Introducing C

Lecture 17

Instructor: C. Pu (Ph.D., Assistant Professor)

puc@marshall.edu

Origins of C

- C is a by-product of UNIX, developed at Bell Laboratories by Ken Thompson, Dennis Ritchie, and others.
- Thompson designed a small language named B.
- B was based on BCPL, a systems programming language developed in the mid-1960s.

Origins of C

- By 1971, Ritchie began to develop an extended version of B.
- He called his language NB (“New B”) at first.
- As the language began to diverge more from B, he changed its name to C.
- The language was stable enough by 1973 that UNIX could be rewritten in C.

Standardization of C

- *K&R C*
  - Described in Kernighan and Ritchie, *The C Programming Language* (1978)
  - De facto standard
- *C89/C90*
  - ANSI standard X3.159-1989 (completed in 1988; formally approved in December 1989)
  - International standard ISO/IEC 9899:1990
- *C99*
  - International standard ISO/IEC 9899:1999
  - Incorporates changes from Amendment 1 (1995)

C-Based Languages

- ***C++*** includes all the features of C, but adds classes and other features to support object-oriented programming.
- ***Java*** is based on C++ and therefore inherits many C features.
- ***C#*** is a more recent language derived from C++ and Java.
- ***Perl*** has adopted many of the features of C.
- Why C???

Properties of C

- Low-level
  - Access to machine-level concepts (bytes and address)
  - Operations that correspond closely to a computer's built-in instructions.
- Small
  - Limited set of features
  - Relies heavily on a "library" of standard functions
- Permissive
  - A wide degree of programming
  - Doesn't mandate the detailed error-checking

Strengths of C

- Efficiency
- Portability
- Power
- Flexibility
- Standard library
- Integration with UNIX

Weaknesses of C

- Programs can be error-prone.
- Programs can be difficult to understand.
- Programs can be difficult to modify.

```
v,i,j,k,l,s,a[99];
main()
{
  for(scanf("%d",&s);*a-s;v=a[j*=v]-a[i],k=i<s,j+=(v=j<s&&
(!k&&!!printf(2+"\n\n%c"-(!l<<!j)," #Q"[l^v?(l^j)&1:2])&&
++l||a[i]<s&&v&&v-i+j&&v+i-j))&&!(l%=s),v||(i==j?a[i+=k]=0:
++a[i])>=s*k&&++a[--i])
    ;
}
```

Effective Use of C

- Learn how to avoid pitfalls.
- Use software tools (debuggers) to make programs more reliable.
- Take advantage of existing code libraries.
- Adopt a sensible set of coding conventions.
- Avoid “tricks” and overly complex code.
- Stick to the standard.

Program: Printing Hello World

```
#include <stdio.h>

int main(void)
{
  printf(“Hello World!\n");

  return 0;
}
```

- This program might be stored in a file named `ch02_01.c`.
- The file name doesn’t matter, but the `.c` extension is often required.

Compiling and Linking

- Before a program can be executed, three steps are usually necessary:
  - ***Preprocessing.*** The ***preprocessor*** obeys commands that begin with # (known as ***directives***)
  - ***Compiling.*** A ***compiler*** translates then translates the program into machine instructions (***object code***).
  - ***Linking.*** A ***linker*** combines the object code produced by the compiler with any additional code needed to yield a complete executable program.
- The preprocessor is usually integrated with the compiler.

The GCC Compiler

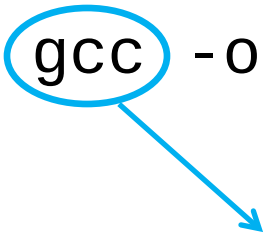
- GCC is one of the most popular C compilers.
- GCC is supplied with Linux but is available for many other platforms as well.
- Using GCC compiler:

```
$ gcc –o ch02_01 ch02_01.c
```

The GCC Compiler

- GCC is one of the most popular C compilers.
- GCC is supplied with Linux but is available for many other platforms as well.
- Using GCC compiler:

```
$ gcc -o ch02_01 ch02_01.c
```

