C++: Compiling

Makefile

C++: g++ compiler

- **C++ compilers**
  - Using g++ or gcc compiler
    - gcc will compile: *.c/*.cpp files as C and C++ respectively.
    - g++ will compile: *.c/*.cpp files but they will all be treated as C++ files.
    - Also, g++ to link the object files it automatically links in the std C++ libraries (gcc does not do this).

C++: g++

— How to compile

- » Manually, it's a tedious process for medium to large projects
  - › **g++ source.cpp source1.cpp source2.cpp ………** –o output
- » g++ source.cpp –o output
  - › -g → turn on debugging (so GDB gives more friendly output)
  - › -Wall → turns on most warnings
  - › -O or -O2 → turn on optimizations
  - › -o <name> → name of the output file
  - › -c → output an object file (.o)

C++: Makefile

- **Why**
  - File to compile your programs in any environment UNIX/Windows
  - Provides a better framework for compiling source code
    - Rules format
      - **[name of rule]** : **[Dependency separated by spaces]**
        - [TAB] **command to execute for this rule**.
  - Name is of the file is "**makefile**" → no extensions

C++: Makefile

Example Rule

```
main: main.cpp
    g++ -c main.cpp
```

The files the rule is dependent on.
It can have multiple files. (*.cpp)

The command associated with
the rule

“target/name” of the rule. When called,
the following command is executed

C++: Makefile

— **Example**

```
all: main help
    g++ main.o help.o -o out
main: main.cpp
    g++ -c main.cpp
help: help.cpp
    g++ -c help.cpp
clean:
    rm *.o out
```

$ make

1. When you type “make” the first rule is executed - *all*

2. The ***all*** is dependent on two rules
main and **help** (order is important)

3. The ***main*** is not dependent on any rules, but main.cpp file. This rule’s command is executed.

4. The ***help*** is not dependent on any rules, but help.cpp file. This rule’s command is executed.

5. Finally, the command under ***all*** rule is executed.

C++: Makefile

- **Example**

```
all: main help
    g++ main.o help.o -o out
main: main.cpp
    g++ -c main.cpp
help: help.cpp
    g++ -c help.cpp
clean:
    rm *.o out
```

You can also invoke a specific rule

```
$ make clean
```

This will invoke clean rule in the make file.
Here we are removing any “.o) files and main.o file.
“rm” mean remove or delete

C++: Makefile

- **What if you use different compilers or flags**
  - Example
    - g++ -v -Wall –O2 main.o help.o -o main.exe
  - Now you want to change g++ → gcc and add/modify/change your flags
    - How do you want do it
  - You use **VARIABLES**

C++: Makefile

- **Define Variables**
  - Example
    - CC=g++ # CC is variable with value g++
    - CFLAGS=-g –Wall –O2 –v # all the required flags
  - How to access them
    - $(CC) $(CFLAGS) -c string.cpp
  - Equivalent to
    - g++ -g –Wall –O2 –v –c string.cpp
  - This way you have manage your compiler and its flags in one location

C++: Makefile

- **More Examples**

```
CC=g++
CFLAGS=-g -Wall
RM=rm -f
all: main.o string.o
        $(CC) $(CFLAGS) -o main main.o frog.o
#create the object file for string.cpp
string.o: string.h string.cpp
$(CC) $(CFLAGS) -c frog.cc
#create the object file for the main file
main.o: string.h main.cpp
$(CC) $(CFLAGS) -c main.cc
#create stringexample
stringexample: string.cpp
$(CC) $(CFLAGS) -o helloworld helloworld.cc
# rule for cleaning files generated during
compilations. Call 'make clean' to #use it
clean: $(RM) *.o main
```

Thank You

Questions, Comments & Feedback