

VSC – 生成文件 (V1)

通过金利

任务

- C++ 编译器安装 · Make 实用程序安装 · 编译和链接 CPP 文件 ·

Makefile 概述

- 将 Makefile 与 Visual Studio Code 集成
- 演示

C++ 编译器 (Windows)

C++ 编译器 (Windows)

- 注意:对于Linux 和Mac 用户,C++ 编译器应该已经预先安装。
- 注意:C++ 编译器命令命名为 gcc、g++ 或 clang (mac)
- 使用的安装包名为MinGW-x64
- 安装基于[MSYS2](#),它提供 GCC、[Mingw-w64](#) 和其他有用的 C++ 工具和库的最新本地构建。
- 有关安装过程,请参阅下一张幻灯片

步骤 (Windows)

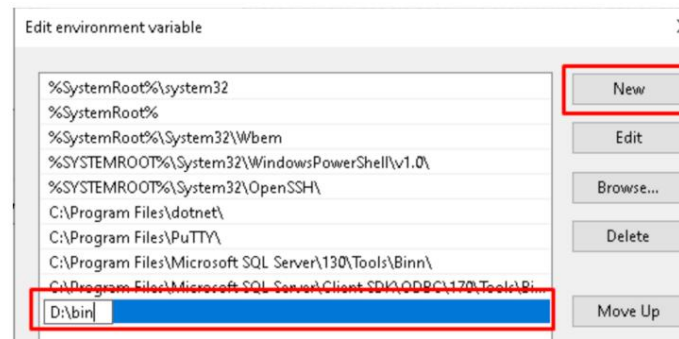
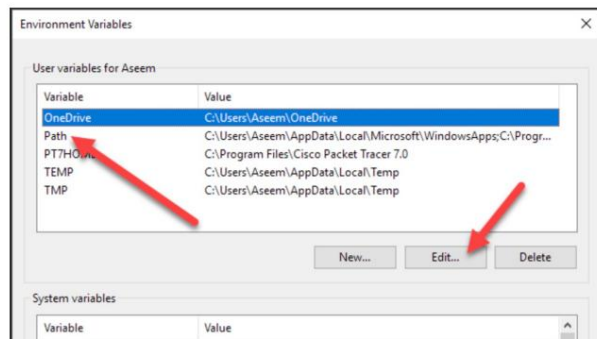
1. 基于MSYS2安装MinGW-x64

- 打开[MSYS2](#)网页并按照安装说明进行操作 · 注意安装文件夹（步骤 2 中需要） ·
MSYS2 完成后会弹出另一个窗口,您需要运行**pacman**来安装 C++ 编译器,如下所示:
- `$ pacman -S mingw-w64-x86_64-gcc`
- 确认 gcc (g++) 确实安装成功 (g++ --version)

- ## 2. 将 MinGW 编译器添加到您的路径 · 默认情况下，
- 路径应位于 `c:\msys64\mingw64\bin` 下 · 打开文件资源管理器以确认 g++ 编译器的位置。

