VSC – Makefile (V1)

By Kinley

Tasks

- C++ Compiler Installation
- Make utility Installation
- Compiling and Linking CPP file
- Makefile Overview
- Integrate Makefile with Visual Studio Code
- Demo

C++ Compiler (Windows)

C++ Compiler (Windows)

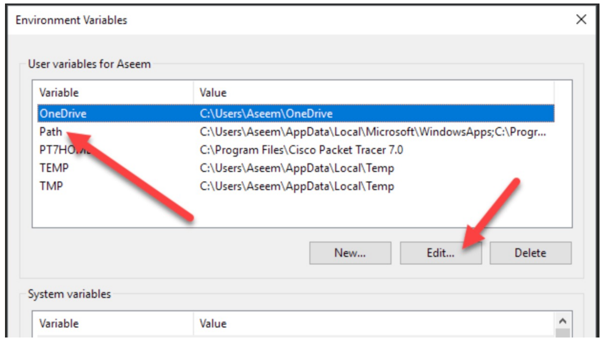
- NOTE: for Linux and Mac users, the C++ compiler should already be pre-installed.
- NOTE: the C++ compiler command is named gcc, g++ or clang (mac)
- The installation package used is called MinGW-x64
- The installation is based on MSYS2, which provides up-to-date native builds of GCC, Mingw-w64, and other helpful C++ tools and libraries.
- See next slides for the installation procedure

Steps (Windows)

1. Install MinGW-x64 based on MSYS2
   - Open MSYS2 web page and follow the instructions under installation
     - TAKE NOTE OF THE INSTALLATION FOLDER (YOU NEED IT IN STEP 2)
   - After MSYS2 is completed, it will pop up another window, and you need to run pacman to install the C++ compiler as follows:
     - $ pacman -S mingw-w64-x86_64-gcc
   - Verify indeed gcc (g++) is installed successfully (g++ --version)
2. Add the MinGW compiler to your path
   - By default, the path should be under c:\msys64\mingw64\bin
   - Open File Explorer to confirm the location of the g++ compiler.

Add MinGW compiler to Windows Path

- In the Windows search bar, type 'settings' to open your Windows Settings.
- Search for **Edit environment variables for your account**.
- Choose the Path variable in your **User variables** and then select **Edit**.



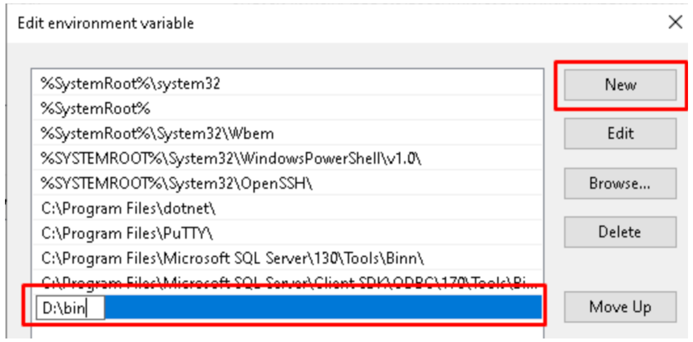
Environment Variables

User variables for Aseem

Variable	Value
OneDrive	C:\Users\Aseem\OneDrive
Path	C:\Users\Aseem\AppData\Local\Microsoft\WindowsApps;C:\Progr...
PT7HO	C:\Program Files\Cisco Packet Tracer 7.0
TEMP	C:\Users\Aseem\AppData\Local\Temp
TMP	C:\Users\Aseem\AppData\Local\Temp

System variables

Variable	Value

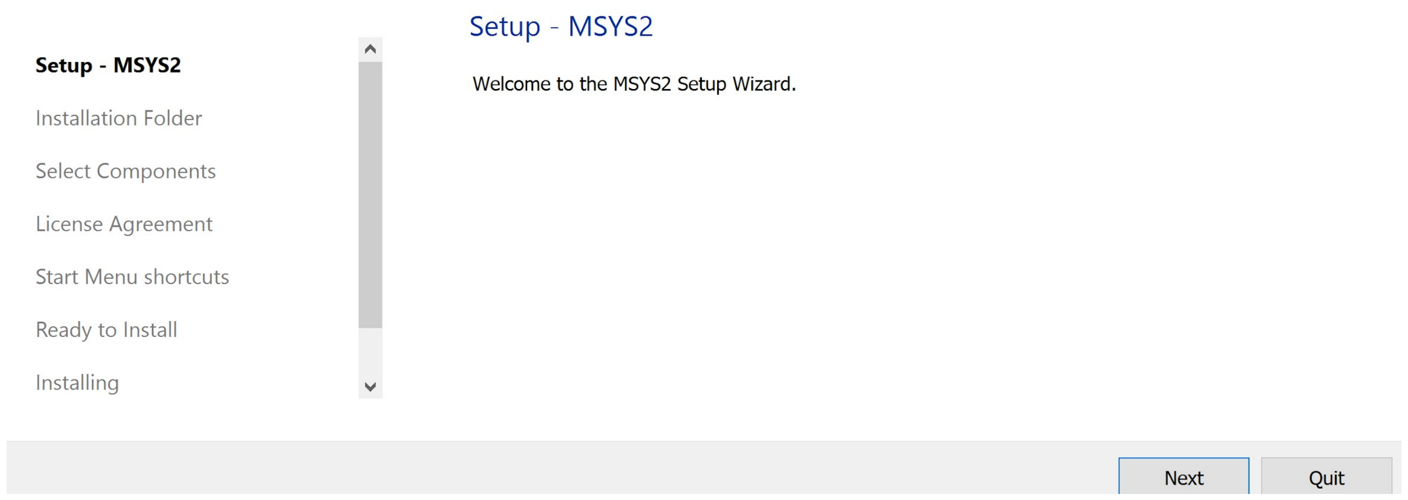


Edit environment variable

```
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\dotnet\
C:\Program Files\PuTTY\
C:\Program Files\Microsoft SQL Server\130\Tools\Binn\
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\170\Tools\Bi...
D:\bin
```

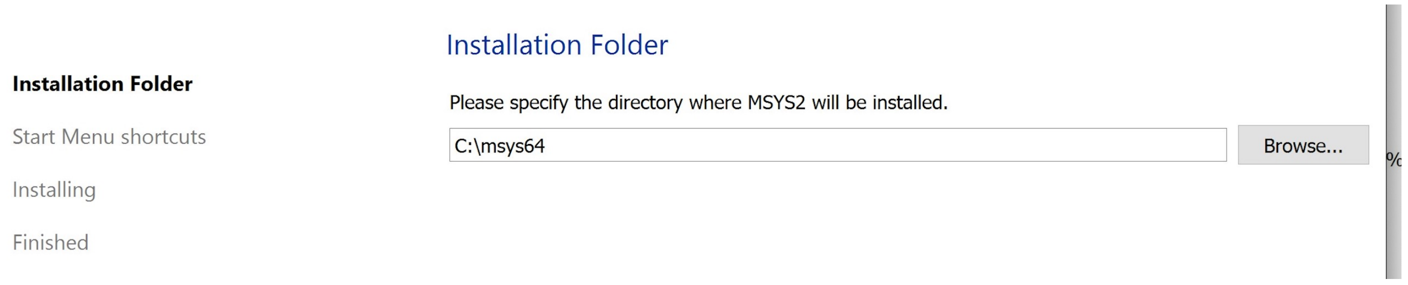
- Select **New** and add the Mingw-w64 destination folder path (ex: C:\msys64\mingw64\bin) to the system path. The exact path depends on which version of Mingw-w64 you have installed and where you installed it. If DEFAULT settings used, the path could be: C:\msys64\mingw64\bin.
- Select **OK** to save the updated PATH. You will need to reopen any console windows for the new PATH location to be available. Type g++ --version to confirm !!!

