

Programming in C



Your First Program

Hello World!

Soon I will control the world!



Program Basics

- The **source code** for a program is the set of instructions written in a high-level, human readable language.

```
X = 0;
```

```
MOVE 0 TO X.
```

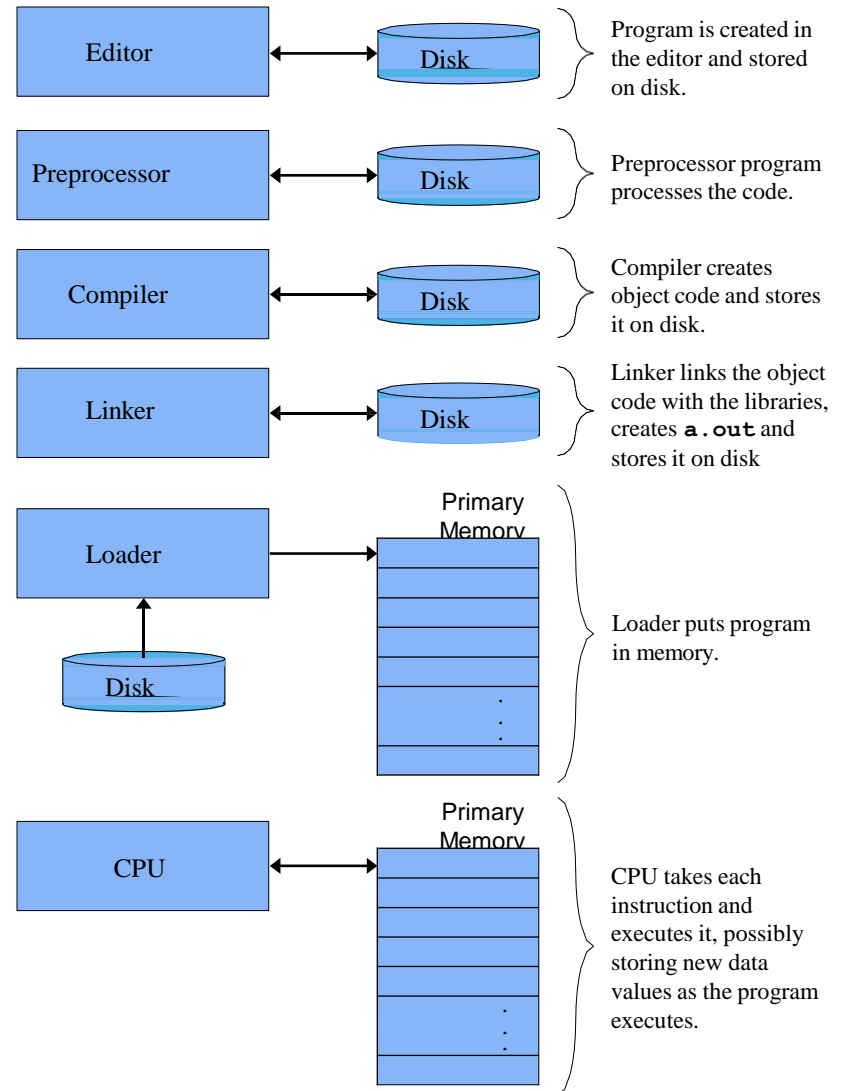
```
X := 0
```

- The source code is transformed into **object code** by a **compiler**. Object code is a machine usable format.
- The computer **executes** a program in response to a command.

Basics of a Typical C Environment

Phases of C Programs:

