

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

```
#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	<code>extern int x;</code>	<code>void foo();</code>
define	<code>int x;</code>	<code>void foo(){ }</code>
initialize	<code>int x=6;</code>	
reference	<code>y=x;x=1;</code>	<code>foo();</code> (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 & MAKE

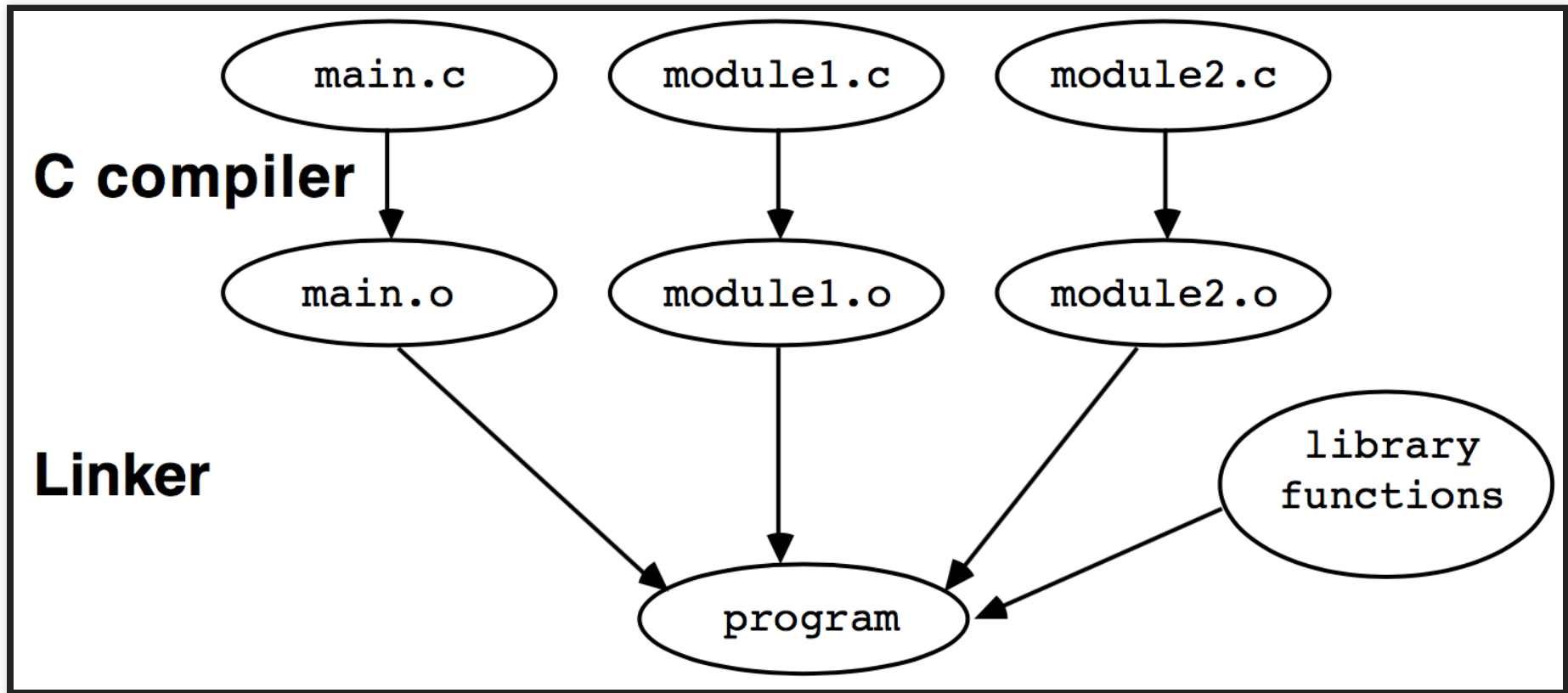
COMPILATION (1)

- 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



Linker

GCC: FLAGS

- `-c` for compile, `-o` for output
- `-Wall, -W` for warning
- `-I` for `#include`
 - header file (storing declarations)
- `-Ldir/-lmylib` for library to link
 - search library for unsolved symbols (functions, global variables) when linking
- `-g` for debug (later)
- ref [[link](#)]