MSYS2 – Screen shoot



Setup - MSYS2

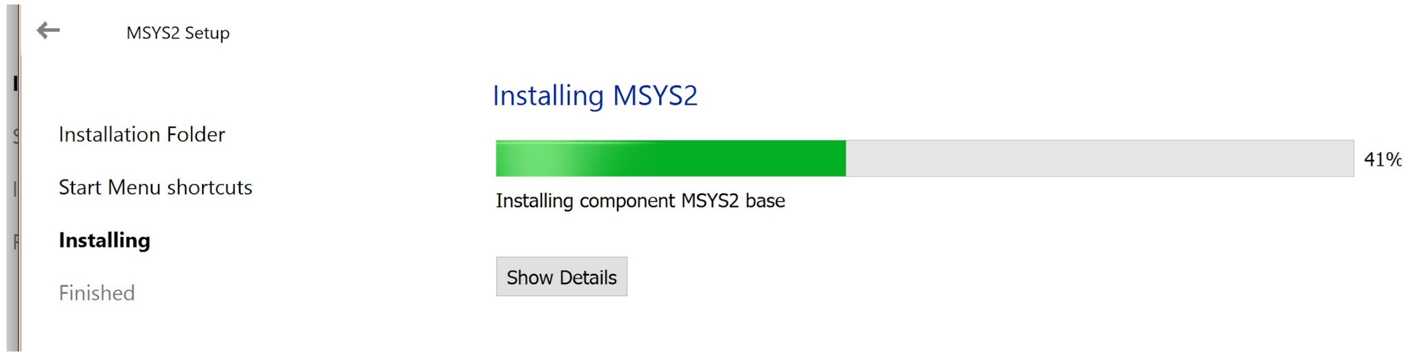
Welcome to the MSYS2 Setup Wizard.



Installation Folder

Please specify the directory where MSYS2 will be installed.

