Programming for Engineers Multiple Source Files

UNIVERSITY AT ALBANY
State University of New York

ICEN 200 – Spring 2018

Prof. Dola Saha

C Source Files

- A C program may be divided among any number of ***source files.***
- By convention, source files have the extension `.c`.
- Each source file contains part of the program, primarily definitions of functions and variables.
- One source file must contain a function named `main`, which serves as the starting point for the program.

Advantage of Splitting

- Splitting a program into multiple source files has significant advantages:
  - Grouping related functions and variables into a single file helps clarify the structure of the program.
  - Each source file can be compiled separately, which saves time.
  - Functions are more easily reused in other programs when grouped in separate source files.

Header

- Problems that arise when a program is divided into several source files:
  - How can a function in one file call a function that's defined in another file?
  - How can a function access an external variable in another file?
  - How can two files share the same macro definition or type definition?
- The answer lies with the `#include` directive, which makes it possible to share information among any number of source files.

Header

- The `#include` directive tells the preprocessor to insert the contents of a specified file.
- Information to be shared among several source files can be put into such a file.
- `#include` can then be used to bring the file's contents into each of the source files.
- Files that are included in this fashion are called ***header files*** (or sometimes ***include files***).
- By convention, header files have the extension `.h`.

MACRO

- Most large programs contain macro definitions and type definitions that need to be shared by several source files.
- These definitions should go into header files.

Example MACRO

- Suppose that a program uses macros named `BOOL`, `TRUE`, and `FALSE`.
- Their definitions can be put in a header file with a name like `boolean.h`:

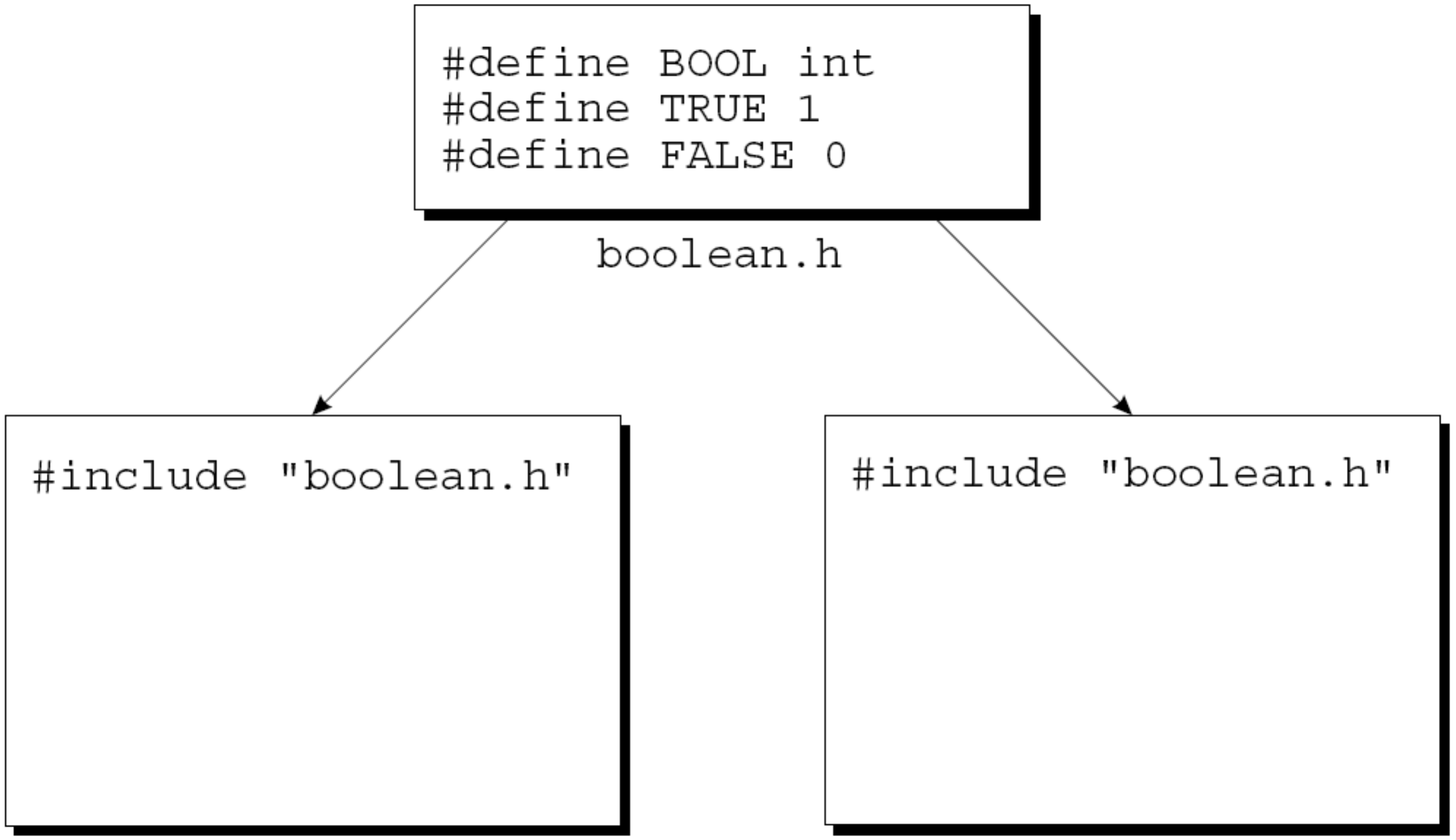
```
#define BOOL int
#define TRUE 1
#define FALSE 0
```

- Any source file that requires these macros will simply contain the line

```
#include "boolean.h"
```

Example Sharing MACRO

- A program in which two files include `boolean.h`:



```
#define BOOL int
#define TRUE 1
#define FALSE 0
```

```
#include "boolean.h"
```

```
#include "boolean.h"
```

MACRO Sharing – Why?

