-OOC4F37: ~  
Get:41 http://security.ubuntu.com/ubuntu focal-security/multiverse Translation-en [5504 B]  
Get:42 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 c-n-f Metadata [548 B]  
Fetched 27.9 MB in 36s (773 kB/s)  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
80 packages can be upgraded. Run 'apt list --upgradable' to see them.  
liao@DESKTOP-OOC4F37: ~$ sudo apt install g++ -y  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  binutils binutils-common binutils-x86-64-linux-gnu cpp cpp-9 g++-9 gcc gcc-10-base gcc-9 gcc-9-base libasan5  
  libatomic1 libbinutils libc-dev-bin libc6-dev libc++1-0 libcrypt-dev libctf-nobfd0 libctf0 libgcc-9-dev libgcc-s1  
  libgomp1 libisl22 libitm1 liblsan0 libmpc3 libquadmath0 libstdc++-9-dev libstdc++6 libtsan0 libubsan1 linux-libc-dev  
  manpages-dev  
Suggested packages:  
  binutils-doc cpp-doc gcc-9-locales g++-multilib g++-9-multilib gcc-9-doc gcc-multilib make autoconf automake libtool  
  flex bison gdb gcc-doc gcc-9-multilib glibc-doc libstdc++-9-doc  
The following NEW packages will be installed:  
  binutils binutils-common binutils-x86-64-linux-gnu cpp cpp-9 g++ g++-9 gcc gcc-9 gcc-9-base libasan5 libatomic1  
  libbinutils libc-dev-bin libc6-dev libc++1-0 libcrypt-dev libctf-nobfd0 libctf0 libgcc-9-dev libgomp1 libisl22  
  libitm1 liblsan0 libmpc3 libquadmath0 libstdc++-9-dev libtsan0 libubsan1 linux-libc-dev manpages-dev  
The following packages will be upgraded:  
  gcc-10-base libgcc-s1 libstdc++6  
3 upgraded, 31 newly installed, 0 to remove and 77 not upgraded.  
Need to get 44.7 MB of archives.  
After this operation, 197 MB of additional disk space will be used.  
Get:1 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 gcc-10-base amd64 10.5.0-1ubuntu1~20.04 [20.8 kB]  
Get:2 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libstdc++6 amd64 10.5.0-1ubuntu1~20.04 [501 kB]  
1% [2 libstdc++6 2613 B/501 kB 1%]
```

input the command to install g++



## 1.3 Verify GCC on WSL

- You can input command: `gcc --version` or `g++ --version` to check whether the GCC is installed well

```
liao@DESKTOP-OOC4F37: ~  
Setting up binutils-x86-64-linux-gnu (2.34-6ubuntu1.6) ...  
Setting up binutils (2.34-6ubuntu1.6) ...  
Setting up libgcc-9-dev:amd64 (9.4.0-1ubuntu1~20.04.2) ...  
Setting up cpp (4:9.3.0-1ubuntu2) ...  
Setting up gcc-9 (9.4.0-1ubuntu1~20.04.2) ...  
Setting up libstdc++-9-dev:amd64 (9.4.0-1ubuntu1~20.04.2) ...  
Setting up gcc (4:9.3.0-1ubuntu2) ...  
Setting up g++-9 (9.4.0-1ubuntu1~20.04.2) ...  
Setting up g++ (4:9.3.0-1ubuntu2) ...  
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode  
Processing triggers for man-db (2.9.1-1) ...  
Processing triggers for libc-bin (2.31-0ubuntu9.9) ...  
liao@DESKTOP-OOC4F37:~$ gcc --version  
gcc (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4.0  
Copyright (C) 2019 Free Software Foundation, Inc.  
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
  
liao@DESKTOP-OOC4F37:~$ g++ --version  
g++ (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4.0  
Copyright (C) 2019 Free Software Foundation, Inc.  
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
  
liao@DESKTOP-OOC4F37:~$ _
```

Input `gcc --version` or `g++ --version` to check if the compiler is installed successfully



## 2 Download and install LLVM on macOS

### 2.1 Install CLT (Xcode Command Line Tool) on macOS

- Open the Terminal on your Mac
- Input `g++` to check whether the CLT or GCC is installed
- If not, the system will guide you to install CLT
- You can also install CLT by package provided by Apple:  
<https://developer.apple.com/download/more/>
- For more info regarding the CLT installation you can refer to  
<https://www.easeus.com/computer-instruction/install-xcode-command-line-tools-on-mac.html>





## 2.2 Verify LLVM on macOS

- The same as verifying GNU, using: `g++ --version`

