ECE 220 Computer Systems & Programming

Introduction to C

ECE ILLINOIS

ILLINOIS

C – Higher Level Language

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 = 3.141`)

Compiling a C Program

Preprocessor

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

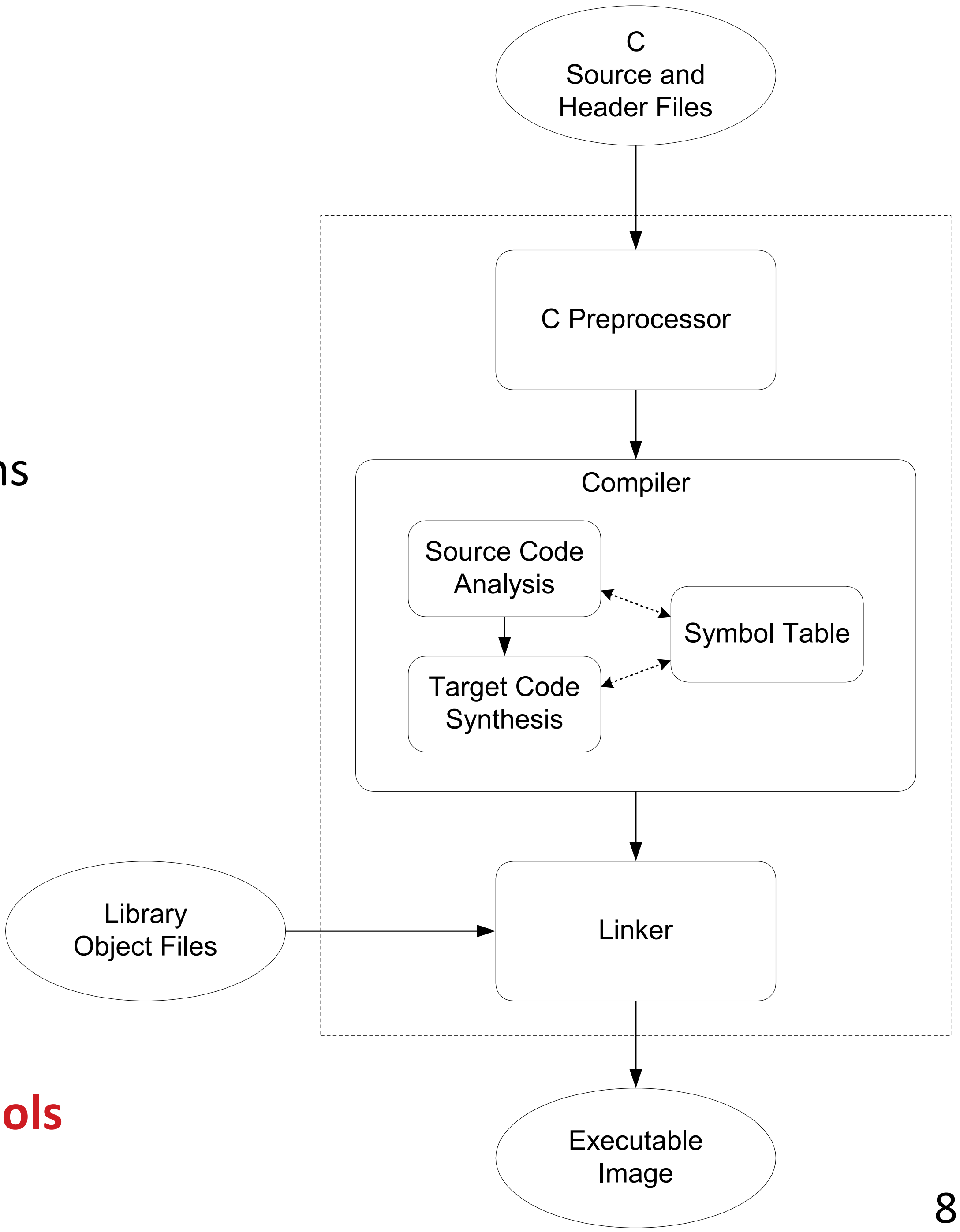
Compiler

- generates object file
  - ➢ machine instructions

Linker

- combine object files (including libraries) into executable image

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

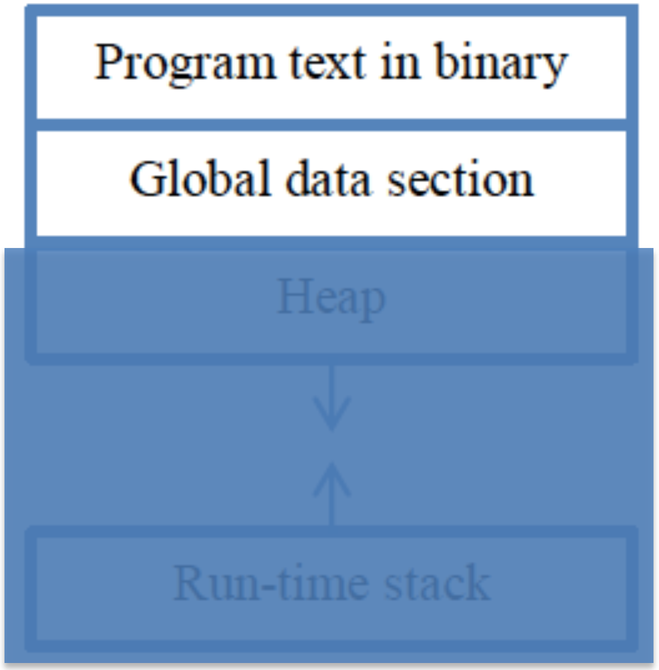


Variables in C

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

Scope: local vs. global

Storage class: static vs. automatic



Expressions and Statements

- Expressions evaluate to something. Examples:
  - 1
  - x
  - x*y
- Statements are building blocks of a C program. Examples:
  - x = 3;

