# CSCI 2132 – Software Development

## Writing Large Programs

March 22, 2017

Dalhousie University

Meng He

## Header Files

- Files that allow different source files (*.c) to share
  - Function prototypes
  - Type definitions
  - Macro definitions
  - …
- Naming convention: *.h

## The #include Directive

- Tells the preprocessor to open a specified file and inserts its content into the current file
- Form 1: #include <file name>
  - Search the directory/directories in which system header files reside
    - Bluenose: /usr/include, …
- Form 2: #include “file name”
  - First search the current directory, if not found then
  - directories in which system header files reside
- Question: Which form for your own header files?

## Dividing a Program into Files

- Example: decimal2binary

- Step 1: Breaking program logically into source files (*.c)
  - decimal2binary.c: the main program
  - stack.c: Stack implementation

## Step 2: Sharing…

- Sharing type definitions
  - bit.h:
    typedef int Bit;
- Sharing macro definitions
  - Not needed as STACK_SIZE is used by stack.c only
- Sharing function prototypes
  - stack.h
- Advantages of using both bit.h and stack.h instead of one header file:
  - bit.h cold be used by another program

## Step 3: Protecting Header Files

- Issue: nested header files
  - Example:

| stack.h: | stack.c |
|---|---|
| … | … |
| #include “bit.h” | #include “bit.h” |
| … | #include “stack.h” |

  - Can you see a problem?

- Solution: Protect each header file using conditional compilation

## Conditional Compilation

- Example (bit.h)

```
#ifndef BIT_H
#define BIT_H
typedef int Bit;
#endif
```

- Meaning

  If BIT_H is not defined

  Define BIT_H

  Include the code up to:

  #endif

bit.h:

```
#ifndef BIT_H
#define BIT_H

typedef int Bit;

#endif
```

stack.h:

```
#ifndef STACK_H
#define STACK_H

#include <stdbool.h>

#include "bit.h"

void make_empty(void);
bool is_empty(void);
bool is_full(void);
void push(Bit i);
Bit pop(void);
void stack_overflow(void);
void stack_underflow(void);

#endif
```

stack.c:

```
#include <stdio.h>
#include <stdlib.h>

#include "bit.h"
#include "stack.h"

#define STACK_SIZE 100

Bit contents[STACK_SIZE];
int top = 0;


void make_empty(void) {
  top = 0;
}
```

…and other stack function definitions

decimal2binary.c:

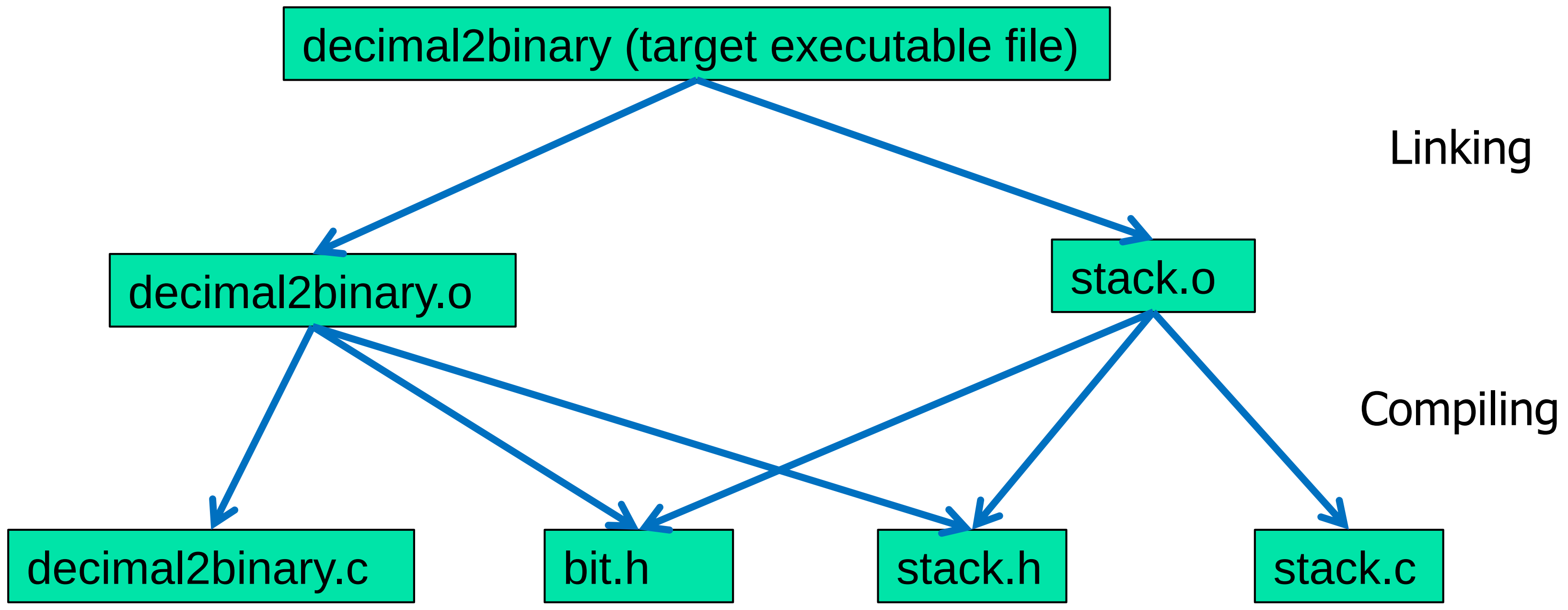
```
#include <stdio.h>

#include "bit.h"
#include "stack.h"

int main(void) {
  int decimal;
  Bit bit;

  printf("Enter a decimal integer: ");
  scanf("%d", &decimal);

  while (decimal > 0) {

  …rest of the main function
}
```

# The Make Utility

- How do we compile these files?
- The make utility
  - Manages the compilation and linking of multi-file software
  - Reads a makefile (name: makefile or Makefile) that specifies
    - The targets to be built
    - Commands used to build them
    - How the modules of a software system depend on each other (key)

## Dependencies

- A directed, acyclic graph
- Object file (*.o): a file containing machine instructions of one module
- We typically generate one object file for each *.c file



makefile:

```
all: decimal2binary hello

decimal2binary: decimal2binary.o stack.o
	gcc -std=c99 -o decimal2binary decimal2binary.o
stack.o

decimal2binary.o: decimal2binary.c stack.h bit.h
	gcc -std=c99 -c decimal2binary.c

stack.o: stack.c stack.h bit.h
	gcc -std=c99 -c stack.c

hello: hello.c
	gcc -o hello hello.c

clean:
	rm decimal2binary decimal2binary.o stack.o hello
```

# More About the makefile

- White space before each gcc command:
  - Exactly one tab character!
- The –c option: compiling without linking

## Some make Commands

- make
  - Make first target
  - all:decimal2binary, hello
- make target
  - make decimal2binary
  - make all
  - make clean

## More about make

- If a file is edited, only those that depend on it will be rebuilt
  - Think of a program of 100 *.c *h files (you might have to use this many files for the 4th-year compiler course, depending on the instructor expectation)
- Make for gdb
  - -g for all gcc commands
  - break filename:line_number
  - break filename:function_name (filename: can be omitted)