Managing the Implementation of C Projects

Multiple Files

- Why multiple files?
  - Modularity!
- Execution of a program begins in the **main** function. The **main** function can call other functions
  - Functions defined in the same file
  - Function defined in other files or libraries
- A module is a collection of related functions
- Review:
  - Function Prototypes
  - Header Files

Function Prototypes (Declarations)

- Informs compiler about a function's return type, name, and parameter types
  - Used to check whether function is called correctly
- Example

  ```
  void FunctionDefinedLater(int);
  ```

- Should be placed at the top of the file (after the #include directives)

Example of Function Prototype

```
int doNothing(int y);   Function
                        Prototype

int main(void) {
 printf("%d", doNothing(10));
}

int doNothing(int y) {   Function
 return y;               Definition
}
```

Header Files

- Typically contain:
  - Function prototypes for the module

    `int doNothing(int);`
  - Constants (#define)

    `#define MAX 80`
  - Structures and external variables

    `extern char line[MAX];`

5

Modular Project Example

utils.h

```
#ifndef UTILS_H
#define UTILS_H

#define MAX 80
extern char line[MAX];

void startup();
void help(char * line);

#endif
```

utils.c

```
#include "utils.h"

void startup()
{ ... }

void help(char * line)
{ ... }
```

smartshell.c

```
#include "utils.h"

int main()
{
  startup();
  ...
}
```

6

Using Conditionals in Header Files

- Example:

```
#ifndef CALC_H
#define CALC_H

/* Insert header file info */

#endif
```

- Prevents multiple inclusions of header files, which may cause compilation problems

7

Summing Up …

- A C project can have only one main module (or file) with its main function
- To use functions defined in other modules, add the function prototype to a header file and include the header file:

```
#include "utils.h"
```

8

Another Example

calc.h
```
#ifndef CALC_H
#define CALC_H

int square(int x);

#endif /* CALC_H */
```

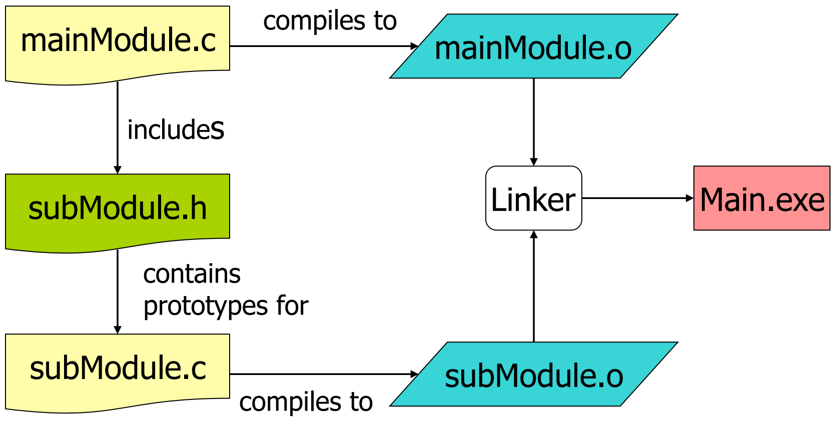
calc.c
```
#include "calc.h"

int square(int x) {
  return x*x;
}
```

driver.c
```
#include <stdio.h>
#include "calc.h"

int main() {
  int x = 5;

  printf("\nSquare of %d is %d\n", x, square(x));
  return 0;
}
```

More on #include

- #include <some.h>
  - searches the system path (the directories stored in the PATH environment variable) for some.h
- #include "some.h"
  - searches the current directory only
- -Idir (I is "capital i" and stands for "Include")
  - Use this directive in the compile command
  - appends dir to list of directories searched for header files and libraries for both " " and < >;

Files (.h,.c,.o,.exe)



Incremental Compilation

- Run

  gcc –c

  on sub-modules to generate object files
- Run

  gcc mainModule.o subModule.o

  to build final executable

Advantages of using Modules

- Modules can be written and tested separately
- Large projects can be developed in parallel
- Reduces length of program, making it more readable
- Promotes the concept of abstraction

13

Hands-On:

- Reorganize your smartshell project modularly:
  - smartshell.c should contain the main function only
  - environment.c should contain the functions related to the PATH environment (commands “path” and “where”)
  - history.c should contain the functions related to the shell history (commands “history” and “run”)
  - trash.c should contain the functions related to the trash (commands “trash”, “delete”, “undelete” and “empty”)
  - utils.c should contain all the other functions
- Write a header file for each of module
- Include the header files only where needed
- Work on ONE MODULE AT A TIME!

14