# ECE 220 Computer Systems & Programming

## Lecture 7 – Introduction to C

## September 19, 2017

- Midterm 1 is scheduled on Thursday, 9/28, 7pm to 8:30pm
- Conflict sign-up due this Thursday at 10pm

ECE ILLINOIS

ILLINOIS

# C – Higher Level Language

(2017 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 exits. */
#include <stdio.h>
#define PI 3.1416f
int main()
{
    float pi = PI;
    printf("pi=%f\n", pi);
    return 0;
}
```

- **Comment**
- **Preprocessor directives**
- **Main function**
- **Variable declaration (type, identifier, scope)**
- **I/O**
- **Return value**
- **Statement termination**

# 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 that 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`)

# Compiling a C Program

**Preprocessor**

- macro substitution
- conditional compilation
- "source-level" transformations
  - ➢ output is still C

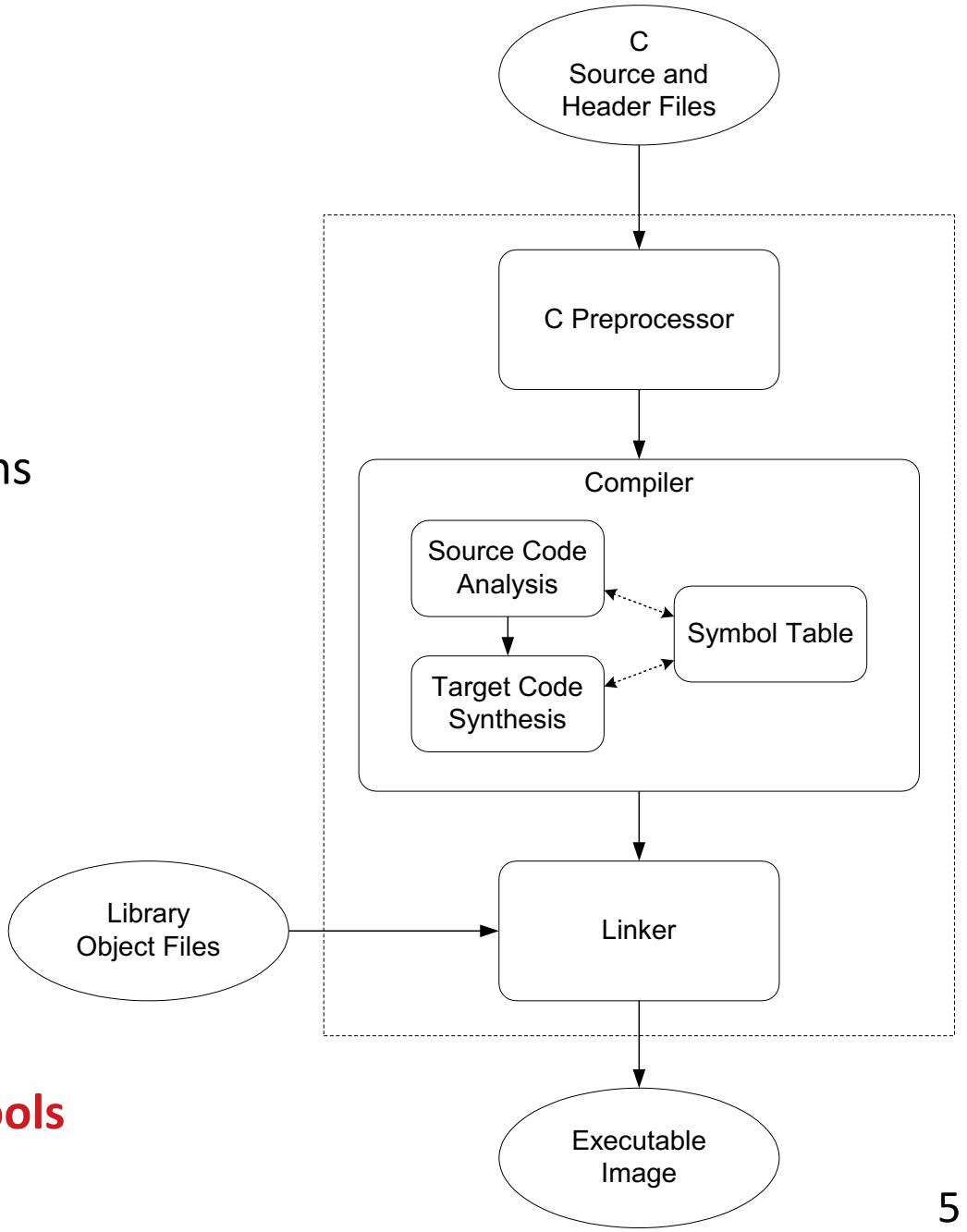
**Compiler**

- generates object file
  - ➢ machine instructions

**Linker**

- combine object files (including libraries) into executable image

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

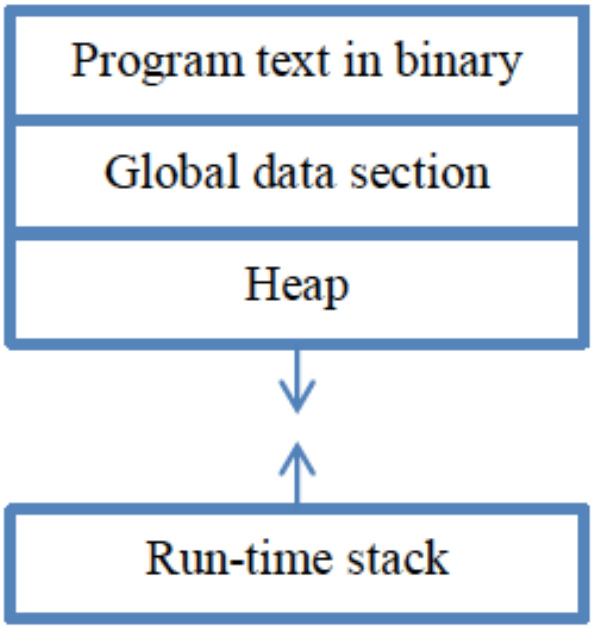


# Variables in C

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

**Scope**: local v.s. global

**Storage class**: static v.s. automatic



# Operators

- Expression  v.s. Statement
- ‘=‘ v.s. ‘==‘
- 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/pre)

  example: `x = 4; y = ++x; z = 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>**

- **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

```
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;
}
```