LECTURE NOTES IN CIS300

YUZHE (RICHARD) TANG

SPRING, 2018

SECTION 2: C/C++ PROGRAMMING

REFERENCES

- "Unix Programming Tools", [link]
- Computer Systems: A Programmer's Perspective, Randal E. Bryant and David R. O'Hallaron, Chapter 1, [online pdf]

HELLOWORLD C

```c
#include <stdio.h> //preprocessor
int y = 3; //global var. (def. & init.)
//extern int y; //global var. (dec.)
int main() //function (def.)
{
    int x = 0; //local var. (def. & init.), literal,
    printf("helloworld: y = %d\n",y); //function (invocation)
    return 0;
}
```

- printf: format string
- header files

LIFE OF A C CONSTRUCT

	variable	function
declare	`extern int x;`	`void foo();`
define	`int x;`	`void foo(){}`
initialize	`int x=6;`	
reference	`y=x;x=1;`	`foo();` (invocation)
destroy		

COMPILATION & EXECUTION: BASICS

- GCC: GNU Compilation Collection
- In your terminal, run the following commands

```
gcc hello.c
./a.out
```

EXERCISES

- Write a C program that prints out your name. Compile and execute it in Ubuntu. Submit the C program to BB.
- Write a C program that computes the sum of 1,2,3,...,956. Compile and execute the program in Ubuntu. Submit the C program to BB.

GCC

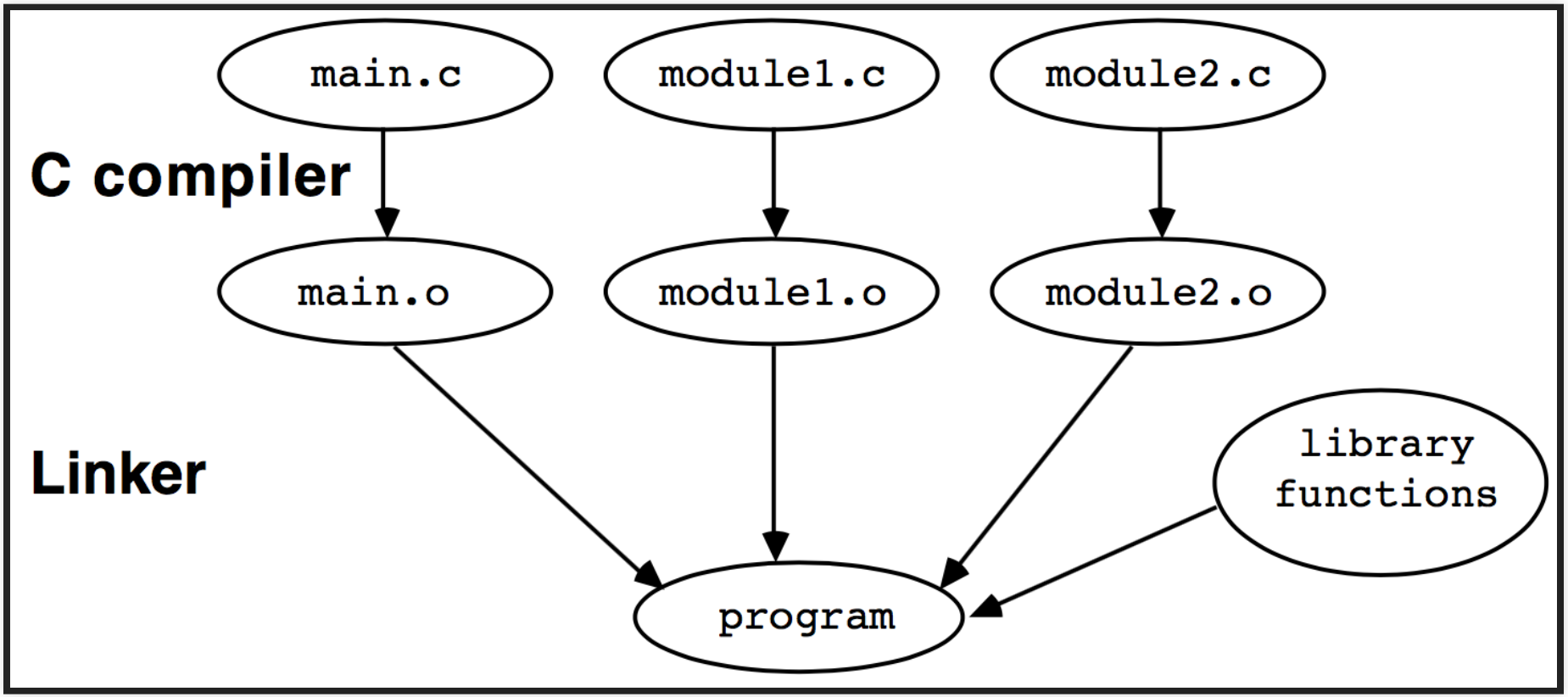
COMPILATION (1)