```
gdjs2@xiaozhaoqideMacBook-Pro ~  
$ g++ --version  
Configured with: --prefix=/Library/Developer/CommandLineTools/usr --with-gxx-include-dir=/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include/c++/4.2.1  
Apple clang version 12.0.0 (clang-1200.0.32.28)  
Target: x86_64-apple-darwin20.2.0  
Thread model: posix  
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
```



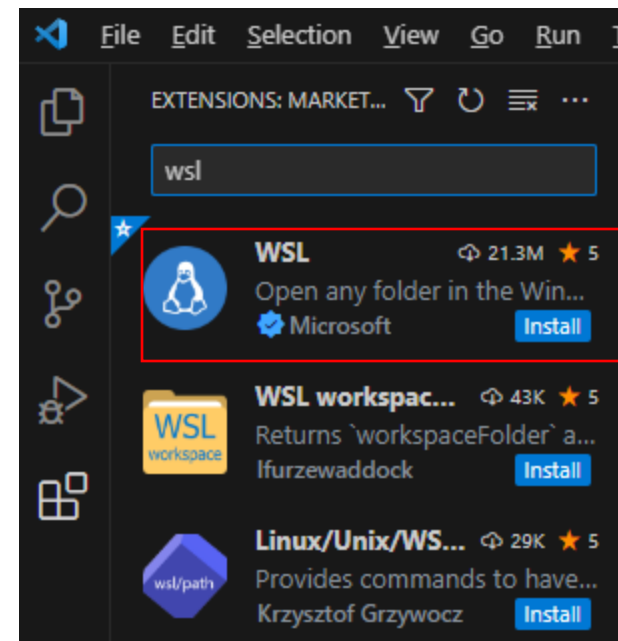
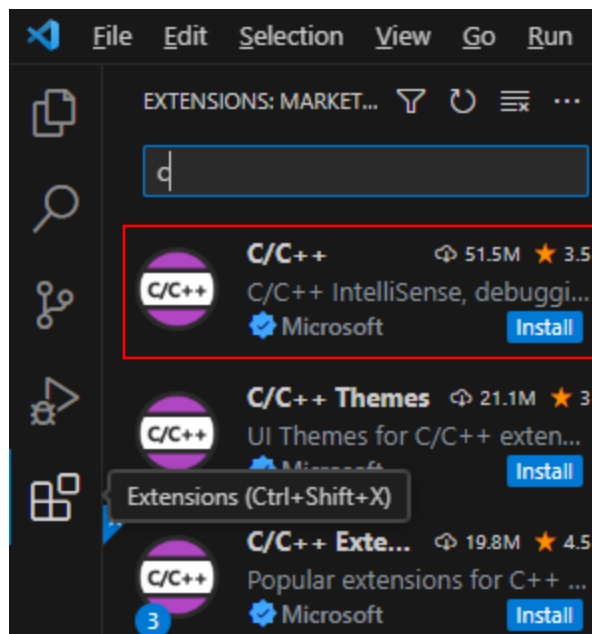
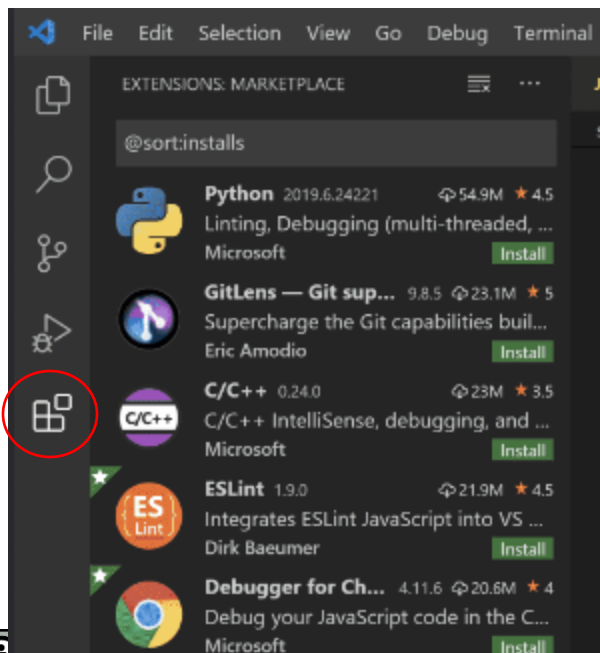
# 3.Download and install editor

To install **VSCode**, you can visit: <https://code.visualstudio.com/> to download the package for your platform (Linux, Windows or macOS).

After you install VSCode, you need to install two plugins at least to support your development:

1. C/C++ plugin
2. WSL plugin

Start VSCode, press the “Extensions” icon on the left margin, select the two plugins or search **c** and **wsl** key words respectively to find the two plugins.

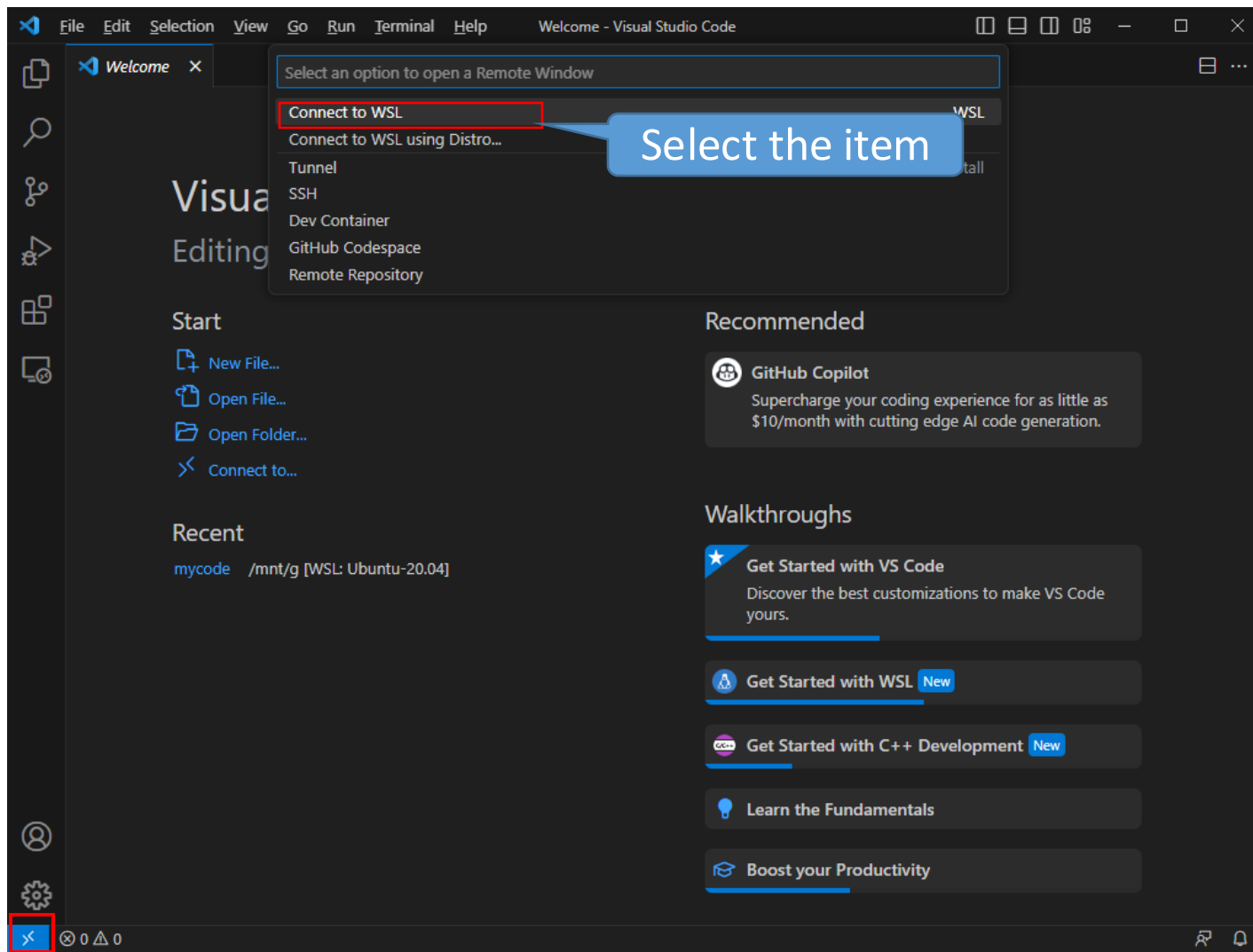