1. Edit
2. Preprocess
3. Compile
4. Link
5. Load
6. Execute



GCC Program Basics

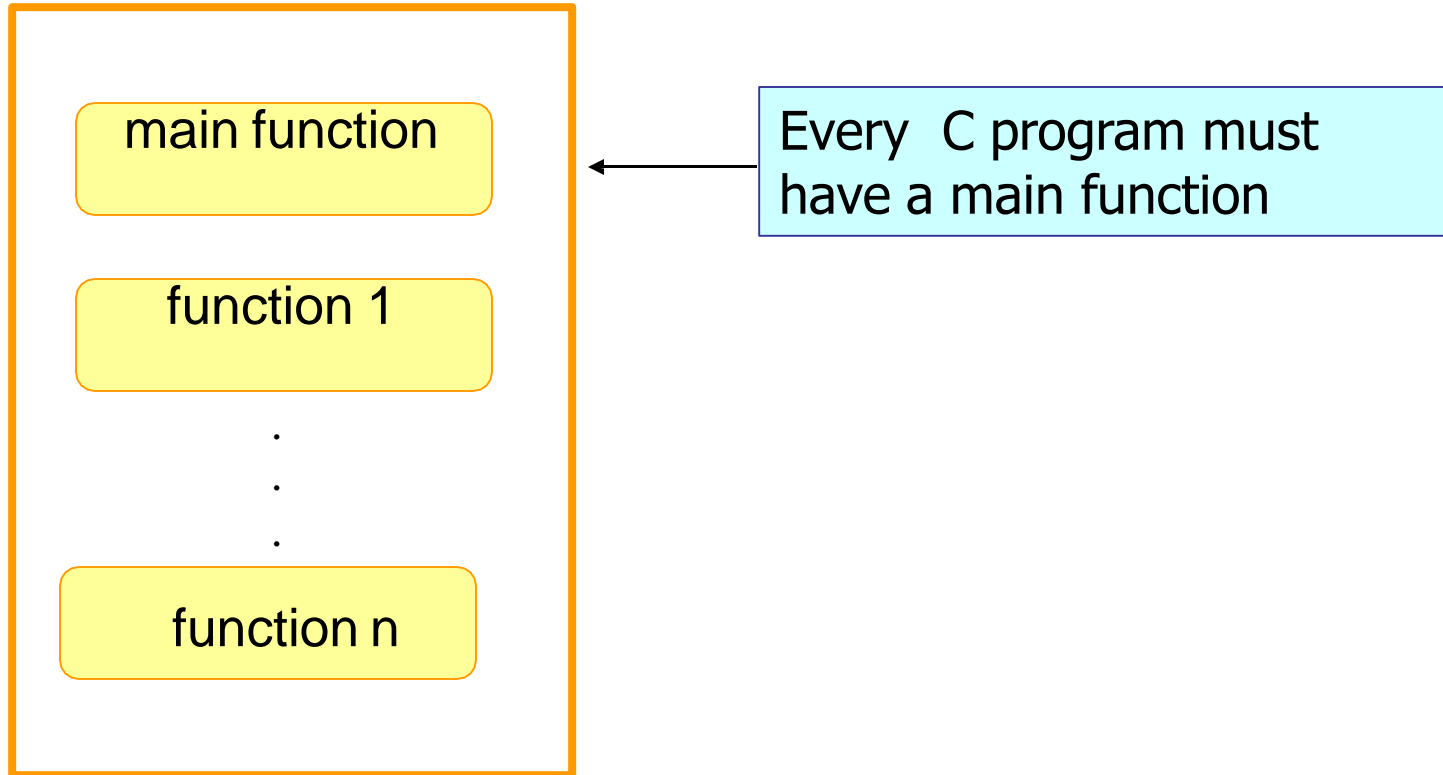
- The basic program writing sequence:
 1. create or modify a file of instructions using an editor
 - Unix: Pico, vi, gEdit, emacs, ...
 2. compile the instructions with GCC
 3. execute or run the compiled program
 - repeat the sequence if there are mistakes



Pico:

<http://www.bgsu.edu/departments/compsci/docs/pico.html>

Structure of a C Program



Functions

- Each function consists of a **header** followed by a **basic block**.
- General format:

<return-type> fn-name (parameter-list)
basic block

← *header*

The Basic Block

```
{  
    declaration of variables  
    executable statements  
}
```

- A semi-colon (;) is used to terminate a statement
- A block consists of zero or more statements
- Nesting of blocks is legal and common
 - Each interior block may include variable declarations

Return statement

- *return expression*
 1. Sets the return value to the value of the expression
 2. Returns to the caller / invoker
- Example:

```
int main()           // header
{                   // beginning of basic block
    // ...
    return 0;       // program ending successfully
}                   // end of basic block
```

SSH Secure Shell

- On-Campus / VPN

- SSH to one of the machines in the list
- machine.cs.clemson.edu

- Off-Campus

- SSH to access.cs.clemson.edu
- ssh machine.cs.clemson.edu

```
access.cs.clemson.edu - psterli Term Off Campus - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles

This copy of SSH Secure Shell is a non-commercial version.
This version does not include PKI and PKCS #11 functionality.

Last login: Mon Jul 12 17:35:28 2010 from 71-12-9-167.dhc
*****
                        Welcome to the
                CLEMSON UNIVERSITY SCHOOL OF COMPUTING
                Unauthorized use is prohibited!

                PARTIAL LIST OF PUBLIC CLIENT MACHINES
Hostnames                OS/Arch
-----
dragon1 - dragon24       CentOS 5/x86_64
frog1 - frog27           CentOS 5/x86_64
gecko1 - gecko22        CentOS 5/x86_64
sparc1 - sparc2         Solaris 10/SPARC

* Questions or problems regarding Unix systems should be addressed to
  "ithelp@clemson.edu" or the friendly folks in 109 or 137 McAdams

***** Last updated: 2010-07-06 *****
[17:36:47] userid.@sparc1:~ [99]
Connected to access.cs.clemson.edu  SSH2 - aes128-cbc - hmac-md5 - nc 80x24
```

Unix Commands: mkdir & cd

`mkdir cpsc1110`

- Creates a new directory / folder

`cd cpsc1110`

- Changes the current directory

`pico ch02First.c`

- Runs the pico editor to edit file ch02First.c

Our First Program

```
// Program:      ch03First
// Purpose:      A first program in C
//               Printing a line of text
// Author:       Ima Programmer
// Date:        mm/dd/yy

#include <stdio.h>

int main() {
    printf("Go Tigers!!!\n");
    return 0; //indicates program ended successfully
}
```

Go Tigers!!!