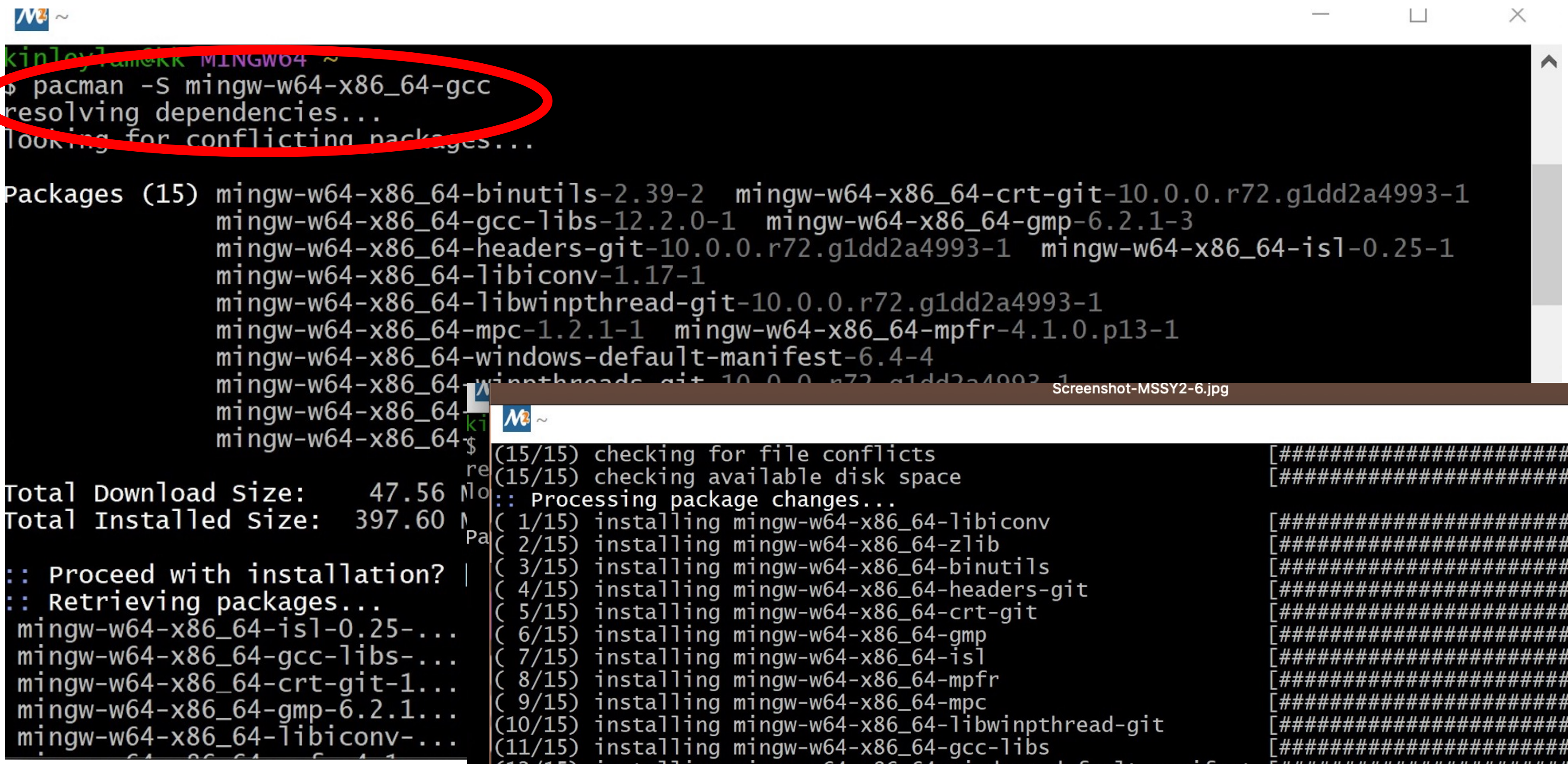
C:\msys64



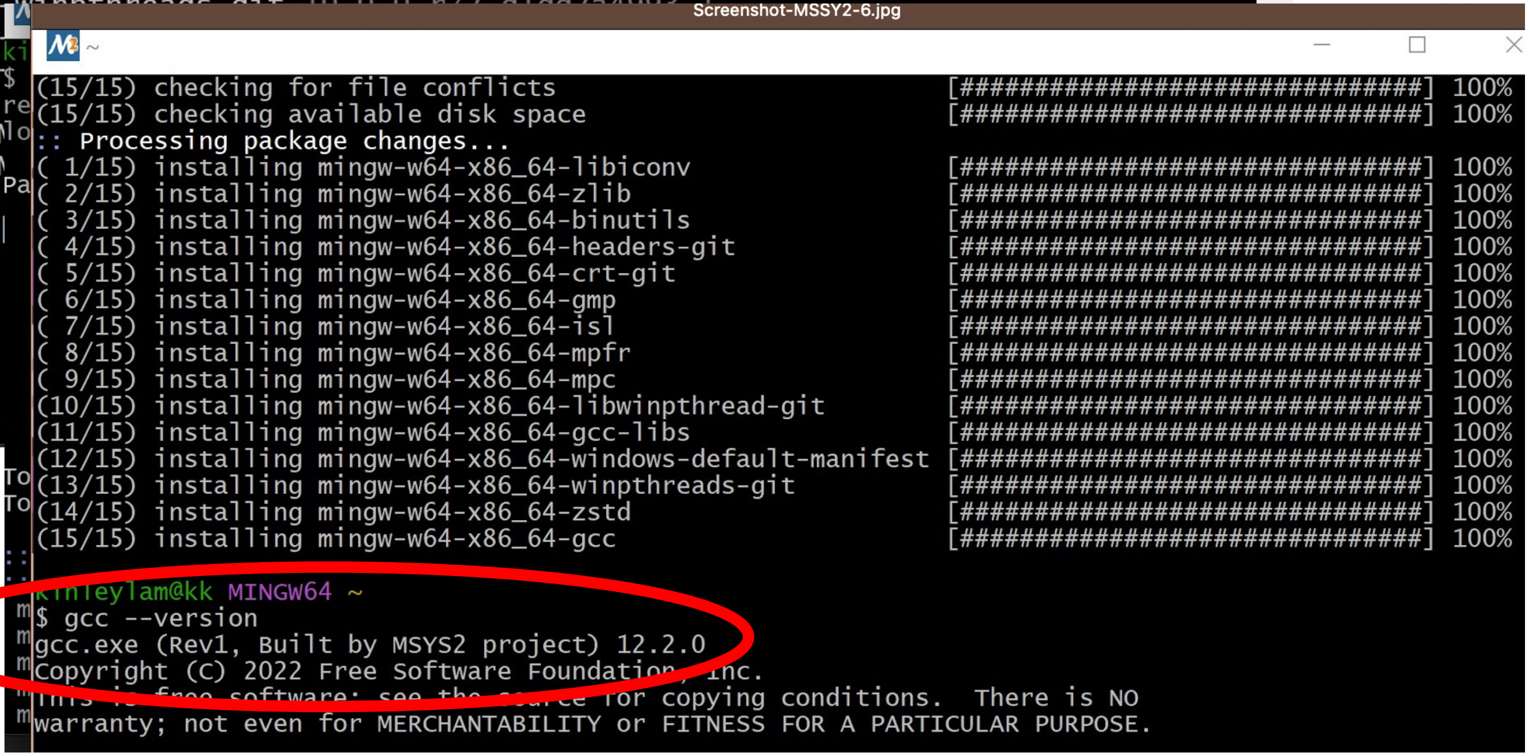
Installing MSYS2

41%

Installing component MSYS2 base



```
kinleylam@kk MINGW64 ~
$ pacman -S mingw-w64-x86_64-gcc
resolving dependencies...
looking for conflicting packages...

Packages (15) mingw-w64-x86_64-binutils-2.39-2  mingw-w64-x86_64-crt-git-10.0.0.r72.g1dd2a4993-1
              mingw-w64-x86_64-gcc-libs-12.2.0-1  mingw-w64-x86_64-gmp-6.2.1-3
              mingw-w64-x86_64-headers-git-10.0.0.r72.g1dd2a4993-1  mingw-w64-x86_64-isl-0.25-1
              mingw-w64-x86_64-libiconv-1.17-1
              mingw-w64-x86_64-libwinpthread-git-10.0.0.r72.g1dd2a4993-1
              mingw-w64-x86_64-mpc-1.2.1-1  mingw-w64-x86_64-mpfr-4.1.0.p13-1
              mingw-w64-x86_64-windows-default-manifest-6.4-4
              mingw-w64-x86_64-winpthreads-git-10.0.0.r72.g1dd2a4993-1
              mingw-w64-x86_64
              mingw-w64-x86_64

Total Download Size:    47.56 M
Total Installed Size:  397.60 M

:: Proceed with installation?
:: Retrieving packages...
 mingw-w64-x86_64-isl-0.25-...
 mingw-w64-x86_64-gcc-libs-...
 mingw-w64-x86_64-crt-git-1...
 mingw-w64-x86_64-gmp-6.2.1...
 mingw-w64-x86_64-libiconv-...
```



Screenshot-MSSY2-6.jpg

```
(15/15) checking for file conflicts                    [##############################] 100%
(15/15) checking available disk space                  [##############################] 100%
:: Processing package changes...
( 1/15) installing mingw-w64-x86_64-libiconv           [##############################] 100%
( 2/15) installing mingw-w64-x86_64-zlib               [##############################] 100%
( 3/15) installing mingw-w64-x86_64-binutils           [##############################] 100%
( 4/15) installing mingw-w64-x86_64-headers-git        [##############################] 100%
( 5/15) installing mingw-w64-x86_64-crt-git            [##############################] 100%
( 6/15) installing mingw-w64-x86_64-gmp                [##############################] 100%
( 7/15) installing mingw-w64-x86_64-isl                [##############################] 100%
( 8/15) installing mingw-w64-x86_64-mpfr               [##############################] 100%
( 9/15) installing mingw-w64-x86_64-mpc                [##############################] 100%
(10/15) installing mingw-w64-x86_64-libwinpthread-git  [##############################] 100%
(11/15) installing mingw-w64-x86_64-gcc-libs           [##############################] 100%
(12/15) installing mingw-w64-x86_64-windows-default-manifest [##############################] 100%
(13/15) installing mingw-w64-x86_64-winpthreads-git    [##############################] 100%
(14/15) installing mingw-w64-x86_64-zstd               [##############################] 100%
(15/15) installing mingw-w64-x86_64-gcc                [##############################] 100%

kinleylam@kk MINGW64 ~
$ gcc --version
gcc.exe (Rev1, Built by MSYS2 project) 12.2.0
Copyright (C) 2022 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.
```

Make utility (Windows)

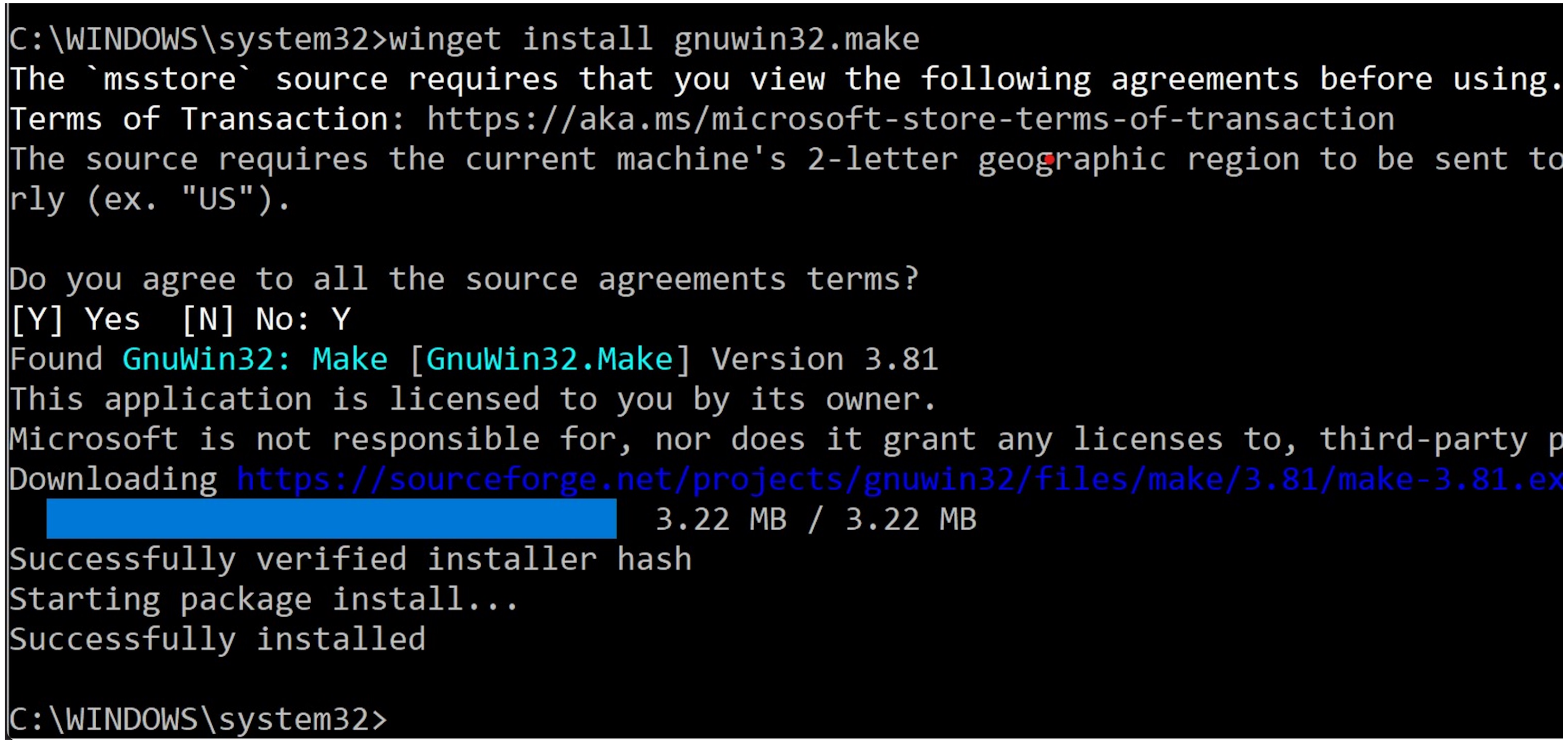
Make utility (Windows)