# 3.Download and install editor (Cont.)

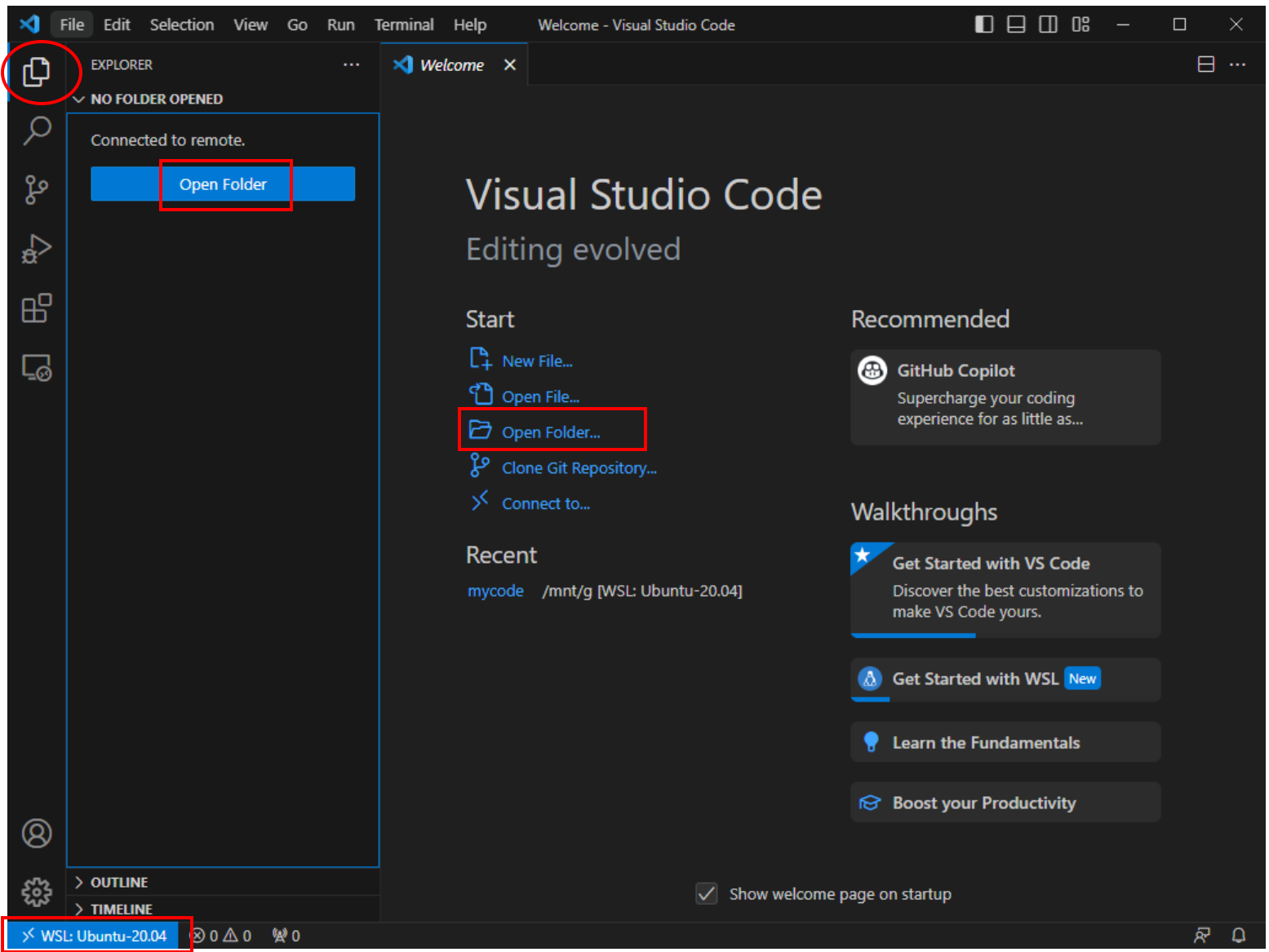
Now, you need to switch your VSCode to WSL system.

Click the blue button on left of the bottom, and choose **“Connect to WSL”**





Click “Open Folder...”in Welcome page or “Open Folder” button in Explorer or “File”→”Open Folder...” menu item.





File Edit Selection View Go Run Terminal Help

Welcome - Visual Studio Code

EXPLORER

NO FOLDER OPENED

Connected to remote.

Open Folder

/mnt/e/

..

\$RECYCLE.BIN

CCode

QMDownload

System Volume Info

tools

OK

Show Local

Type /mnt/,you can see all drives in your computer, and select one, such as e

Select the folder

Start

New File...

Open File...

Open Folder...

Clone Git Repository...

Connect to...

Recent

mycode /mnt/g [WSL: Ubuntu-20.04]

Recommended

GitHub Copilot

Supercharge your coding experience for as little as...

Walkthroughs

Get Started with VS Code

Discover the best customizations to make VS Code yours.