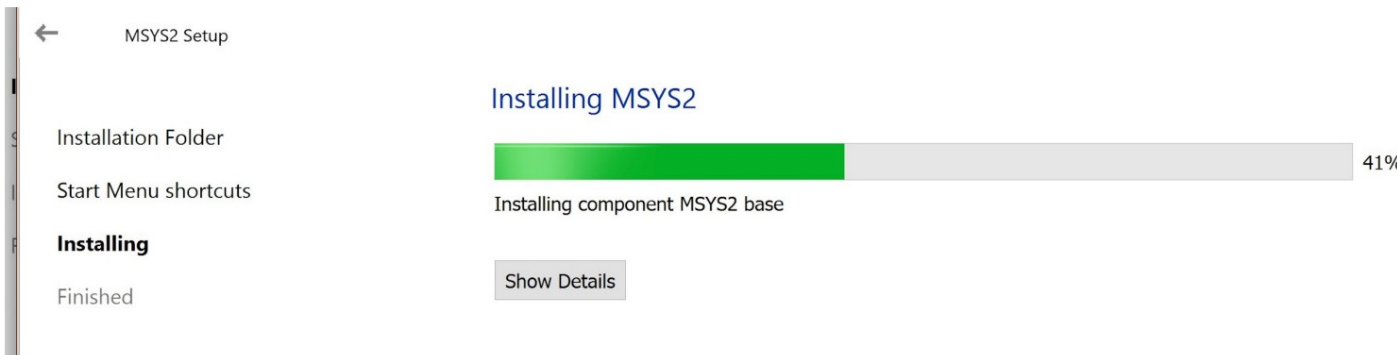
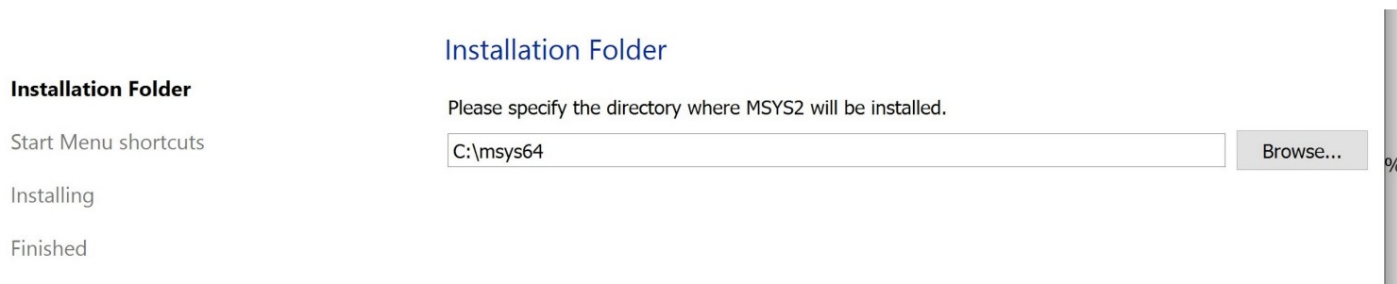
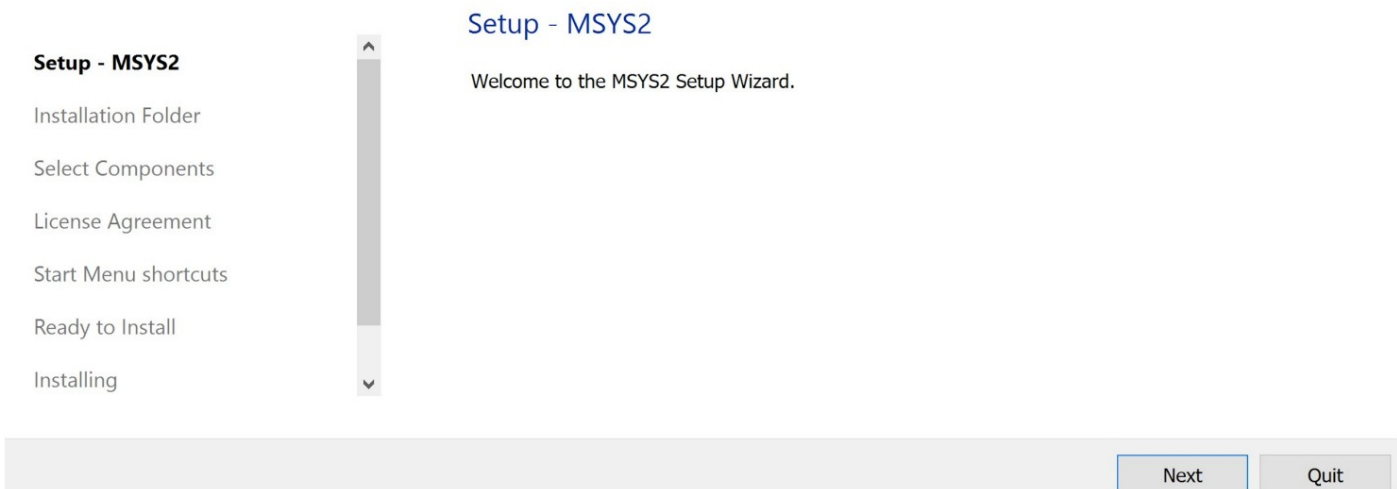
将 MinGW 编译器添加到 Windows 路径

- 在 Windows 搜索栏中,键入“设置”以打开您的 Windows 设置。
- 为您的帐户搜索编辑环境变量。
- 在用户变量中选择路径变量,然后选择编辑。



- 选择新建并将 Mingw-w64 目标文件夹路径 (例如:C:\msys64\mingw64\bin)添加到系统小路。具体路径取决于您安装的 Mingw-w64 版本以及安装位置。如果使用 DEFAULT 设置,路径可能是:C:\msys64\mingw64\bin。
- 选择确定保存更新的路径。您将需要重新打开任何控制台窗口以使新的 PATH 位置可用。输入 `g++ --version` 确认!!!