- NOTE: for Linux and Mac users, the Make utility should already be pre-installed.
- NOTE: the Make command is named make (lower-case for Linux and Mac), case-insensitive for Windows.
- Use the Windows command “winget” to install the Make utility

Steps (Windows)

1. Run windows console with administrator privilege (Run as administrator)
2. Connect VPN
3. From the console, type: winget install gnuwin32.make
   - See screen shot on next slide.
4. Add the Make command to your path

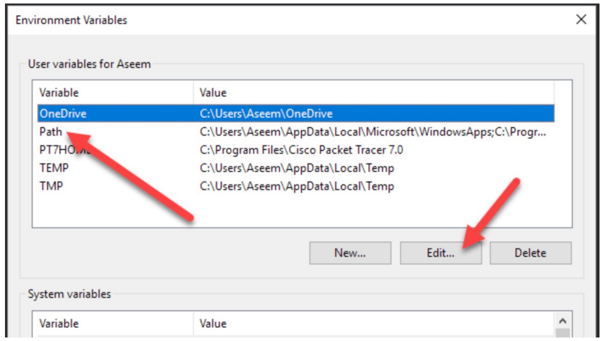
Make – screenshot



```
C:\WINDOWS\system32>winget install gnuwin32.make
The `msstore` source requires that you view the following agreements before using.
Terms of Transaction: https://aka.ms/microsoft-store-terms-of-transaction
The source requires the current machine's 2-letter geographic region to be sent to
rly (ex. "US").

Do you agree to all the source agreements terms?
[Y] Yes  [N] No: Y
Found GnuWin32: Make [GnuWin32.Make] Version 3.81
This application is licensed to you by its owner.
Microsoft is not responsible for, nor does it grant any licenses to, third-party p
Downloading https://sourceforge.net/projects/gnuwin32/files/make/3.81/make-3.81.ex
                                  3.22 MB / 3.22 MB
Successfully verified installer hash
Starting package install...
Successfully installed

C:\WINDOWS\system32>
```

Add Make to Windows Path

- In the Windows search bar, type 'settings' to open your Windows Settings.
- Search for **Edit environment variables for your account**.
- Choose the Path variable in your **User variables** and then select **Edit**.



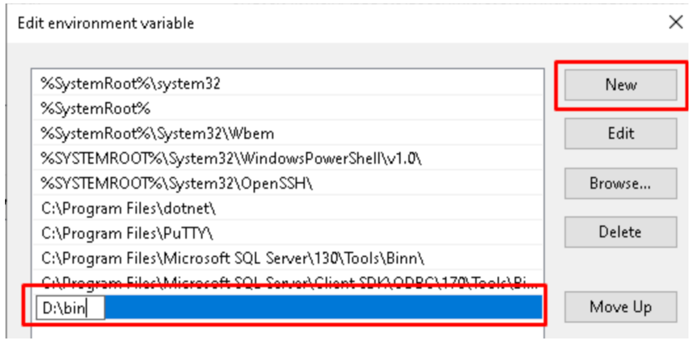
Environment Variables

User variables for Aseem

Variable	Value
OneDrive	C:\Users\Aseem\OneDrive
Path	C:\Users\Aseem\AppData\Local\Microsoft\WindowsApps;C:\Progr...
PT7HO	C:\Program Files\Cisco Packet Tracer 7.0
TEMP	C:\Users\Aseem\AppData\Local\Temp
TMP	C:\Users\Aseem\AppData\Local\Temp

System variables

Variable	Value



Edit environment variable

```
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\dotnet\
C:\Program Files\PuTTY\
C:\Program Files\Microsoft SQL Server\130\Tools\Binn\
C:\Program Files\Microsoft SQL Server\Client SDK\ODBC\170\Tools\Bi...
D:\bin
```

- Select **New** and add the Make destination folder path (ex: C:\Program Files(x86)\GnuWin32\bin) to the system path. The exact path depends on which version of Make you have installed and where you installed it. If DEFAULT settings used, the path could be: C:\Program Files(x86)\GnuWin32\bin.
- Select **OK** to save the updated PATH. You will need to reopen any console windows for the new PATH location to be available. Type make –version to confirm !!!

Compiling & Linking (Windows & Mac) - High-level

Steps

- In general, you compile all the cpp files into object files (.o) and then create the binary or executable file (called linking)
- Assumption: we only have a simple cpp structures (all cpp and headers under same directory and a simple header structure)
- Compiling a single cpp file (say foo.cpp)
  - g++ -std=c++17 foo.cpp –c foo.o

    or
  - g++ -std=c++17 foo.cpp –c (default name for the object file)
- Creating an executable file (called linking)
  - g++ -std=c++17 -o foo foo.o
- NOTE: you could combine compiling with linking and as one command:
  - g++ -std=c++17 foo.cpp -o foo
  - This command would not generate the object file.

Multiple Files

- If your CPP program is composed of multiple cpp files, you need to compile each cpp separately as follows:
  - g++ -std=c++17 <fileX>.cpp -c
    - fileX is one of the cpp files
- Next, link up all the objects file into the executable file as follows:
  - g++ -std=c++17 –o <exe> file1.o file2.o file3.o ….
    - file1, file2, file3 …. are all the compiled object files needed by the executable program.

Makefile (Windows & Mac)

Makefile - high-level

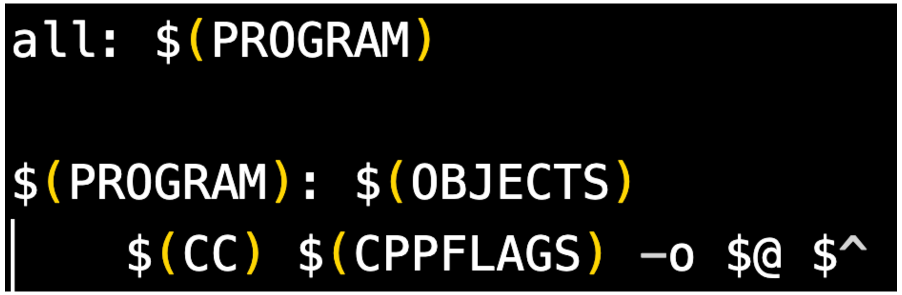
- The general syntax of a Makefile target rule is as follows:

```
target [target...] : [dependent ....]
[ command ...]
```

- Note: the arguments in brackets are optional and ellipsis means one or more. A tab to preface each command is required.
- Examples:

```
hello: main.o factorial.o hello.o
	$(CC) main.o factorial.o hello.o -o hello
```

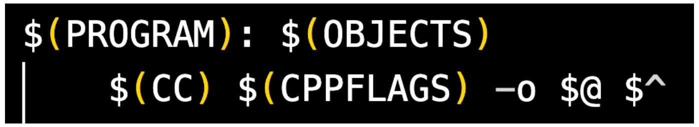
- You can define many target rules in a single Makefile.
- To execute a target:
  - make <target>



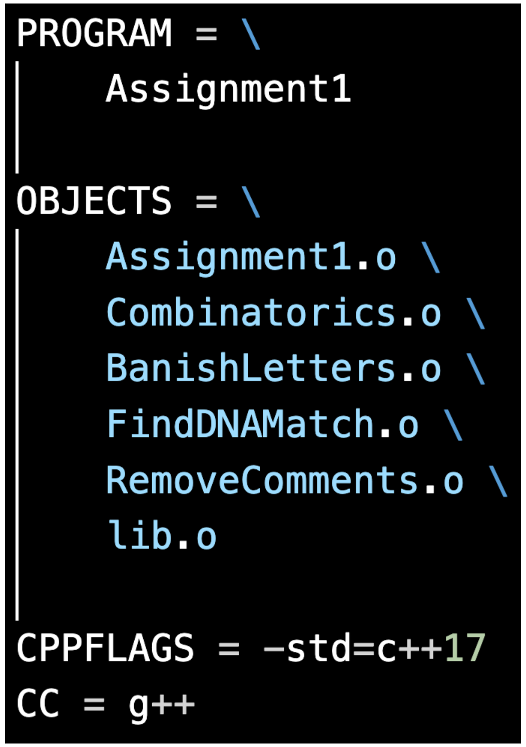
```
all: $(PROGRAM)

$(PROGRAM): $(OBJECTS)
	$(CC) $(CPPFLAGS) -o $@ $^
```