Get Started with WSL

New

Learn the Fundamentals

Got a moment to help the VS Code team? Please tell us about your experience with VS Code so far.

Give Feedback

Remind Me Later

WSL: Ubuntu-20.04

0 0 0



FileEditSelectionViewGoRun...Welcome - CCode [WSL: Ubuntu-20.04] - Visual Studio Code

EXPLORER

CCODE [WSL: UBUNTU-20.04]

Visual Studio Code

Do you trust the authors of the files in this folder?

Code provides features that may automatically execute files in this folder.

If you don't trust the authors of these files, we recommend to continue in restricted mode as the files may be malicious. See [our docs](#) to learn more.

/mnt/e/CCode [WSL: Ubuntu-20.04]

☐ Trust the authors of all files in the parent folder 'e'

Yes, I trust the authors

No, I don't trust the authors

Trust folder and enable all features

Browse folder in restricted mode

Copilot

ghs

ted with VS Code

ted with WSL

Learn the Fundamentals

This workspace is on the Windows file system (/mnt/). For best performance, we recommend moving the workspace to the Linux file system (~/.home).

Source: WSL (Extension)

Read More

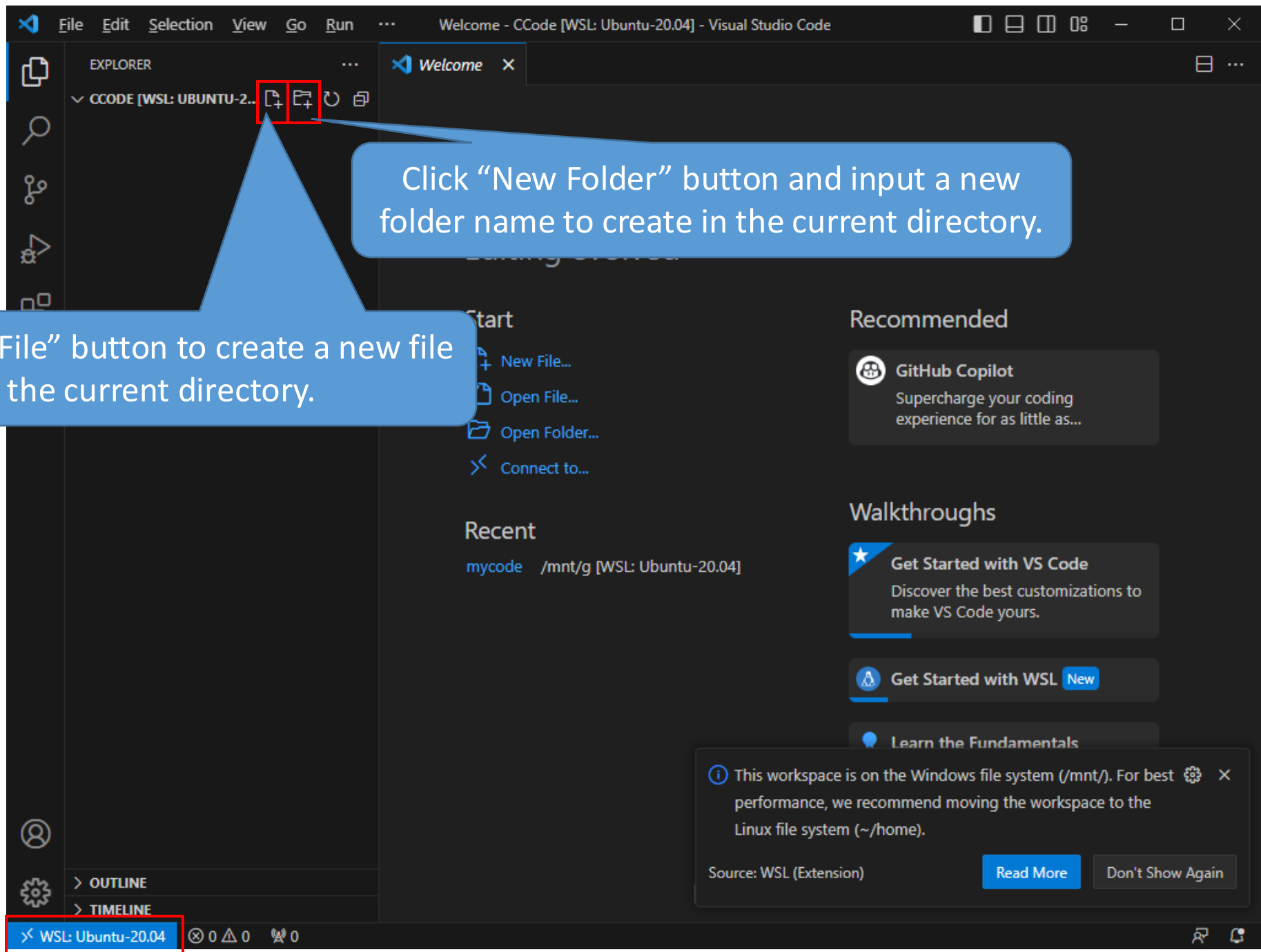
Don't Show Again

OUTLINE

TIMELINE

WSL: Ubuntu-20.04

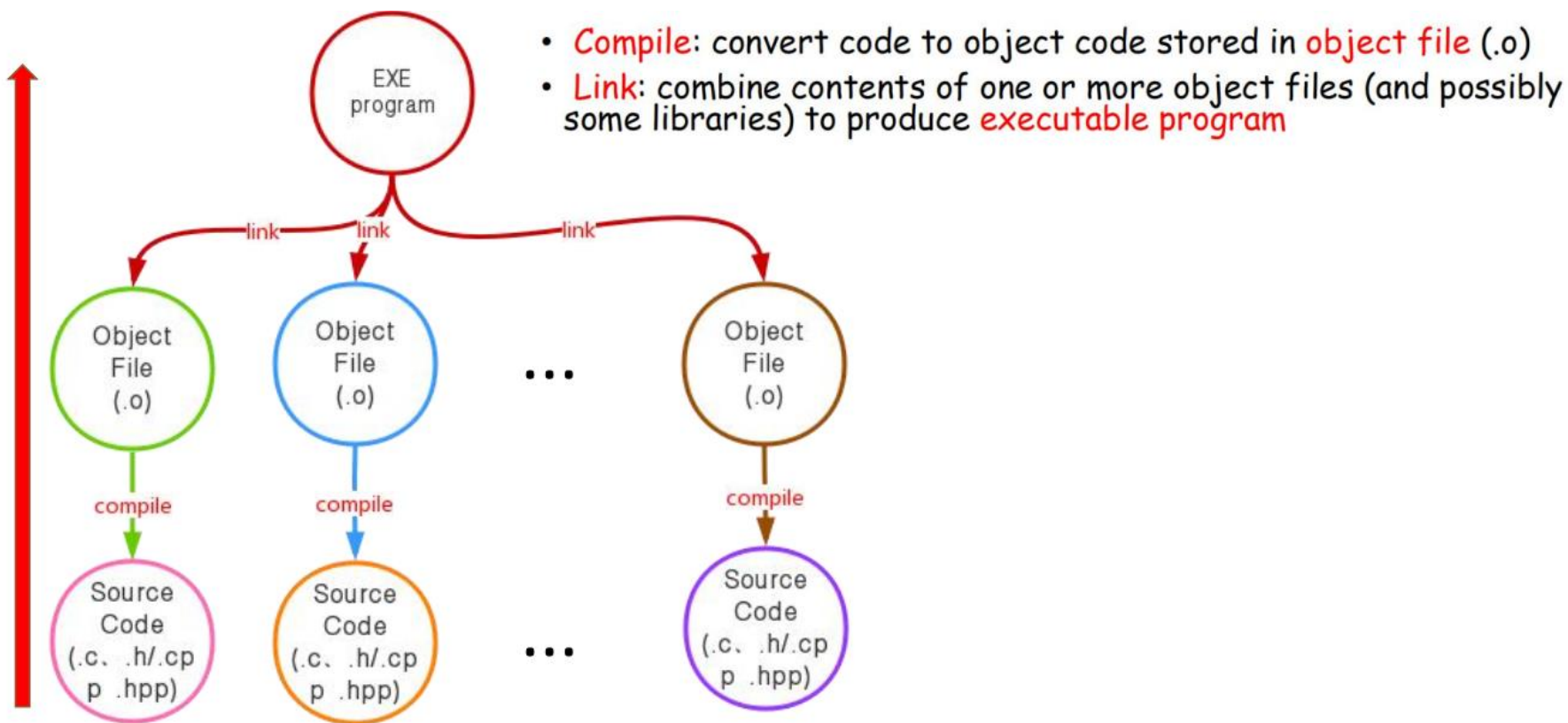
CC BY NC SA





# 4 Compile, Link and Run C/C++ Programs

## 4.1 The program compilation process

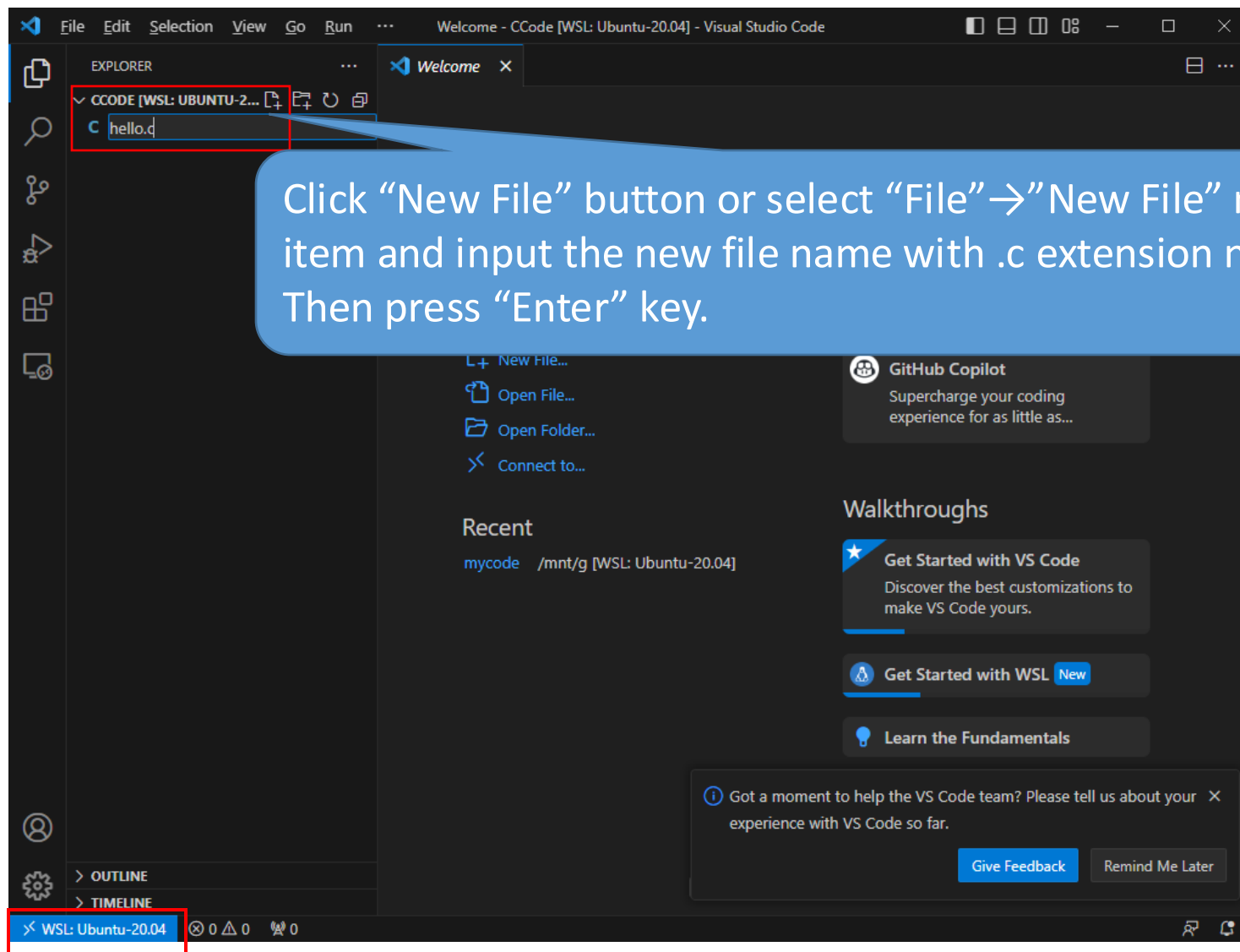






## 4.2 Compile, Link and Run C programs

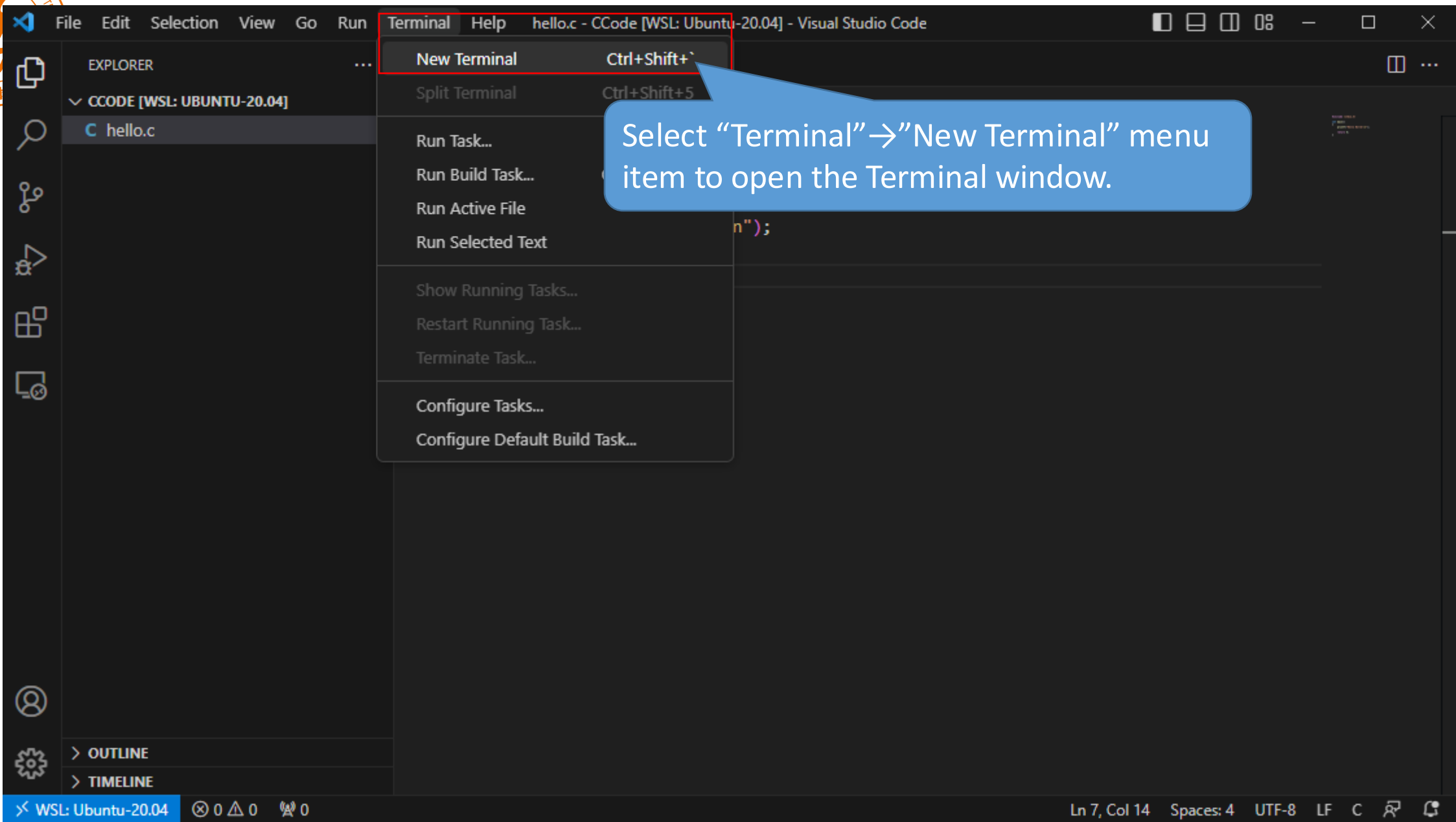
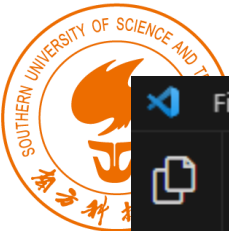
Compile/Link/Run a simple C program – hello.c



The image shows the Visual Studio Code editor interface. The title bar at the top reads "hello.c - CCode [WSL: Ubuntu-20.04] - Visual Studio Code". The Explorer sidebar on the left shows a file named "hello.c" under the "CCODE [WSL: UBUNTU-20.04]" folder. The main editor window displays the code for "hello.c". The code is as follows:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Hello World!\n");
6
7     return 0;
8 }
```