Compiling and Running a Program

- To compile and print all warning messages, type

```
gcc -Wall prog-name.c
```

- If using math library (math.h), type

```
gcc -Wall prog-name.c -lm
```



- By default, the compiler produces the file *a.out*

Compiling and Running a Program

- To execute the program type
`./a.out`
 - The ./ indicates the current directory
- To specify the file for the object code, for example, `p1.o`, type
`gcc -Wall prog1.c -o p1.o`
then type
`./p1.o`
to execute the program

Comments

- Make programs easy to read and modify
- Ignored by the C compiler
- Two methods:
 1. `//` - line comment
 - everything on the line following `//` is ignored

```
//Purpose:   Display Go Tigers!
```

2. `/* */` - block comment
 - everything between `/* */` is ignored

```
/*  
Program:   ch02First  
Purpose:   Display Go Tigers!  
Author:    Ima Programmer  
Date:     mm/dd/yy  
*/
```

Preprocessor Directive: #include

- A C program line beginning with # that is processed by the compiler before translation begins.
- #include pulls another file into the source
 - `#include <stdio.h>` causes the contents of the named file, `stdio.h`, to be inserted where the # appears. File is commonly called a header file.
 - `<>`'s indicate that it is a compiler standard header file.
 - `#include "myfunctions.h"` causes the contents of `myfunctions.h` to be inserted
 - `"`'s indicate that it is a user file from current or specified directory

Introduction to Input/Output

- *Input data* is read into variables
- *Output data* is written from variables.
- Initially, we will assume that the user
 - enters data via the terminal keyboard
 - views output data in a terminal window on the screen

```
[16:35:02] psterli@frog6:~/cpssc111 [104] ./a.out
Enter two integers: 6 20
Enter a floating point number: 3.5
6 / 20 = 0
3.50 / 20 = 0.17
sqrt(3.500000) = 1.87
[16:35:35] psterli@frog6:~/cpssc111 [105] █
```

Program Input / Output

- The C run-time system automatically opens two files for you at the time your program starts:
 - **stdin** – standard input (from the keyboard)
 - **stdout** – standard output (to the terminal window in which the program started)
- Later, how to read and write files on disk
 1. Using `stdin` and `stdout`
 2. Using `FILE`'s

Console Input/Output

- Defined in the C library included in `<stdio.h>`
 - Must have this line near start of file:
`#include <stdio.h>`
 - Includes input functions `scanf`, `fscanf`, ...
 - Includes output functions `printf`, `fprintf`, ...

Console Output - printf

- Print to standard output, typically the screen
- General format (value-list may not be required):
`printf("format string", value-list);`

```
printf("Go Tigers!!!");
```

Console Output

What can be output?

- Any data can be output to display screen
 - Literal values
 - Variables
 - Constants
 - Expressions (which can include all of above)
- Note
 - Values are passed to printf
 - Addresses are passed to scanf

Console Output

- We can
 - Control vertical spacing with blank lines
 - Use the escape sequence “\n”, new line
 - Should use at the end of all lines unless you are building lines with multiple printf’s.
 - If you printf without a \n and the program crashes, you will not see the output.
 - Control horizontal spacing
 - Spaces
 - Use the escape sequence “\t”, tab
 - Sometimes undependable.

Terminal Output - Examples

```
printf("Hello World!\n");
```

- Sends string "Hello World" to display, skipping to next line

```
printf("Good morning\nMs Smith.\n");
```

- Displays the lines
 Good morning
 Ms Smith.

Program Output: Escape Character \

- Indicates that a “special” character is to be output

| Escape Sequence | Description |
|-----------------|--|
| <code>\n</code> | Newline. Position the screen cursor to the beginning of the next line. |
| <code>\t</code> | Horizontal tab. Move the screen cursor to the next tab stop. |
| <code>\r</code> | Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line. |
| <code>\a</code> | Alert. Sound the system bell. |
| <code>\\</code> | Backslash. Used to print a backslash character. |
| <code>\"</code> | Double quote. Used to print a double quote character. |

Template: a.c

- Starting point for a new program
 - Read into (^R in pico) or
 - Copy into (cp command) a new file
 - Ex: `cp a.c prog1.c`

```
/*
   Program:   ?
   Purpose:   ?
   Author:    Im A Programmer
   Date:      mm/dd/yy
*/

#include <stdio.h>

int main() {

    return 0;    // Return normally
}
```

Programming in C



Chapter 2 *Your First Program*

T H E E N D