Macros - Makefile

- Macro (Custom)
  - You can define your own macros (PROGRAM, OBJECTS) or override the default ones (CC, CPPFLAGS).
- Special Macros (built-in)
  - $@ name of the file to be made (target)
  - $* prefix of the file to be made (target)
  - $? names of the CHANGED dependents.
  - $^ names of ALL the dependents (dependency).
  - $< name of the related file that caused the action



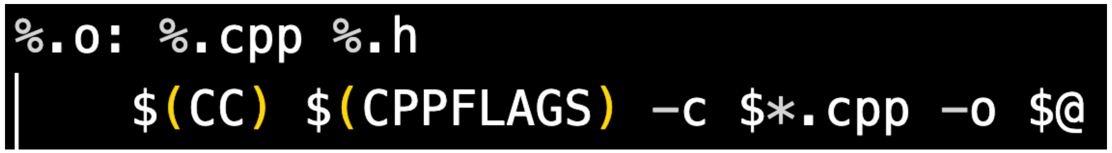
```
$(PROGRAM): $(OBJECTS)
	$(CC) $(CPPFLAGS) -o $@ $^
```



```
PROGRAM = \
	Assignment1

OBJECTS = \
	Assignment1.o \
	Combinatorics.o \
	BanishLetters.o \
	FindDNAMatch.o \
	RemoveComments.o \
	lib.o

CPPFLAGS = -std=c++17
CC = g++
```

Makefile – wildcard compile target

- For a simple header and cpp structure: one cpp file to one header file with same name, such as:
    - foo1.cpp foo1.h
    - foo2.cpp foo2.h ….
- Then you could create a general target (compile target) for all cpp files as ONE SINGLE TARGET as follows:

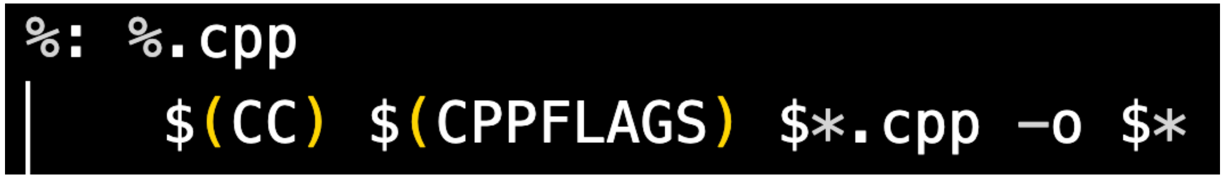


```
%.o: %.cpp %.h
	$(CC) $(CPPFLAGS) -c $*.cpp -o $@
```

- To compile any cpp file, type “make <file>.o”

Makefile – wildcard Link target

- If all executable files are single-source based (each cpp file has its own main() entry).
- Then you could create a general link target for all cpp files as ONE SINGLE TARGET as follows:



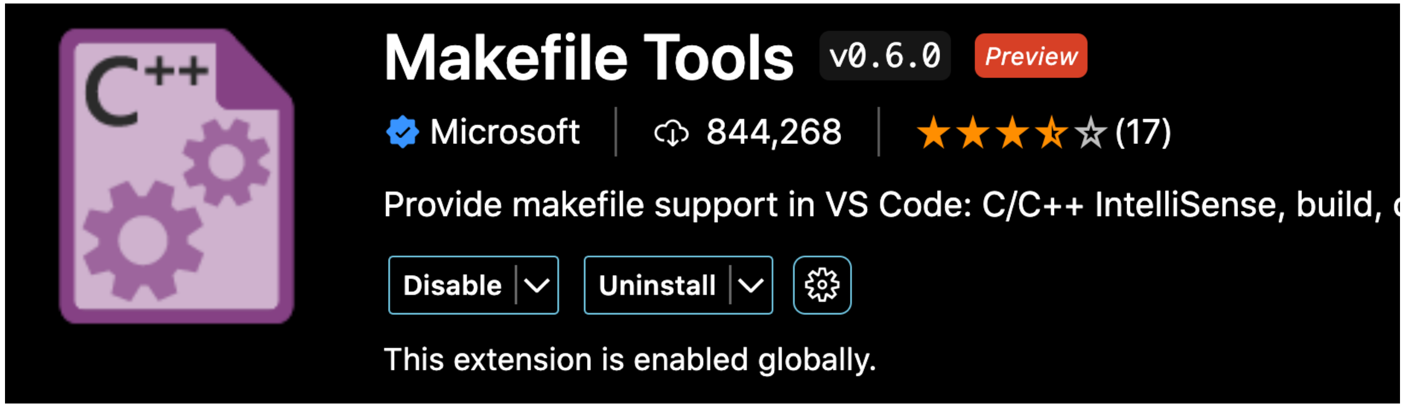
```
%: %.cpp
	$(CC) $(CPPFLAGS) $*.cpp -o $*
```

- Say, you created a standalone hello-world cpp file named hw.cpp, to make the executable file, you run ”make hw”.

VSC – Makefile Extension (Windows & Mac)

Prerequisite (Windows & Mac)

- Install Make extension to your VSC



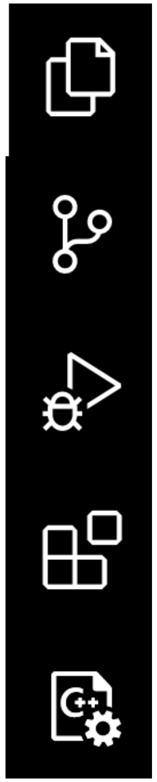
Makefile Tools v0.6.0 Preview

Microsoft | 844,268 | ★★★★☆ (17)

Provide makefile support in VS Code: C/C++ IntelliSense, build,

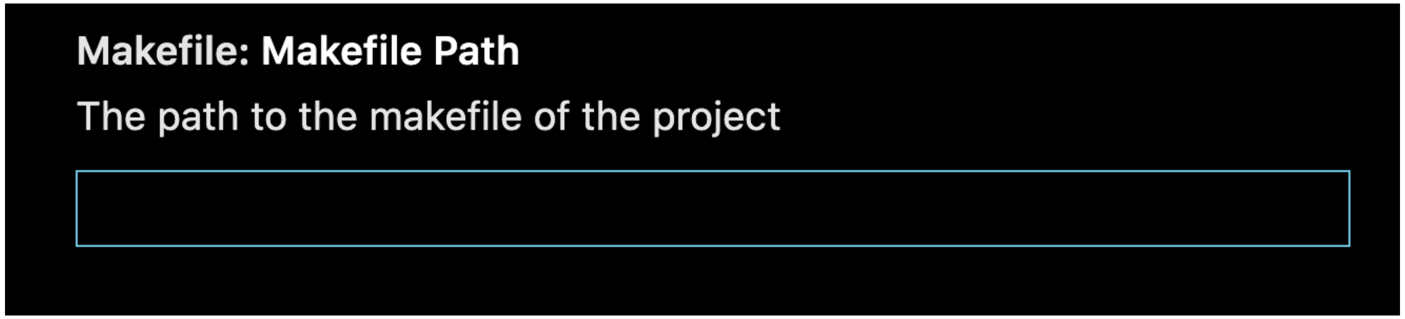
This extension is enabled globally.

- You will find the Makefile Icon on the sidebar



Steps - VSC Makefile

1. Create a home folder (make it simple, english letters preferred)
2. Run VSC and open the home folder
3. Create a makefile, such as makefile or Makefile right under your home folder
   - NOTE: if your makefile is NOT under the home folder, you have to change the setting.