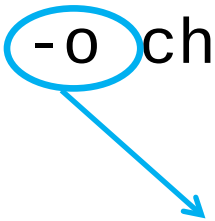


Command to compile and link

The GCC Compiler

- GCC is one of the most popular C compilers.
- GCC is supplied with Linux but is available for many other platforms as well.
- Using GCC compiler:

```
$ gcc -o ch02_01 ch02_01.c
```

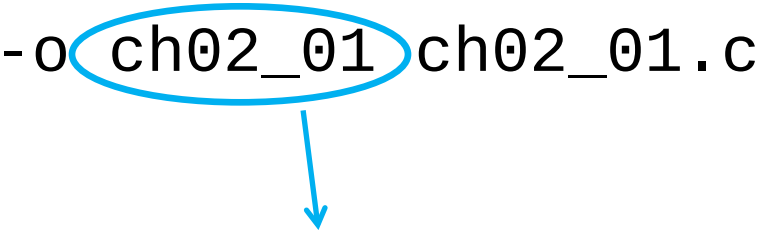


Command option: specify the name of the executable program

The GCC Compiler

- GCC is one of the most popular C compilers.
- GCC is supplied with Linux but is available for many other platforms as well.
- Using GCC compiler:

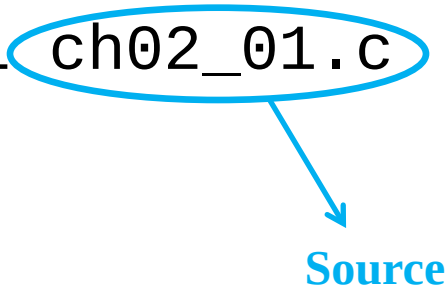
```
$ gcc -o ch02_01 ch02_01.c
```



The GCC Compiler

- GCC is one of the most popular C compilers.
- GCC is supplied with Linux but is available for many other platforms as well.
- Using GCC compiler:

```
$ gcc -o ch02_01 ch02_01.c
```



The GCC Compiler

- GCC is one of the most popular C compilers.
- GCC is supplied with Linux but is available for many other platforms as well.
- Using GCC compiler:

```
$ gcc -o ch02_01 ch02_01.c
```

- http://www.tutorialspoint.com/compile_c_online.php

Integrated Development Environments

- An ***integrated development environment (IDE)*** is a software package that makes it possible to edit, compile, link, execute, and debug a program without leaving the environment.
- Visual Studio 2015

The **General Form** of a Simple Program

- Simple C programs have the form

directives

```
int main(void)
{
   statements
}
```

```
1   #include <stdio.h>
2
3   int main(void)
4   {
5
6       printf("Hello World!\n");
7
8       return 0;
9   }
10
```

The General Form of a Simple Program

- Even the simplest C programs rely on three key language features:
  - Directives
  - Functions
  - Statements

```
directives

int main(void)
{
   statements
}
```

Directives

- Before a C program is compiled, it is first edited by a **preprocessor**.
- Commands intended for the preprocessor are called **directives**.
- Example:

```
#include <stdio.h>
```

```
Standard Input and Output Library
```

- `<stdio.h>` is a ***header*** containing information about C's standard I/O library, included in our program.
  - printf

Directives

- Directives always begin with a # character.
- By default, directives are one line long; there's no semicolon or other special marker at the end.
- Format:

  #include <filename>

- Example:

  #include <math.h>

  http://www.tutorialspoint.com/c_standard_library/math_h.htm

Functions

- A ***function*** is a series of statements that have been grouped together and given a name.
- ***Library functions*** are provided as part of the C implementation.
- A function that computes a value uses a **`return`** statement to specify what value it "returns":

```
return x + 1;
```

The `main` Function

- The `main` function is **mandatory**.
- `main` is special: it gets called automatically when the program is executed.
- `main` returns a status code; the value 0 indicates normal program termination.
- If there's no `return` statement at the end of the `main` function, many compilers will produce a warning message.