- Advantages of putting definitions of macros and types in header files:
  - Saves time. We don’t have to copy the definitions into the source files where they’re needed.
  - Makes the program easier to modify. Changing the definition of a macro or type requires editing a single header file.
  - Avoids inconsistencies caused by source files containing different definitions of the same macro or type.

Sharing Function Prototype

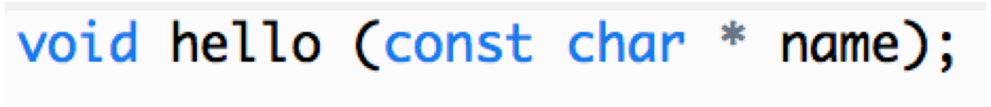
- Suppose that a source file contains a call of a function `f` that's defined in another file, `foo.c`.
- Calling `f` without declaring it first is risky.
  - The compiler assumes that `f`'s return type is `int`.
  - It also assumes that the number of parameters matches the number of arguments in the call of `f`.
- So, we put `f`'s prototype in a header file (`foo.h`), then include the header file in all the places where `f` is called.
- We'll also need to include `foo.h` in `foo.c`, enabling the compiler to check that `f`'s prototype in `foo.h` matches its definition in `foo.c`.

Sharing Variable

- To share a variable among files, we put its *definition* in one source file, then keyword `extern` is used to declare a variable without defining it.
- For example,
  - `int i; // in file1.c`
  - `extern int i; // in file2.c`
- `extern` informs the compiler that `i` is defined elsewhere in the program, so there's no need to allocate space for it.

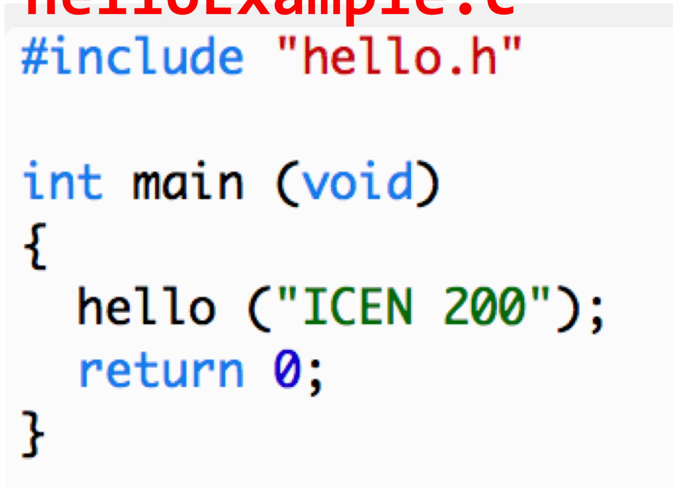
Compiling Multiple Source Files

hello.h



```c
void hello (const char * name);
```

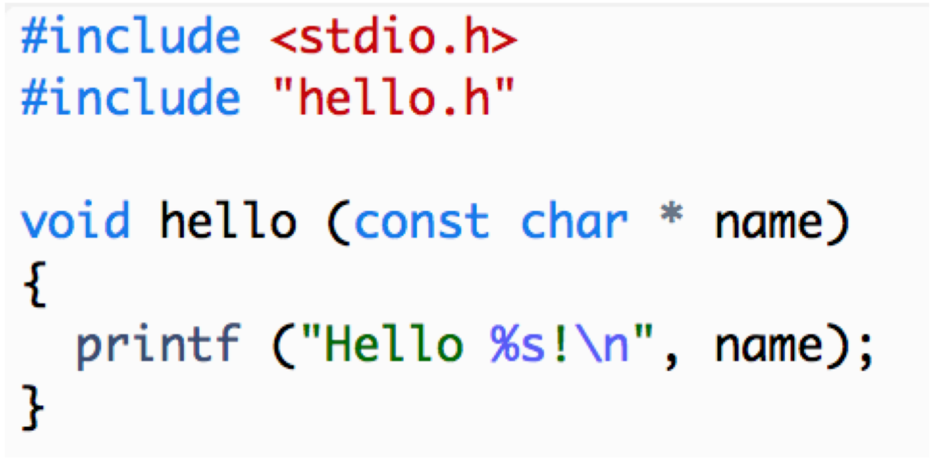
helloExample.c



```c
#include "hello.h"

int main (void)
{
  hello ("ICEN 200");
  return 0;
}
```

helloFn.c



```c
#include <stdio.h>
#include "hello.h"

void hello (const char * name)
{
  printf ("Hello %s!\n", name);
}
```

```
$gcc helloExample.c helloFn.c -o hello
$./hello
Hello ICEN 200!
$
```

UNIVERSITY AT ALBANY
State University of New York