

# ECE 220 Computer Systems & Programming

Lecture 7 – Introduction to C

February 5, 2019



# C – Higher Level Language

(2018 top programming languages ranked by [IEEE Spectrum](#))

## Gives symbolic names to values

- don't need to know which register or memory location

## Provides abstraction of underlying hardware

- operations do not depend on instruction set
- example: can write “ $a = b * c$ ”, even though LC-3 doesn't have a multiply instruction

## Provides expressiveness

- use meaningful symbols that convey meaning
- simple expressions for common control patterns (if-then-else)

## Enhances code readability

## Safeguards against bugs

- can enforce rules or conditions at compile-time or run-time

# Basic C Program

```
/*  
 * My first program in C. It will print the value of PI  
 * and then exit.  
 */  
#include <stdio.h>  
#define PI 3.1416f  
int main() {  
    float pi = PI;  
    printf("pi=%f\n", pi);  
    return 0;  
}
```

- a. Comment
- b. Preprocessor directives
- c. Main function
- d. Variable declaration (type, identifier, scope)
- e. I/O
- f. Return value
- g. Statement termination

\*[Github more Examples on Variables] 3

# Characteristics of C

C is a **procedural language**

- the program specifies an explicit sequence of steps to follow to produce a result; program is composed of functions (aka subroutines)

C programs are **compiled** rather than interpreted

- a compiler translates a C program into machine code that is directly executable on hardware
- interpreted programs (e.g. MATLAB) are executed by another program, called interpreter

C programs are **statically typed**

- the type of each expression is checked at compile time for type inconsistencies (e.g., `int x = 3.411`)
- [github: Constant variable example]

# Compiling a C Program

## Preprocessor

- macro substitution
- conditional compilation
- “source-level” transformations
  - output is still C

## Compiler

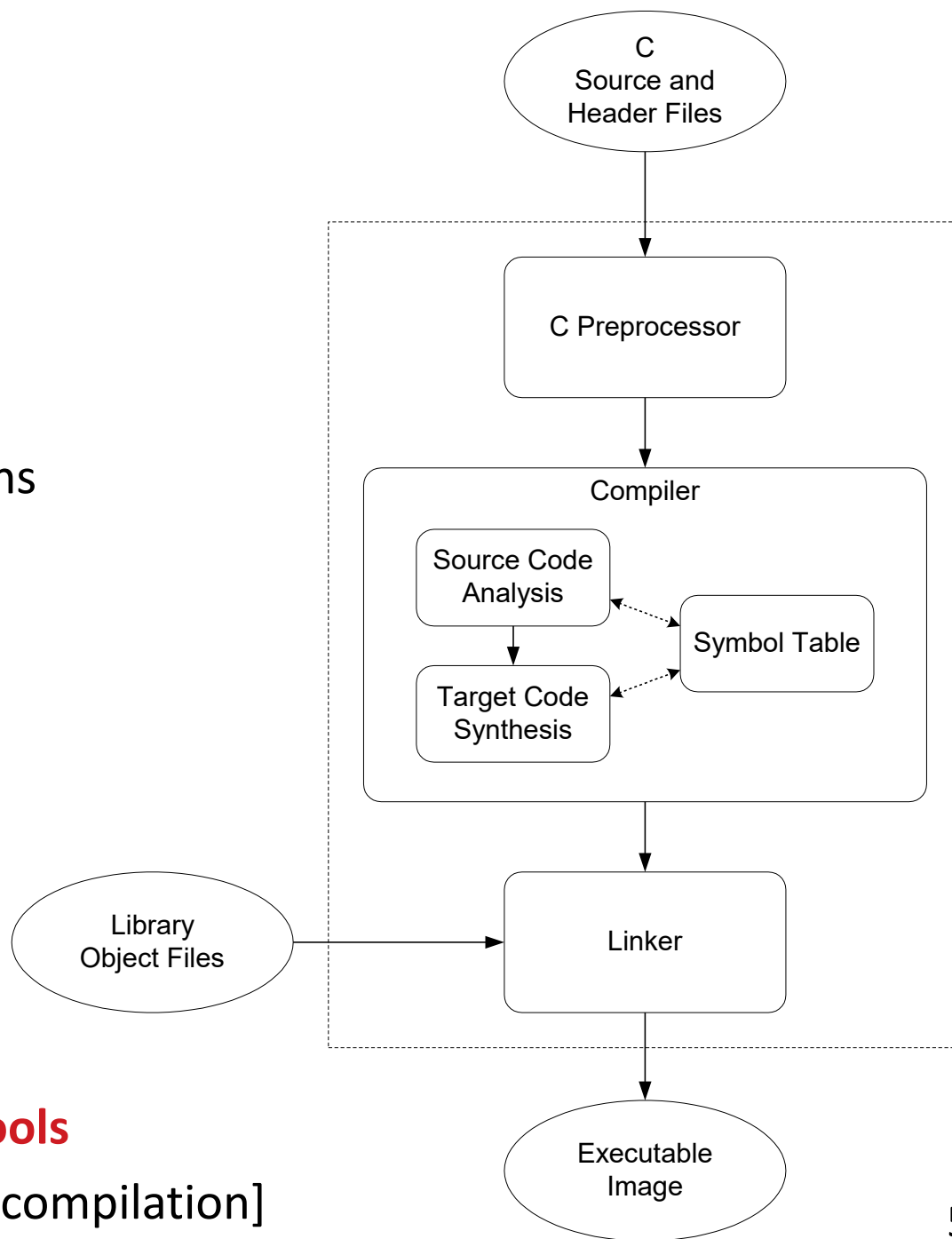
- generates object file
  - machine instructions