Operators (1/8): Assignment

=

Lvalues

Evaluates to

Operators (2/8): Relational

```
>  <  >=  <=  ==  !=
```

Warning: `=` `!=` `==`

Operators (3/8): Arithmetic

+ – * / %

Operators (4/8): Bitwise

| &   ^   ~   <<   >>

Operators (5/8): Logical

| | & &   !

Operators (6/8): Increment/Decrement

++ −−

Operators (7/8): Ternary / Conditional

`?:`

Usage:

```
expr ? ift : iff
```

Operators (8/8): Compound Assignment

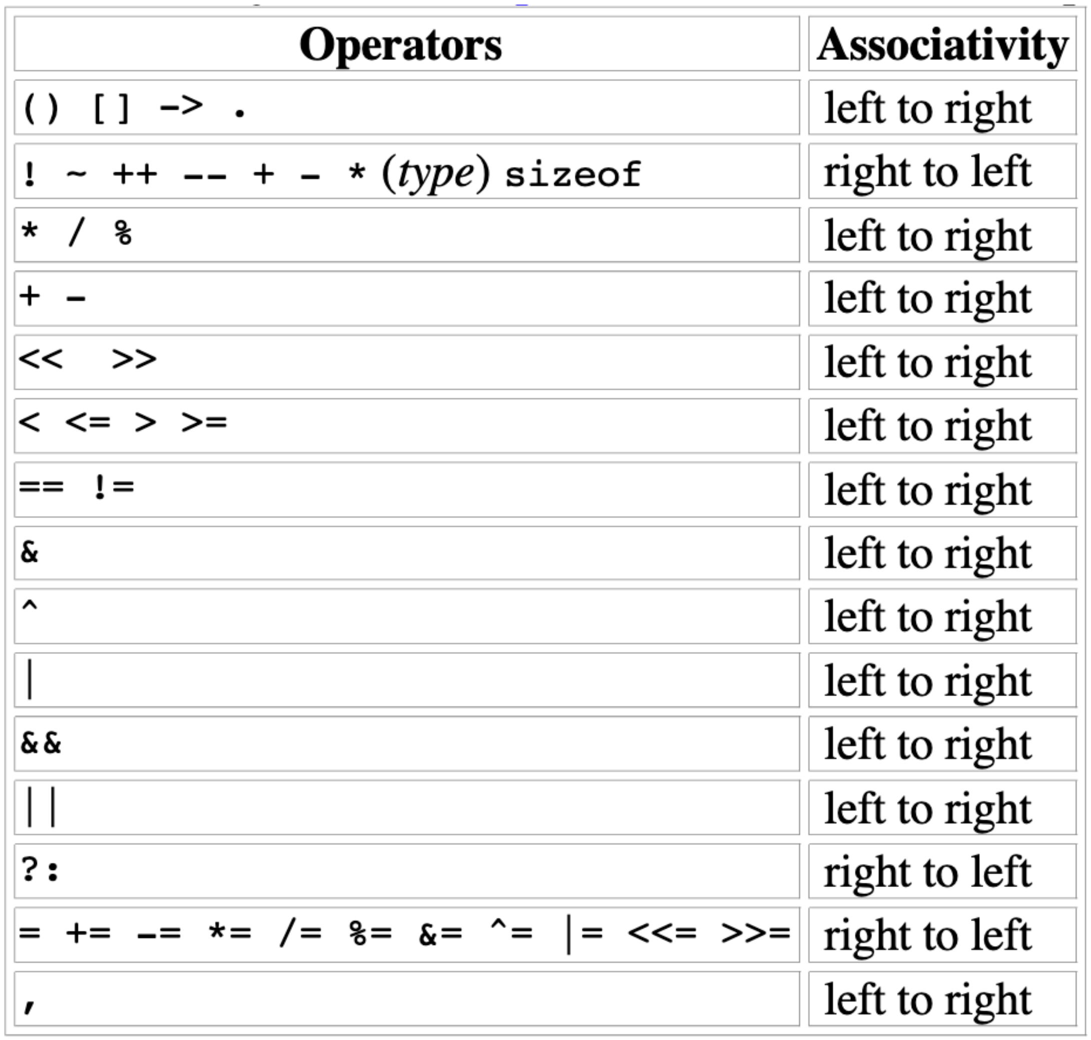
```
+=  -=  *=  /=  %=

&=  |=  ^=  <<=  >>=
```

Usage:

```
lval += expr
```

Precedence and Associativity (see Table 12.5)



Operators	Associativity
`() [] -> .`	left to right
`! ~ ++ -- + - *` (*type*) `sizeof`	right to left
`* / %`	left to right
`+ -`	left to right
`<< >>`	left to right
`< <= > >=`	left to right
`== !=`	left to right
`&`	left to right
`^`	left to right
`|`	left to right
`&&`	left to right
`||`	left to right
`?:`	right to left
`= += -= *= /= %= &= ^= |= <<= >>=`	right to left
`,`	left to right