MSYS2 - 屏幕截图




```
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-zlib-1.2.13-1  
mingw-w64-x86_64-zstd-1.5.2-1
```

```
Total Download Size: 47.56
```

```
Total Installed Size: 397.60
```

```
:: 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-...
```

```
(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.
```

制作实用程序 (Windows)

制作实用程序 (Windows)

- 注意:对于Linux 和Mac 用户,Make 实用程序应该已经预先安装。
- 注意:Make 命令名为 make (Linux 和 Mac) ,Windows 不区分大小写。
- 使用 Windows 命令 “winget”安装 Make 实用程序

步骤 (Windows)

1. 以管理员权限运行 windows 控制台（以管理员身份运行）
2. 连接 VPN
3. 从控制台输入 `:winget install gnuwin32.make`
 - 参见下一张幻灯片的屏幕截图。
4. 将 Make 命令添加到您的路径

制作 截图

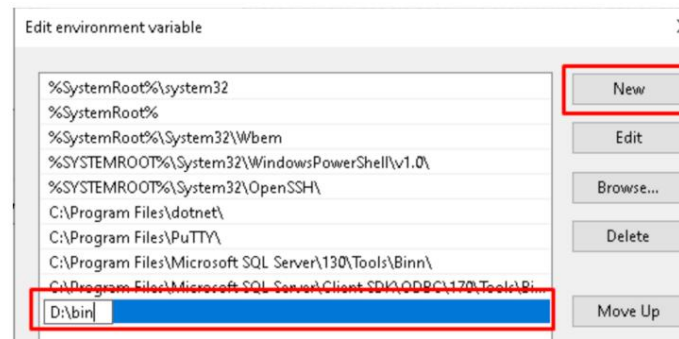
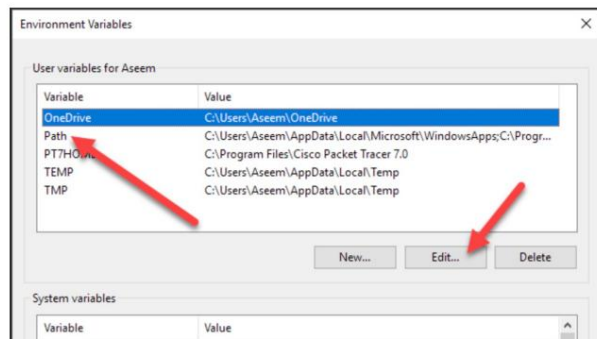
```
C:\WINDOWS\system32>winget install gnuwin32.make
The `msstore` source requires that you view the following agreements before using.
Terms of Transaction: https://aka.ms/microsoft-store-terms-of-transaction
The source requires the current machine's 2-letter geographic region to be sent to
rly (ex. "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.exe
████████████████████████████████████████████████████████████████████████████████ 3.22 MB / 3.22 MB
Successfully verified installer hash
Starting package install...
Successfully installed

C:\WINDOWS\system32>
```

将 Make 添加到 Windows 路径

- 在 Windows 搜索栏中,键入“设置”以打开您的 Windows 设置。
- 为您的帐户搜索编辑环境变量。
- 在用户变量中选择路径变量,然后选择编辑。



- 选择新建并将生成目标文件夹路径 (例如 :C:\Program Files(x86)\GnuWin32\bin) 添加到系统路径。确切的路径取决于您安装的 Make 版本以及安装位置。如果使用 DEFAULT 设置,路径可能是 :C:\Program Files(x86)\GnuWin32\bin。
- 选择确定保存更新的路径。您将需要**重新打开**任何控制台窗口以使新的 PATH 位置可用。输入 **make -version** 确认!!!