- Two steps of compilation:
  - *compiling*: text `.c` file to relocatable `.o` (object) file
  - *linking*: multiple relocatable `.o` files to one executable `.o` file
    - *symbol*: reference to link construct (declaration) in one `.o` file to construct (definition) in another `.o` file

COMPILATION (2)

```
gcc hello.c -o a.out
gcc -S hello.c -o hello.s #compiler
gcc -c hello.s -o hello.o #assembler
gcc hello.o -o a.out #linker
```

- compilation system
  - tools: *gcc/gdb* for compiling and debugging
  1. **preprocessor**: from source file to source
  2. **compiler**: from source to assembly file
     - *assembly file*
  3. **assembler**: from assembly file to relocatable object file
  4. **linker**: from multiple objects to an executable object


main.c
module1.c
module2.c
C compiler
main.o
module1.o
module2.o
Linker
library
functions
program

Linker

COMPILING MULTIPLE C PROGRAMS

In `file1.c`:

```c
#include <stdio.h>
extern void foo();
int main(){
    printf("main();\n");
    foo();
}
```

In `file2.c`:

```c
#include <stdio.h>
void foo(){
    printf("foo();\n");
}
```

COMPILING MULTIPLE C PROGRAMS (2)

```
gcc file1.c file2.c
# try this?
gcc file1.c
gcc file2.c
```

COMPILING MULTIPLE C PROGRAMS (3)

```
gcc -c file1.c # compiler & assembler
gcc -c file2.c # compiler & assembler
gcc file1.o file2.o # linker
```

Or

```
gcc -S file1.c # compiler
gcc -c file1.s # assembler
gcc -S file2.c # compiler
gcc -c file2.s # assembler
gcc file1.o file2.o # linker
```

LINK LIBRARY FILES

```
gcc -S file1.c # compiler
gcc -c file1.s # assembler
gcc file1.o file2.o # linker
```

```
mv file2.o ../libfile2.a
gcc file1.o ../libfile2.a # linker
gcc file1.o -L.. file2.o # linker
gcc file1.c -L.. file2.o # linker
```

- Gcc flag: `-Ldir -lmylib` for library to link

INCLUD HEADER FILE

In `header1.h`:

```
extern foo();
```

In `file11.c`:

```
#include <stdio.h>
#include "header1.h"
extern void foo();
int main(){
    printf("main();\n");
    foo();
}
```

```
gcc file11.c file2.c
```

INCLUDE HEADER FILE (2)

Header file in another directory

```
mv header1.h ..
#will this work?
gcc file11.c file2.c
gcc -I .. file11.c file2.c
```

- Gcc flag: `-I dir`

GCC FLAGS (SUMMARY)

- `-c` for compile, `-o` for output
- `-Ldir -lmylib` for linking a library
  - search library for unsolved symbols (functions, global variables) when linking
- `-I` for `#include`
  - header file (storing declarations)
- `-Wall`, w for warning
- `-g` for debug (later): `gcc -g file1.c file2.c`
- ref [link]

EXERCISE

- Write two C files:
  - `filea.c` defines functions `main()` and `bar()`
  - `fileb.c` defines function `foo()`
  - function `main()` calls `foo()`
  - function `foo()` calls `bar()`
  - Compile your program.
  - Submit the program and commands to BB.