Statements

- A ***statement*** is a command to be executed when the program runs.
- `ch02_01.c` uses only two kinds of statements. One is the **`return` statement**; the other is the ***function call.***
- Asking a function to perform its assigned task is known as ***calling*** the function.
- `Ch02_01.c` calls `printf` to display a string:

```
printf(“Hello World.\n");
```

Statements

- C requires that each statement end with a semicolon.

```
3
4   #include <stdio.h>
5
6   int main(void) {
7
8       printf("Hello World!\n");
9
10      return 0;
11  }
12
13
```

  - There's one exception: the compound statement.

Printing Strings

- When the `printf` function displays a ***string literal***—characters enclosed in **double quotation marks**—it doesn't show the quotation marks.
- `printf` doesn't automatically advance to the next output line when it finishes printing.
- To make `printf` advance one line, include **`\n`** (the ***new-line character***) in the string to be printed.

Comments

- A ***comment*** begins with **/*** and end with ***/**.

```
/* This is a comment */
```

- Comments may appear almost anywhere in a program, either on separate lines or on the same lines as other program text.
- Comments may extend over more than one line.

```
/* Name: ch2_04.c
   Purpose: Prints Hello World
   Date:07-12-2017
   Author: Cong Pu */
```

Comments

```
printf(“My ");
printf(“name is ");
printf(“James ");
printf(“Bond.\n");
```

Comments

```
printf("My ");        /* forgot to close this comment
printf(“name is ");
printf(“James ");     /* so it ends here */
printf(“Bond");
```

- ***Warning***: Forgetting to terminate a comment may cause the compiler to ignore part of your program:

Comments

- ***Warning***: Forgetting to terminate a comment may cause the compiler to ignore part of your program:

```
printf("My ");  /* forgot to close this comment
printf(“name is ");
printf(“James ");   /* so it ends here */
printf(“Bond");
```

Comments

- ***Warning***: Forgetting to terminate a comment may cause the compiler to ignore part of your program:

```
printf("My ");  /* forgot to close this comment
printf(“name is ");
printf(“James ");   /* so it ends here */
printf(“Bond");
```

Comments

- Comments can also be written in the following way:

```
// This is a comment
```

- This style of comment ends automatically at the end of a line.

Variables and Assignment

- Most programs need to a way to store data temporarily during program execution.
- These storage locations are called ***variables.***

Types

- Every variable must have a ***type.***
- https://www.geeksforgeeks.org/data-types-in-c/
- C has a wide variety of types, including **`int`** and **`float`**.
- A variable of type `int` (short for *integer*) can store a whole number such as 0, 1, 392, or –2553.
  - 2 Bytes, -32,768 to 32,767

  OR
  - 4 Bytes, -2,147,483,648 to 2,147,483,647

Declarations

- Variables must be ***declared*** before they are used.
- Variables can be declared one at a time:

```
int height;
float profit;
```

- Alternatively, several can be declared at the same time:

```
int height, length, width, volume;
float profit, loss;
```

Declarations

- **In C89**, when `main` contains declarations, these must **precede statements**:

```
int main(void)
{
   declarations
   statements
}
```

- **In C99**, declarations don't have to come before statements.

Assignment

- A variable can be given a value by means of ***assignment:***

  ```
  height = 8;
  ```

  The number **8** is said to be a ***constant*****.**
- Before a variable can be assigned a value—or used in any other way—it must first be declared.

Assignment

- A constant assigned to a `float` variable usually contains a decimal point:

  ```
  profit = 2150.48;
  ```

- It's best to append the letter `f` to a floating-point constant if it is assigned to a `float` variable:

  ```
  profit = 2150.48f;
  ```

  Failing to include the `f` may cause a warning from the compiler.

Assignment

- An `int` variable is normally assigned a value of type `int`, and a `float` variable is normally assigned a value of type `float`.
- Mixing types (such as assigning an `int` value to a `float` variable or assigning a `float` value to an `int` variable) is possible but not always safe.

Assignment

- Once a variable has been assigned a value, it can be used to help compute the value of another variable:

```
height = 8;
length = 12;
width = 10;

volume = height * length * width;
  /* volume is now 960 */
```