编译 & 链接 (Windows & Mac) - 高级

脚步

- 通常,您将所有 cpp 文件**编译**为目标文件 (.o),然后创建二进制或可执行文件 (称为**链接**)
- 假设:我们只有一个简单的 cpp 结构 (所有 cpp 和头文件在同一目录下,以及一个简单的头文件结构)
 - 编译单个 cpp 文件 (比如 foo.cpp) · `g++ -std=c++17 foo.cpp -c`
foo.o
 - 或者
· `g++ -std=c++17 foo.cpp -c` (目标文件的默认名称) · 创建
可执行文件 (称为链接) · `g++ -std=c++17 -o foo foo.o`
- 注意:您可以将编译与链接结合为一个命令: · `g++ -std=c++17 foo.cpp -o foo` · **此命令不会生成目标文件。**

多个文件

- 如果您的 C++ 程序由多个 cpp 文件组成,则需要分别编译每个 cpp,如下所示:
 - `g++ -std=c++17 <fileX>.cpp -c` ·
fileX 是 cpp 文件之一
- 接下来,将所有对象文件链接到可执行文件中,如下所示:
 - `g++ -std=c++17 -o <exe> file1.o file2.o file3.o ...`。
 - 文件1、文件2、文件3……是可执行程序所需的所有编译目标文件。

生成文件 (Windows 和 Mac)