A red rectangle highlights the code block from line 1 to line 8. A blue callout bubble points to the "hello.c" tab in the editor, containing the text: "Input your code in the new file and save it by Ctrl+s or select 'File'→'Save' menu item." The status bar at the bottom shows "WSL: Ubuntu-20.04", "0 errors", "0 warnings", and "0 info". The current cursor position is "Ln 7, Col 14". The encoding is "UTF-8", line endings are "LF", and the tab size is "4 spaces".





Use **gcc** to compile the .c file.

EXPLORER

CCODE [WSL: UBUNTU-20.04]

hello.c

hello.c

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Hello World!\n");
6
7     return 0;
8 }
```

TERMINAL

bash - CCode

```
lia@DESKTOP-00C4F37:/mnt/e/CCode$ gcc -c hello.c
lia@DESKTOP-00C4F37:/mnt/e/CCode$ ls
hello.c hello.o
lia@DESKTOP-00C4F37:/mnt/e/CCode$ gcc hello.o -o hello
lia@DESKTOP-00C4F37:/mnt/e/CCode$ ls
hello hello.c hello.o
lia@DESKTOP-00C4F37:/mnt/e/CCode$ ./hello
Hello World!
lia@DESKTOP-00C4F37:/mnt/e/CCode$
```

OUTLINE

TIMELINE

WSL: Ubuntu-20.04

Ln 7, Col 14 Spaces: 4 UTF-8 LF C

compile

link

run

The output



The default output executable file is called “a.exe”(Windows) or “a.out”(Unix and Mac OS) if you don’t specify the name in compiling and linking step.