## Linker

- combine object files (including libraries) into executable image

✓ **gcc compiler – invoke all these tools**

[Github:see the code on conditional compilation]



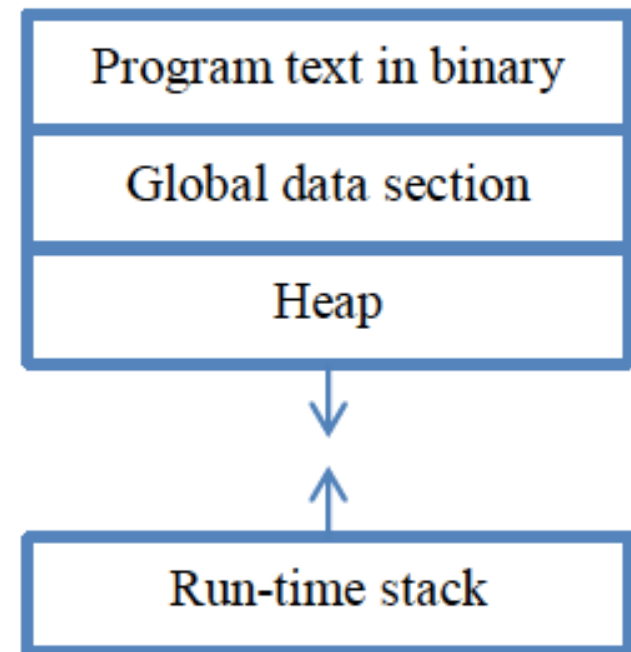
# Variables in C

- **int** (long, long long, unsigned), can also use hex representation 0xD
- **float** (double)
- **char** (character)
- **const** (constant qualifier)

**Scope:** local vs. global

**Storage class:** static vs. automatic

[Github: Example Codes on local vs. global and static]



# Operators

- Expression vs. Statement
- '=' vs. '=='
- The Assignment Operator (=):

- Arithmetic Operators:

- Order of evaluation:

precedence --  $x = 2 + 3 * 4$

associativity --  $x = 2 + 3 - 4 + 5$

parentheses --  $x = a * (b + c) * d / 2$

- Logical Operators: \_\_\_\_\_
- Bitwise Operators: \_\_\_\_\_
- Relational Operators: \_\_\_\_\_

## Operators (continued)

- Increment/Decrement Operators: ++, -- (post vs. pre)  
example: x = 4; y = ++x; vs. x = 4; y = x++;
- Special operator (conditional):  
variable = condition ? value\_if\_true : value\_if\_false;  
example: x = (y < z) ? 5 : 7
- Compound Assignment Operators:  
a += b; <--> a = a + b;

Expression with multiple operators (Table 12.5 of textbook)



# Basic I/O

**#include <stdio.h>**

**/\* header file for Standard Input Output \*/**

- **printf examples**

```
printf("%d is a prime number", 43);  
printf("43 + 59 in decimal is %d\n", 43+59);  
printf("a+b=%f\n", a+b);  
printf("%d+%d=%d\n", a, b, a+b);
```

- **scanf examples**

```
scanf("%c", &nextchar);  
scanf("%f", &radius);  
scanf("%d %d", &length, &height);
```

Formatting option: %d, %x, %c, %s, %f, \n,

Use “**man**” to look up library functions

# C Programming Exercise 1

```
int main() {  
    /* declare integer variables x, y and z */  
  
    /* set x to 5, set y to 3 */  
  
    /* increment x by 4 */  
  
    /* left shift x by y and then store the result to z */  
  
    /* print x, y, and z */  
  
    return 0;  
}
```

## C Programming Exercise 2

```
/*  
 * Write a C program to calculate the circumference of a circle when  
 * a user inputs the radius.  
 */  
  
/* preprocessor directives */  
  
int main() {  
    /* declare floating point variables (radius, circumference) */  
  
    /* prompt user to enter a floating point value for radius */
```

```
/* call scanf to get user input */  
  
/* calculate the circumference */  
  
/* print the result */  
  
/* return out */  
  
}
```