SCM513: Introduction to Scientific Computing
Part 2: C Programming
Compiling

Elliot Menkah

elliotsmenkah@gmail.com

November 29, 2022

National Institute for Mathematical Sciences Ghana

Outline

1 Introduction
2 Compiler
3 Case Study
4 Compiling
5 Makefile

National Institute for Mathematical Sciences Ghana

where are we?

1. Introduction
2. Compiler
3. Case Study
4. Compiling
5. Makefile

National Institute for Mathematical Sciences Ghana

Compiling in C

- Compile to object code
- Compile to executable
- You can compile directly to executable
- You can also compile to a library

National Institute for Mathematical Sciences Ghana

where are we?

1 Introduction
2 Compiler
3 Case Study
4 Compiling
5 Makefile

National Institute for Mathematical Sciences Ghana

C Compiler and Flags

C Compiler and some common flags

gcc	C Compiler
-I	For including paths to the system path.
-l	For including libraries.
-o	For specifying output.
-c	For specifying compile source without linking so generate object (.o) files.

National Institute for Mathematical Sciences Ghana

where are we?

1 Introduction
2 Compiler
3 Case Study
4 Compiling
5 Makefile

National Institute for Mathematical Sciences Ghana

Case Study: header

myfunc.h : header file for describing our program function

```
1 /*
2 example include file
3 */
4
5 void welcomeMessage(void);
```

National Institute for Mathematical Sciences Ghana

Case Study: function

myfunc.c : program file for implementing our program function

```
1 #include <stdio.h>
2 #include <myfunc.h>
3
4 void welcomeMessage(void) {
5
6    printf("Hello NIMS Fellows!\n");
7
8    return;
9 }
```

National Institute for Mathematical Sciences Ghana

Case Study: program

myprog.c : program file for executing our program function

```c
#include <myfunc.h>

int main() {
   // call a function in another file
   welcomeMessage();

   return(0);
}
```

National Institute for Mathematical Sciences Ghana

where are we?

1 Introduction
2 Compiler
3 Case Study
4 Compiling
5 Makefile

National Institute for Mathematical Sciences Ghana

Compiling with gcc

Compiling

```
1 gcc -o myprog myprog.c myfunc.c -I.
```

- This compiles the two .c files and names the executable myprog.
- The -I. is for defining paths to include for header files
- In this case the current folder (.) is included in system path

National Institute for Mathematical Sciences Ghana

where are we?

1. Introduction
2. Compiler
3. Case Study
4. Compiling
5. Makefile

National Institute for Mathematical Sciences Ghana

Using Makefile: 1

Makefile

```
1 myprog: myprog.c myfunc.c
2 	gcc -o myprog myprog.c myfunc.c -l.
```

- Makefile is a compilation script used by make.
- It saves repeating compile instructions
- COMMANDS: There must be a tab at the begining of every command
- RULES: left side of : is output, right side is input

National Institute for Mathematical Sciences Ghana

Using Makefile: 2

Makefile

```
1 CC=gcc
2 CFLAGS=-I.
3
4 myprog: myprog.o myfunc.o
5       $(CC) -o myprog myprog.o myfunc.o -I.
```

- Constants can be defined and used.
- CC is the macro for defining the compiler to use
- CFLAGS is the list of flags to pass to the compiler command
- By putting the object files (myprog.o and myfunc.o) in the dependency list and in the rule, *make* knows it must first compile the .c versions individually, before building the executable

National Institute for Mathematical Sciences Ghana

Using Makefile: 3

Makefile

```
1 CC=gcc
2 CFLAGS=-I.
3 DEPS = myfunc.h
4
5 %.o: %.c $(DEPS)
6         $(CC) -c -o $@ $< $(CFLAGS)
7
8 myprog: myprog.o myfunc.o
9         gcc -o myprog myprog.o myfunc.o -I.
```

- The DEPS macro defines the set of .h files on which the .c files depend. This ensures that changes in .h files are recompiled
- The .o files depend on their .c versions and the .h files included in the DEPS macro.
- To generate the .o file, *make* needs to compile the .c file using CC.
- The -c flag means generate the object file,
- The **-o $@** means put the output in the file named on the left side of the :
- The **$¡** is the first item in the dependencies list.

National Institute for Mathematical Sciences Ghana

Using Makefile: 4

Makefile

```
1  CC=gcc
2  CFLAGS=-I.
3  DEPS = myfunc.h
4  OBJ = myprog.o myfunc.o
5
6  %.o: %.c $(DEPS)
7          $(CC) -c -o $@ $< $(CFLAGS)
8
9  myprog: $(OBJ)
10         gcc -o $@ $^ $(CFLAGS)
```

- All of the include files should be listed as part of the macro DEPS
- All of the object files should be listed as part of the macro OBJ
- $@ : Left side macro. Inserts left side of rule (myprog)
- $: Right side macro. Inserts right side of rule ($(OBJ) = myprog.o myfunc.o). Compare with step 3

National Institute for Mathematical Sciences Ghana

Using Makefile: 5

Makefile

```
1  IDIR =../include
2  CC=gcc
3  CFLAGS=-I$(IDIR)
4
5  ODIR=obj
6  LDIR =../lib
7
8  LIBS=-lm
9
10 _DEPS = myfunc.h
11 DEPS = $(patsubst %,$(IDIR)/%,$(_DEPS))
12
13 _OBJ = myprog.o myfunc.o
14 OBJ = $(patsubst %,$(ODIR)/%,$(_OBJ))
15
16
17 $(ODIR)/%.o: %.c $(DEPS)
18         $(CC) -c -o $@ $< $(CFLAGS)
19
20 myprog: $(OBJ)
21         gcc -o $@ $^ $(CFLAGS) $(LIBS)
22
23 .PHONY: clean
24
25 clean:
26         rm -f $(ODIR)/*.o *~ core $(INCDIR)/*~
```

- Put all header .h files in an include directory
- Put all source .c files in a src directory
- Put all local libraries in a lib directory
- Hide intermediary .o files
- This makefile should be located in the src directory
- The .PHONY rule keeps *make* from doing something with a file named clean
- clean: Rule for cleaning intermediary files using *make clean*

Special thanks to the following for their initial work:

- *Kwesi A. Smith*
- *Shirley A. Akasreku*