```
File Edit Selection View Go Run Terminal Help hello.c - CCode [WSL: Ubuntu-20.04] - Visual Studio Code

EXPLORER
CCODE [WSL: UBUNTU-20.04]
  hello
  C hello.c
  hello.o

C hello.c
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Hello World!\n");
6
7     return 0;
8 }

TERMINAL
bash - CCode
liao@DESKTOP-00C4F37:/mnt/e/CCode$ gcc -c hello.c
liao@DESKTOP-00C4F37:/mnt/e/CCode$ ls
hello.c hello.o
liao@DESKTOP-00C4F37:/mnt/e/CCode$ gcc hello.o -o hello
liao@DESKTOP-00C4F37:/mnt/e/CCode$ ls
hello hello.c hello.o
liao@DESKTOP-00C4F37:/mnt/e/CCode$ ./hello
Hello World!
liao@DESKTOP-00C4F37:/mnt/e/CCode$ gcc hello.c
liao@DESKTOP-00C4F37:/mnt/e/CCode$ ls
a.out hello hello.c hello.o
liao@DESKTOP-00C4F37:/mnt/e/CCode$ ./a.out
Hello World!
liao@DESKTOP-00C4F37:/mnt/e/CCode$
```

The output

compile and link

run



## 4.3 Compile, Link and Run C++ programs

Compile/Link/Run a simple C++ program – helloworld.cpp

The screenshot shows the Visual Studio Code interface with the following components:

- EXPLORER:** Displays the file structure. A red box highlights the "New Folder" button (represented by a folder icon with a plus sign) in the top right corner of the Explorer panel. A blue callout bubble points to this button with the text: "Click 'New Folder' button to create a new folder in the current directory and input the folder name."
- Editor:** Shows the file "hello.c" with the following code:

```
1 #include <stdio.h>
```
- TERMINAL:** Shows the execution of the program. The commands and output are as follows:

```
lia@DESKTOP-00C4F37:/mnt/e/CCode$ gcc -c hello.c
lia@DESKTOP-00C4F37:/mnt/e/CCode$ ls
hello.c hello.o
lia@DESKTOP-00C4F37:/mnt/e/CCode$ gcc hello.o -o hello
lia@DESKTOP-00C4F37:/mnt/e/CCode$ ls
hello hello.c hello.o
lia@DESKTOP-00C4F37:/mnt/e/CCode$ ./hello
Hello World!
lia@DESKTOP-00C4F37:/mnt/e/CCode$ gcc hello.c
lia@DESKTOP-00C4F37:/mnt/e/CCode$ ls
a.out hello hello.c hello.o
lia@DESKTOP-00C4F37:/mnt/e/CCode$ ./a.out
Hello World!
lia@DESKTOP-00C4F37:/mnt/e/CCode$
```



File Edit Selection View Go Run Terminal Help

hello.c - CCode [WSL: Ubuntu-20.04] - Visual Studio Code

EXPLORER

CCODE [WSL: UBUNTU-2...]

- CPP
  - helloworld.cpp
- a.out
- hello
- hello.c
- hello.o

helloworld.cpp

```
1 #include <stdio.h>
2
3 int main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

bash - CCode

- liac@DESKTOP-00C4F37:/mnt/e/CCode\$ gcc -c hello.c
- liac@DESKTOP-00C4F37:/mnt/e/CCode\$ ls
- hello.c hello.o
- liac@DESKTOP-00C4F37:/mnt/e/CCode\$ gcc hello.o -o hello
- liac@DESKTOP-00C4F37:/mnt/e/CCode\$ ls
- hello hello.c hello.o
- liac@DESKTOP-00C4F37:/mnt/e/CCode\$ ./hello
- Hello World!
- liac@DESKTOP-00C4F37:/mnt/e/CCode\$ gcc hello.c
- liac@DESKTOP-00C4F37:/mnt/e/CCode\$ ls
- a.out hello hello.c hello.o
- liac@DESKTOP-00C4F37:/mnt/e/CCode\$ ./a.out
- Hello World!
- liac@DESKTOP-00C4F37:/mnt/e/CCode\$

> OUTLINE

> TIMELINE

> WSL: Ubuntu-20.04

Ln 7, Col 14 Spaces: 4 UTF-8 LF C

Click "New File" button and input the file name with .cpp extension name in the current folder.



File Edit Selection View Go Run ...

helloworld.cpp - CCode [WSL: Ubuntu-20.04] - Visual Studio Code

EXPLORERCPP helloworld.cppa.outhellohello.chello.o

helloworld.cpp

1#include <iostream>2using namespace std;34int main()5{6cout << "Hello World!!!" << endl;78return 0;9}

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

bash - CCode

liaoo@DESKTOP-00C4F37:/mnt/e/CCode\$ gcc -c hello.cliaoo@DESKTOP-00C4F37:/mnt/e/CCode\$ lshello.chello.o liaoo@DESKTOP-00C4F37:/mnt/e/CCode\$ gcc hello.o -o hello liaoo@DESKTOP-00C4F37:/mnt/e/CCode\$ lshello hello.chello.o liaoo@DESKTOP-00C4F37:/mnt/e/CCode\$ ./helloHello World! liaoo@DESKTOP-00C4F37:/mnt/e/CCode\$ gcc hello.c liaoo@DESKTOP-00C4F37:/mnt/e/CCode\$ lsa.out hello hello.chello.o liaoo@DESKTOP-00C4F37:/mnt/e/CCode\$ ./a.outHello World! liaoo@DESKTOP-00C4F37:/mnt/e/CCode\$ clear

WSL: Ubuntu-20.04

Input your code in the file .

Input "clear" command to delete all the commands.