Makefile - 高级别的

- Makefile 目标规则的一般语法如下：

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

- 注意:括号中的参数是可选的,省略号表示一个或多个。每个命令前都需要一个选项卡。

- 例子：

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

- 您可以在单个 Makefile 中定义许多目标规则。
- 执行目标：
 - 制作<目标>

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

宏 - Makefile

- 宏（自定义）
 - 您可以定义自己的宏（程序、对象）或覆盖默认值（CC、CPPFLAGS）。
- 特殊宏（内置）
 - `$@` 文件名（目标）
 - `$(*)` 文件前缀（目标）
 - `$(?)` 更改的家属的姓名。
 - `$(^)` 所有受抚养人的姓名（依赖关系）。
 - `$(<)` 导致该操作的相关文件的名称

```
$(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 - 通配符编译目标

- 对于简单的头文件和cpp 结构:一个cpp 文件对一个同名头文件,例如:
 - foo1.cpp foo1.h
 - foo2.cpp foo2.h ...。
- 然后您可以为所有 cpp 文件创建一个通用目标 (编译目标)作为一个单一目标,如下所示:

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

- 要编译任何 cpp 文件,请键入 “make <file>.o”

Makefile - 通配符链接目标

- 如果所有可执行文件都是基于单一源的（每个 cpp 文件都有自己的 main() 条目）。
- 然后您可以为所有 cpp 文件创建一个通用链接目标作为 ONE 单一目标如下：

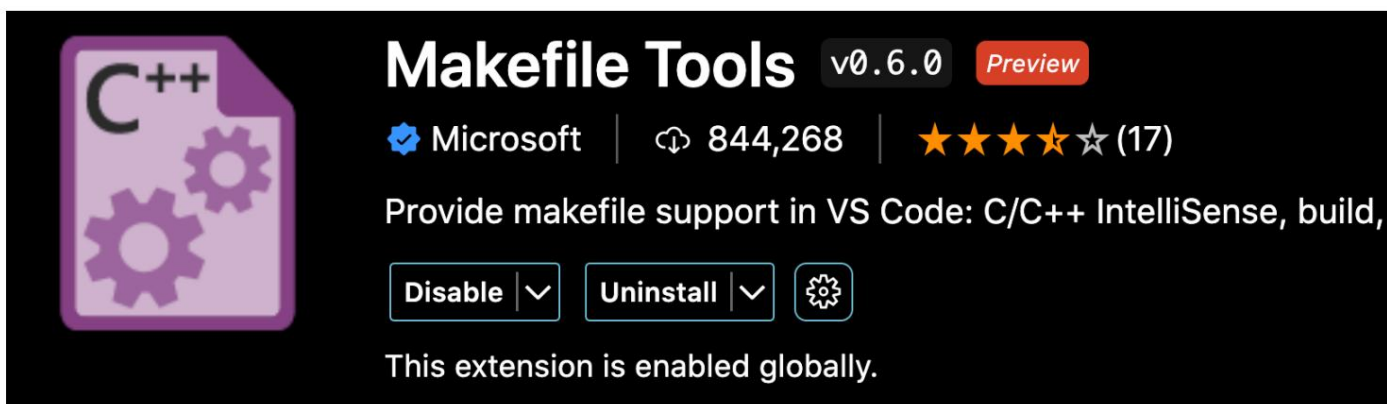
```
%: %.cpp  
| $(CC) $(CPPFLAGS) *.cpp -o $*
```

- 假设您创建了一个名为 hw.cpp 的独立 hello-world cpp 文件,要生成可执行文件,请运行“make hw”。

VSC - 生成文件扩展 (Windows 和 Mac)

先决条件（Windows 和 Mac）

- 将 Make 扩展安装到您的 VSC



- 您会在边栏上找到 Makefile 图标



步骤 - VSC 生成文件

1. 创建一个主文件夹（尽量简单,英文字母优先）
2. 运行VSC并打开主文件夹
3. 在您的主文件夹下创建一个makefile,例如makefile或Makefile

- 注意:如果您的 makefile 不在主文件夹下,您必须更改环境。

Makefile: Makefile Path

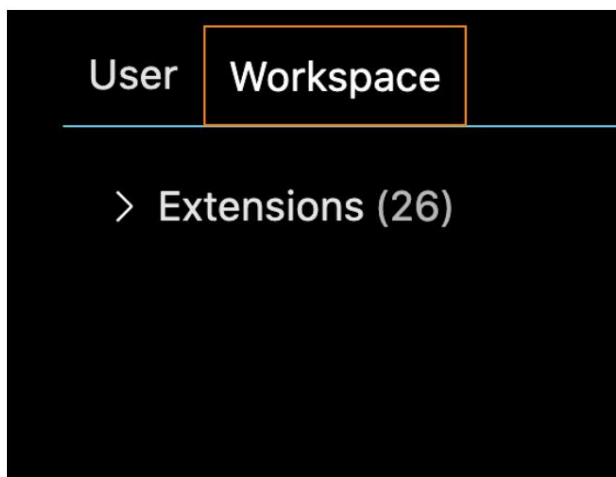
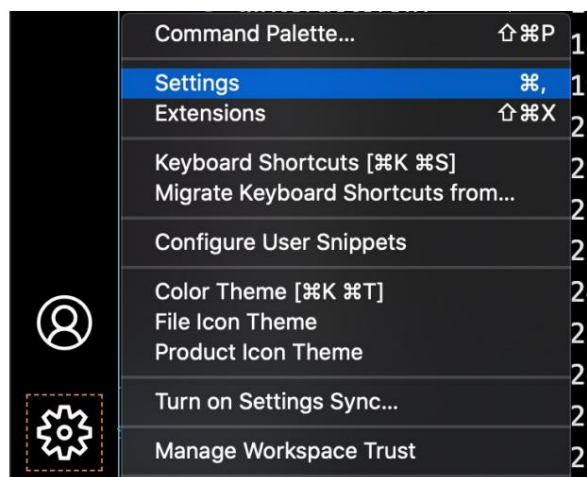
The path to the makefile of the project

- 为便于说明,makefile 位于主目录下

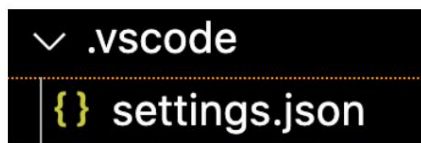
步骤 - VSC 生成文件

4. 创建工作区设置。

- 单击齿轮设置图标（左下角）并选择设置 · 选择工作区选项卡并单击打开的设置图标（右上角）



- 或者,在 .vscode 文件夹下,单击打开 settings.json（如果您已经创建了一个）



步骤 - VSC 生成文件

5. 创建可执行目标： · 创建

makefile.launchConfigurations 条目（如果未找到）：

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

- 定义可执行位置、名称和参数（在启动配置）：

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

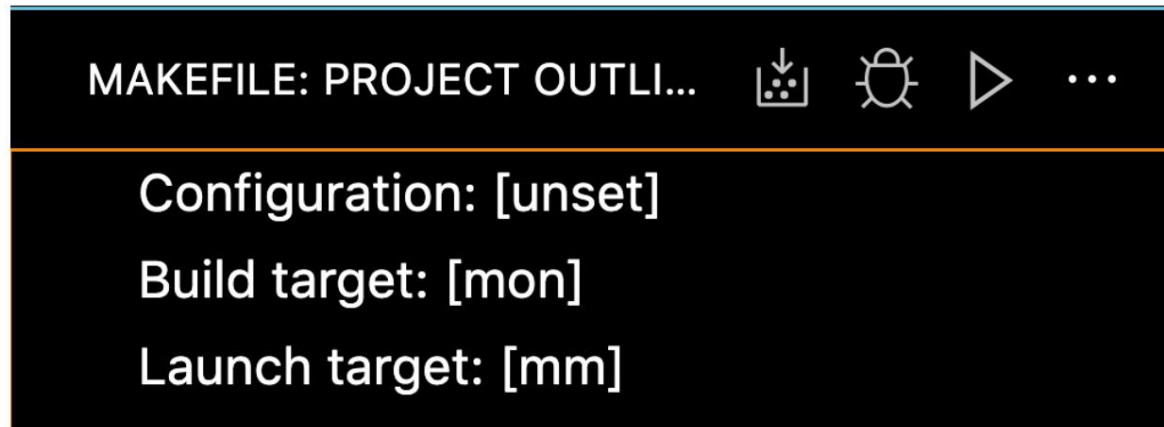
- 有关完整的设置文件,请参见下一张幻灯片。

VSC – 设置（可执行目标）

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

VSC – Makefile 构建和运行

6. 打开 Makefile Extension, 你就可以将构建连接到你自己的 Makefile 并启动



- 当您单击构建按钮时, 它将构建目标集 (在本例中为 mon)
- 当您单击启动按钮时, 它将运行目标集 (在本例中为 mm)

VSC - 演示

VSC – Makefile 演示

- 设置：
 - 在名为CPP的主文件夹中,我创建了五个独立的cpp文件,分别称为mon.cpp、tue.cpp、wed.cpp、thu.cpp和mm.cpp,没有使用任何头文件。
 - Makefile 中的目标:

```
%: %.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 演示

- 源文件,每个 cpp 文件只是在控制台上输出一个简单的字符串,如:

```
#include <iostream>

using namespace std;

int main() {
    cout << "This is Monday" << endl;
}
```

```
#include <iostream>

using namespace std;

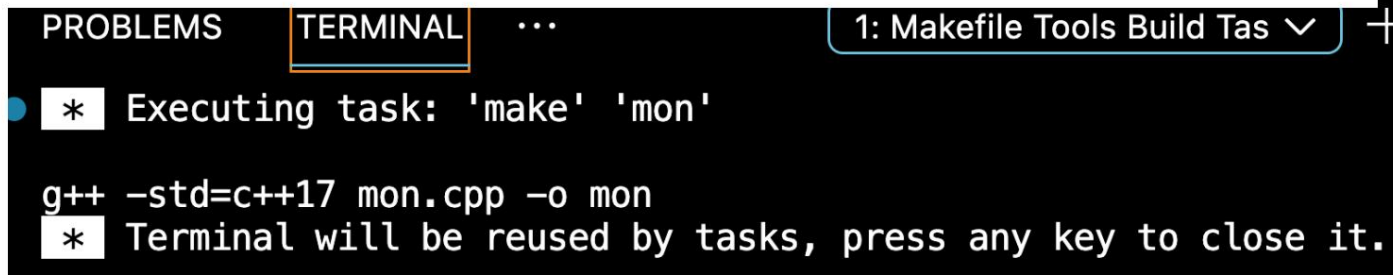
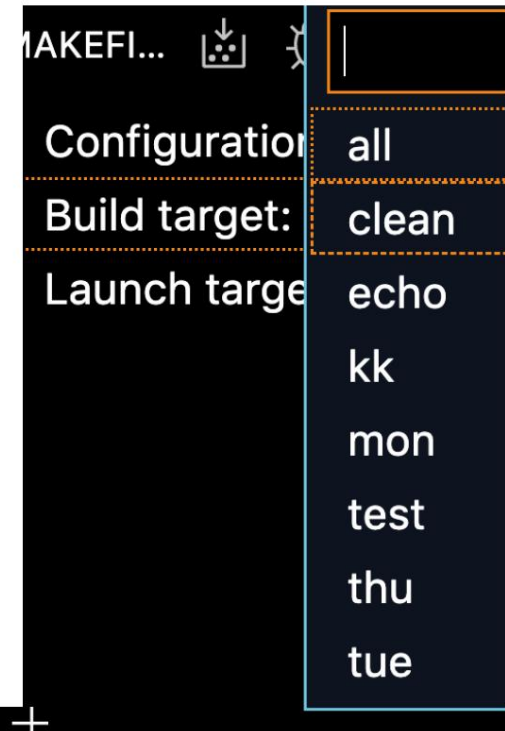
int main() {
    cout << "This is Tuesday" << endl;
}
```


VSC – 生成文件构建

- 单击铅笔图标选择构建目标

Build target: [mon] 

- 要构建,请单击构建按钮



VSC - 生成文件运行

- 单击铅笔图标以选择启动目标。

Launch target: [mm]



注意:列表匹配找到的 exe 文件
在文件夹中。

```
/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()
```

- 要运行程序,请单击启动按钮



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