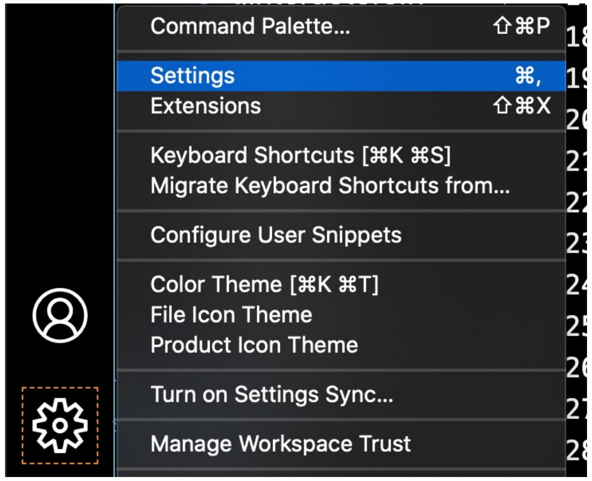
Makefile: Makefile Path

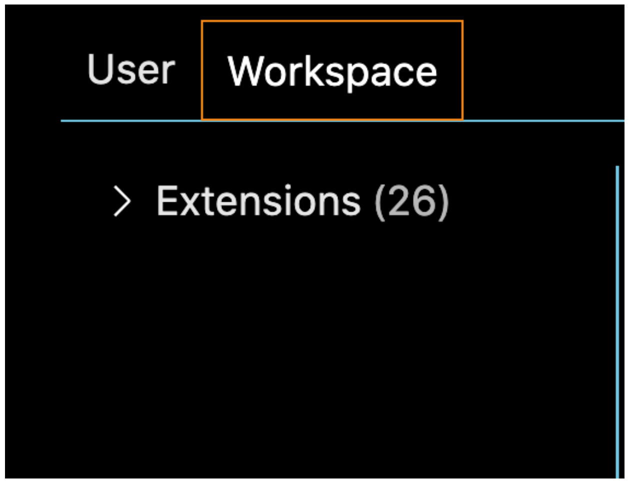
The path to the makefile of the project

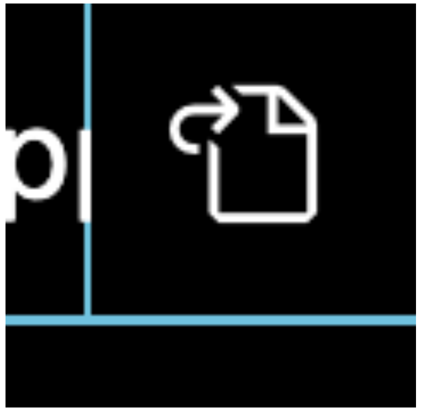
   - For illustration, the makefile is under the home directory

Steps – VSC Makefile

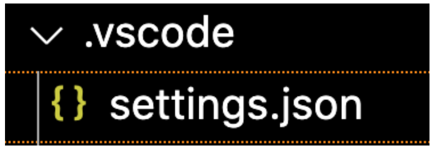
4. Create workspace settings.
   - Click the gear setting icon (lower left corner) and choose Settings
   - Select Workspace tab and click the open settings icon (upper right corner)





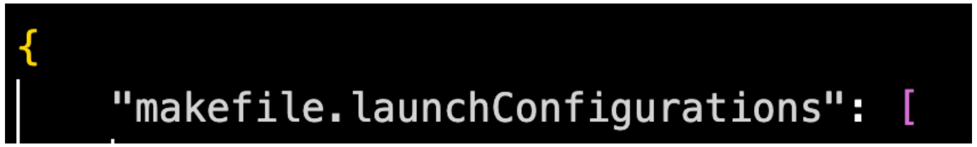


   - Alternatively, under .vscode folder, click open settings.json (if you have already created one)



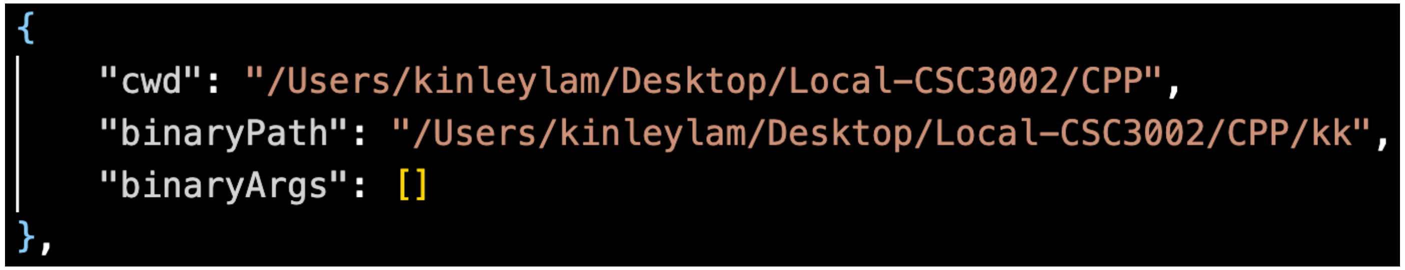
Steps – VSC Makefile

5. Create executable targets:
   - Create makefile.launchConfigurations entry (if not found):



```
{
    "makefile.launchConfigurations": [
```

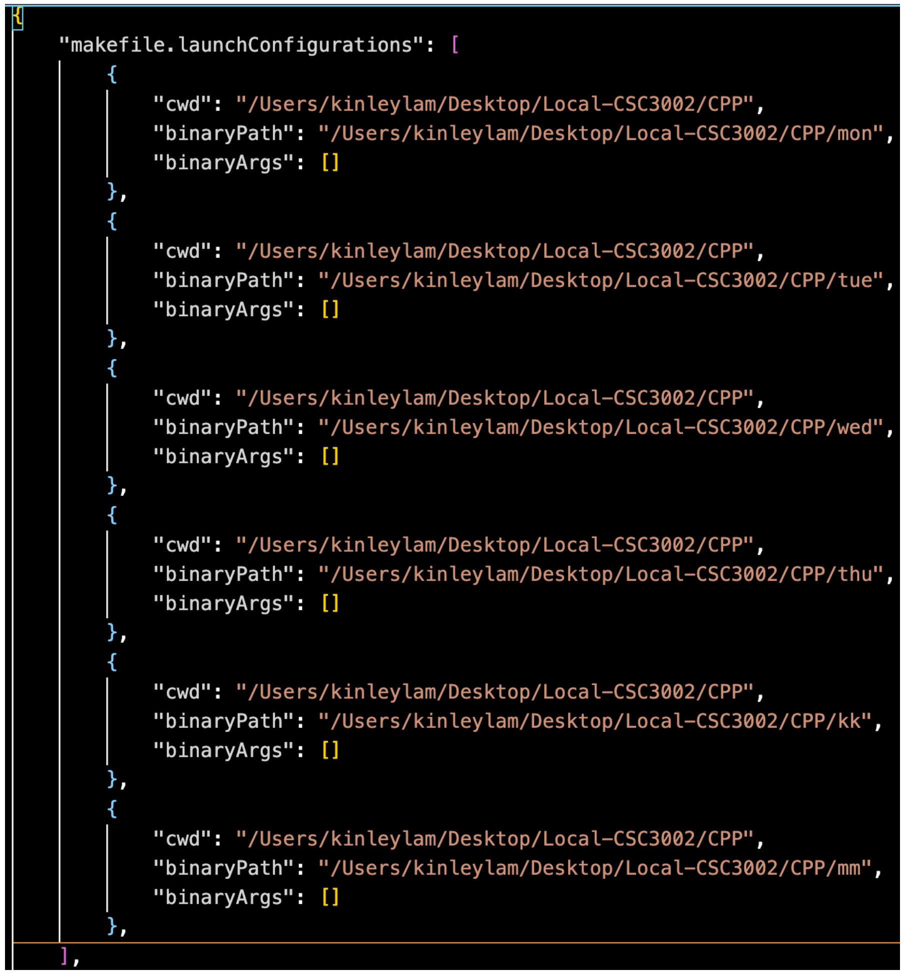
   - Define the executable location, name, and arguments (under launchConfigurations):



```
{
    "cwd": "/Users/kinleylam/Desktop/Local-CSC3002/CPP",
    "binaryPath": "/Users/kinleylam/Desktop/Local-CSC3002/CPP/kk",
    "binaryArgs": []
},
```

   - See next slide for a complete settings file.