You need to use **g++** to compile C++ program. The **-o** option is used to specify the output file name.

```
helloworld.cpp - CCode [WSL: Ubuntu-20.04] - Visual Studio Code
CPP > helloworld.cpp
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      cout << "Hello World!!!" << endl;
7
8      return 0;
9  }
```

**Terminal Output:**

```
liac@DESKTOP-OOC4F37:/mnt/e/CCode$ cd CPP
liac@DESKTOP-OOC4F37:/mnt/e/CCode/CPP$ g++ -c helloworld.cpp
liac@DESKTOP-OOC4F37:/mnt/e/CCode/CPP$ ls
helloworld.cpp  helloworld.o
liac@DESKTOP-OOC4F37:/mnt/e/CCode/CPP$ g++ -o helloworld helloworld.o
liac@DESKTOP-OOC4F37:/mnt/e/CCode/CPP$ ls
helloworld  helloworld.cpp  helloworld.o
liac@DESKTOP-OOC4F37:/mnt/e/CCode/CPP$ ./helloworld
Hello World!!!
liac@DESKTOP-OOC4F37:/mnt/e/CCode/CPP$ g++ helloworld.cpp
liac@DESKTOP-OOC4F37:/mnt/e/CCode/CPP$ ls
a.out  helloworld  helloworld.cpp  helloworld.o
liac@DESKTOP-OOC4F37:/mnt/e/CCode/CPP$ ./a.out
Hello World!!!
liac@DESKTOP-OOC4F37:/mnt/e/CCode/CPP$ g++ -o helloworld2 helloworld.cpp
liac@DESKTOP-OOC4F37:/mnt/e/CCode/CPP$ ls
a.out  helloworld  helloworld.cpp  helloworld.o  helloworld2
liac@DESKTOP-OOC4F37:/mnt/e/CCode/CPP$ ./helloworld2
Hello World!!!
liac@DESKTOP-OOC4F37:/mnt/e/CCode/CPP$
```

**Annotations:**

- compile**: points to `g++ -c helloworld.cpp`
- link**: points to `g++ -o helloworld helloworld.o`
- run**: points to `./helloworld`
- compile and link**: points to `g++ helloworld.cpp`
- run**: points to `./a.out`
- compile and link**: points to `g++ -o helloworld2 helloworld.cpp`
- run**: points to `./helloworld2`

**Callouts:**

- The output: points to `Hello World!!!` (first instance)
- The output: points to `Hello World!!!` (second instance)
- The output: points to `Hello World!!!` (third instance)



# 5 Terminal Output

## 5.1 Formatting output with *printf*

***printf*** (*format-control-string*, *other-arguments*)

***format-control-string*** describes the output format, which consists of conversion specifiers, field widths, precisions and literal characters with percent sign(%).

| Conversion specifier      | Description  |
|---------------------------|--|
| d                         | Display as a <i>signed decimal integer</i> .   |
| i                         | Display as a <i>signed decimal integer</i> . [Note: The i and d specifiers are <i>different</i> when used with scanf.]   |
| o                         | Display as an <i>unsigned octal integer</i> .  |
| u                         | Display as an <i>unsigned decimal integer</i> .  |
| x or X                    | Display as an <i>unsigned hexadecimal integer</i> . X causes the digits 0–9 and the <i>uppercase</i> letters A–F to be used in the display and x causes the digits 0–9 and the <i>lowercase</i> letters a–f to be used in the display. |
| h, l or ll (letter “ell”) | Place <i>before</i> any integer conversion specifier to indicate that a short, long or long long integer is displayed, respectively. These are called <b>length modifiers</b> .  |
| e or E                    | Display a floating-point value in <i>exponential notation</i> .  |
| f or F                    | Display floating-point values in <i>fixed-point notation</i> (F is supported in the Microsoft Visual C++ compiler in Visual Studio 2015 and higher).   |
| g or G                    | Display a floating-point value in either the <i>floating-point form</i> f or the exponential form e (or E), based on the magnitude of the value.   |
| L                         | Place before any floating-point conversion specifier to indicate that a long double floating-point value should be displayed.  |



| Type                   | Format Specifier |
|------------------------|------------------|
| int                    | %d               |
| char                   | %c               |
| float                  | %f               |
| double                 | %lf              |
| short int              | %hd              |
| unsigned int           | %u               |
| long int               | %li              |
| long long int          | %lli             |
| unsigned long int      | %lu              |
| unsigned long long int | %llu             |
| signed char            | %c               |
| unsigned char          | %c               |
| long double            | %Lf              |

Example:

```
int a=1234;
float f=123.456;
char ch='a';
printf("%08d,%02d\n",a,a);
printf("%f,%8f,%8.1f,%.2f,%.2e\n",f,f,f,f,f);
printf("%03c\n",ch);
```


Sample output:

```
1234,1234
123.456000,123.456000, 123.5,123.46,1.23e+02
a
```



## 5.2 *cout*

**cout** << variable1(expression1) [<< variable2 << variable n];

```
CPP >  coutdemo.cpp

4  int main()
5  {
6      int a = 10;
7      float b = 45.7;
8      char c = 'A';
9
10     cout << "a = " << a << ",b = " << b << ",c = " << c << endl;
11
12     return 0;
13 }
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
• liao@DESKTOP-00C4F37:/mnt/e/CCode$ cd CPP
• liao@DESKTOP-00C4F37:/mnt/e/CCode/CPP$ g++ coutdemo.cpp
• liao@DESKTOP-00C4F37:/mnt/e/CCode/CPP$ ./a.out
a = 10,b = 45.7,c = A
• liao@DESKTOP-00C4F37:/mnt/e/CCode/CPP$
```



## 6.1 Exercises

Write a program to initialize three variables which equal to 0.1, 0.2, 0.3, then print them with two decimal points.

A screenshot of a terminal window with a dark background. At the top, there are four tabs: "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", and "TERMINAL", with "TERMINAL" being the active tab. The terminal shows the following text:

```
wdx@DESKTOP-R133B5N:~/Cpp$ g++ -o main main.cpp && ./main
0.10
0.20
0.30
wdx@DESKTOP-R133B5N:~/Cpp$
```

Your output should look like something above. You can use `printf` to achieve this, or you can explore the `cout` way.



## 6.2 Exercises

Copy the following code into 3 files, and compile them together to an executable file. Find the bugs if there are some.

Step 1: Compile main.cpp

Step 2: Compile add.cpp

Step 3: Link the two object files.

### main.cpp

```
#include <iostream>
#include "Add.h"

int main()
{
    int num1 = 2147483647;
    int num2 = 1;
    int result = 0;

    result = add(num1, num2);

    cout << "The result is " << result << endl;
    return 0;
}
```

### add.h

```
#pragma once

int add(int n1, int n2);
```

### add.cpp

```
#include "add.h"

int Add(int number1, int number2);
{
    return n1 + n2;
}
```