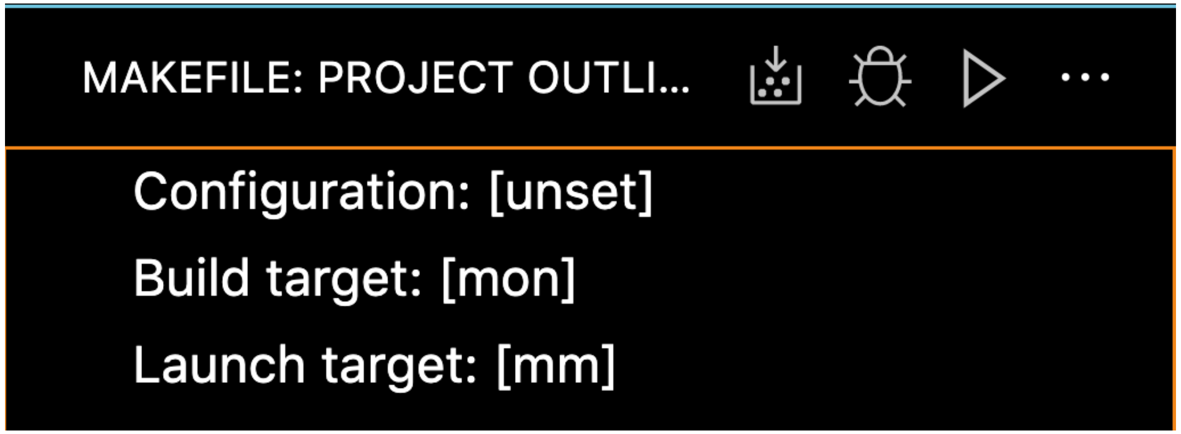
VSC – Settings (Executable targets)



```
{
    "makefile.launchConfigurations": [
        {
            "cwd": "/Users/kinleylam/Desktop/Local-CSC3002/CPP",
            "binaryPath": "/Users/kinleylam/Desktop/Local-CSC3002/CPP/mon",
            "binaryArgs": []
        },
        {
            "cwd": "/Users/kinleylam/Desktop/Local-CSC3002/CPP",
            "binaryPath": "/Users/kinleylam/Desktop/Local-CSC3002/CPP/tue",
            "binaryArgs": []
        },
        {
            "cwd": "/Users/kinleylam/Desktop/Local-CSC3002/CPP",
            "binaryPath": "/Users/kinleylam/Desktop/Local-CSC3002/CPP/wed",
            "binaryArgs": []
        },
        {
            "cwd": "/Users/kinleylam/Desktop/Local-CSC3002/CPP",
            "binaryPath": "/Users/kinleylam/Desktop/Local-CSC3002/CPP/thu",
            "binaryArgs": []
        },
        {
            "cwd": "/Users/kinleylam/Desktop/Local-CSC3002/CPP",
            "binaryPath": "/Users/kinleylam/Desktop/Local-CSC3002/CPP/kk",
            "binaryArgs": []
        },
        {
            "cwd": "/Users/kinleylam/Desktop/Local-CSC3002/CPP",
            "binaryPath": "/Users/kinleylam/Desktop/Local-CSC3002/CPP/mm",
            "binaryArgs": []
        },
    ],
```

VSC – Makefile Build and Run

6. Open the Makefile Extension, you will be able to connect the build and launch to your own Makefile



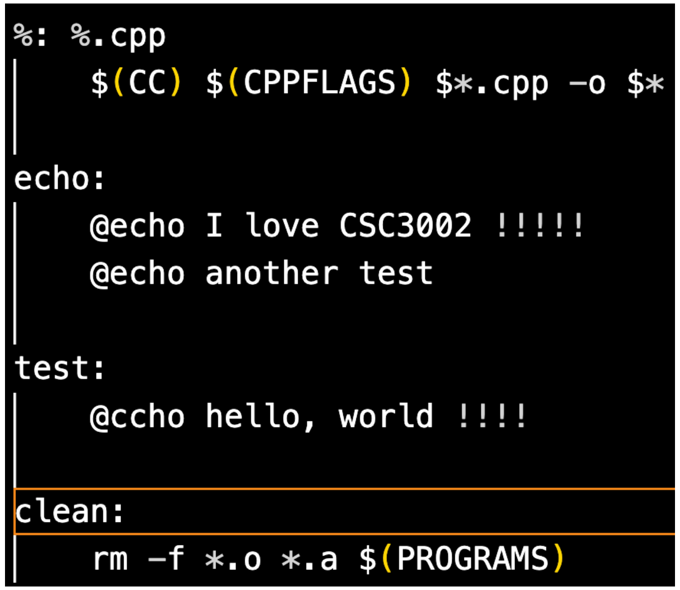
Configuration: [unset]
Build target: [mon]
Launch target: [mm]

- When you click the build button, it will build the target set (in this case, mon)
- When you click the launch button, it will run the target set (in this case, mm)

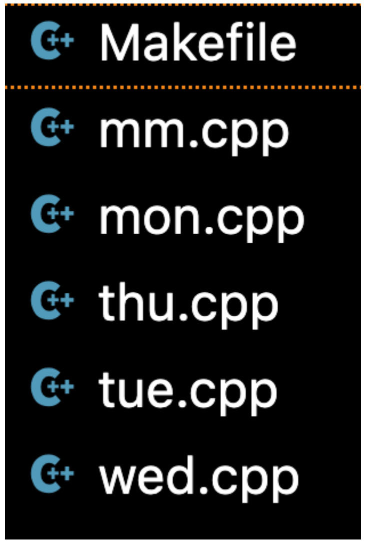
VSC - Demo

VSC – Makefile demo

- Setup:
  - In my home folder called CPP, I have created five standalone cpp files called mon.cpp, tue.cpp, wed.cpp, thu.cpp and mm.cpp, no header files used.
  - Targets in Makefile:



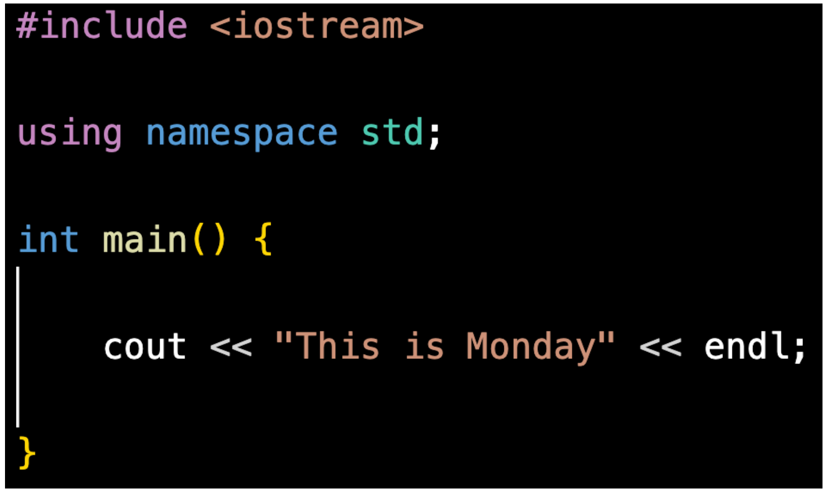
```
%: %.cpp
	$(CC) $(CPPFLAGS) $*.cpp -o $*

echo:
	@echo I love CSC3002 !!!!!
	@echo another test

test:
	@ccho hello, world !!!!

clean:
	rm -f *.o *.a $(PROGRAMS)
```



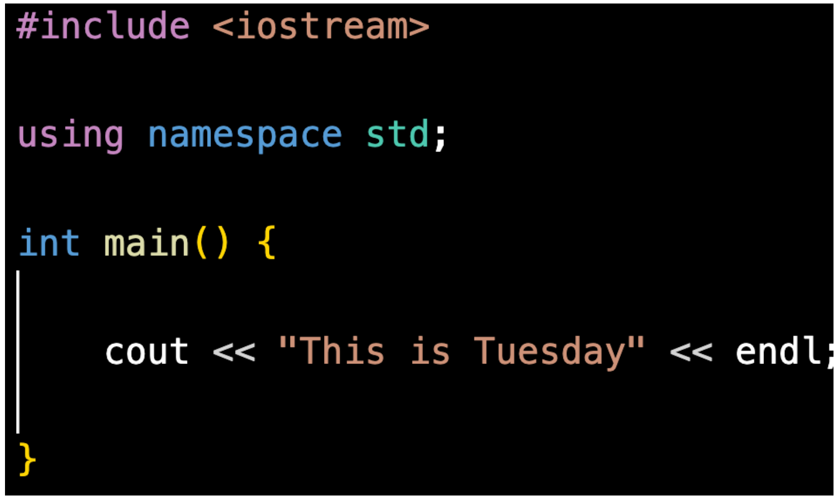
```
Makefile
mm.cpp
mon.cpp
thu.cpp
tue.cpp
wed.cpp
```

VSC – Makefile Demo

- Source file, each cpp file simply output a simple string on the console, such as:



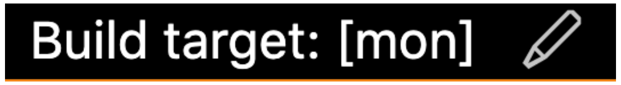
```cpp
#include <iostream>

using namespace std;

int main() {

    cout << "This is Monday" << endl;

}
```



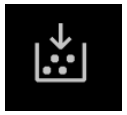
```cpp
#include <iostream>

using namespace std;

int main() {

    cout << "This is Tuesday" << endl;

}
```

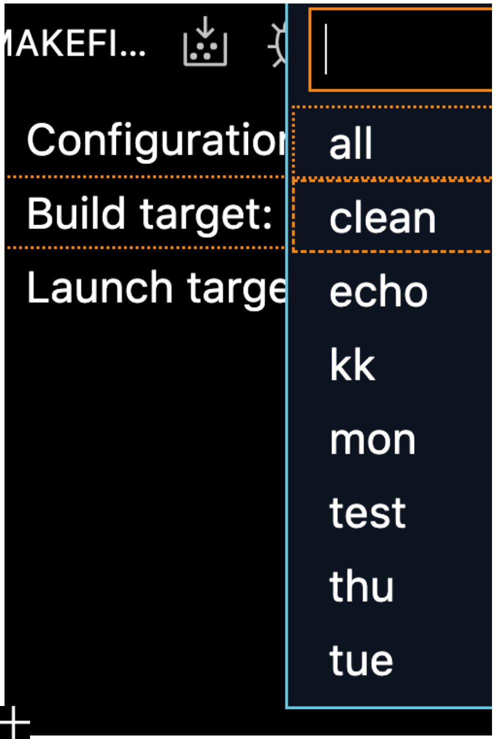
VSC – Makefile Build

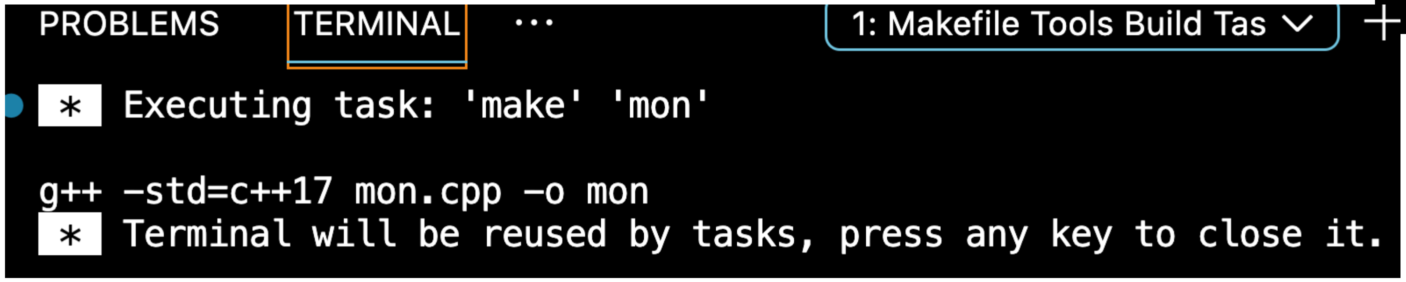
- Click the pencil icon to select the build target


Build target: [mon]

- To build, click the build button



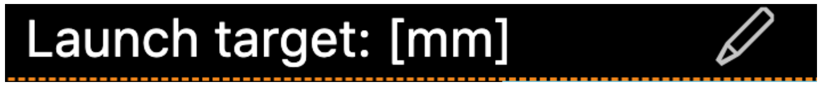




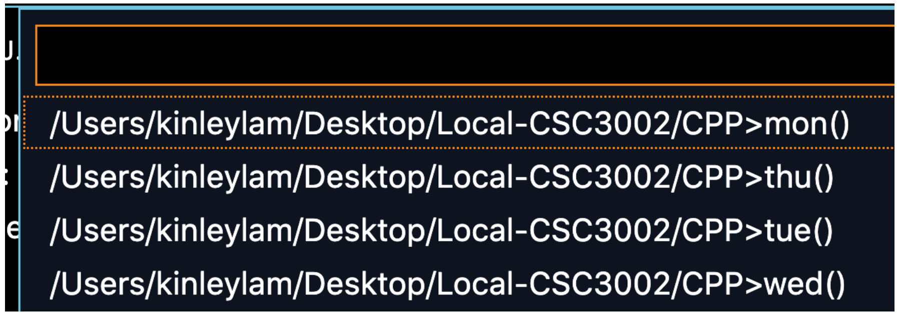
```
 *  Executing task: 'make' 'mon'

g++ -std=c++17 mon.cpp -o mon
 *  Terminal will be reused by tasks, press any key to close it.
```

VSC – Makefile Run

- Click the pencil icon to select the launch target.

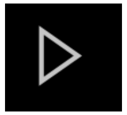


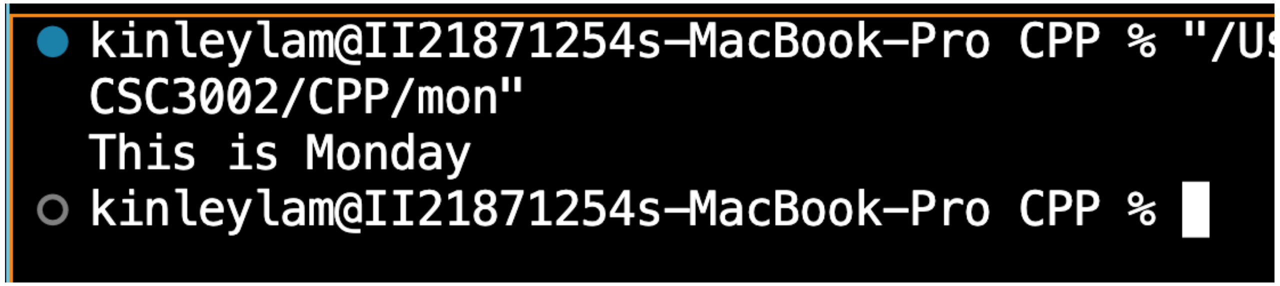
NOTE: The list matches exe files found in the folder.



```
/Users/kinleylam/Desktop/Local-CSC3002/CPP>mon()
/Users/kinleylam/Desktop/Local-CSC3002/CPP>thu()
/Users/kinleylam/Desktop/Local-CSC3002/CPP>tue()
/Users/kinleylam/Desktop/Local-CSC3002/CPP>wed()
```

- To run a program, click the launch button





```
● kinleylam@II21871254s-MacBook-Pro CPP % "/U
  CSC3002/CPP/mon"
  This is Monday
○ kinleylam@II21871254s-MacBook-Pro CPP % █
```