

CSC 4304 - Systems Programming
Fall 2010

LECTURE - II
BASICS OF C PROGRAMMING

Tevfik Koşar

Louisiana State University
August 26th, 2010

Summary of Last Class

- Basics of UNIX:
 - logging in , changing password
 - text editing with vi, emacs and pico
 - file and directory operations
 - file/dir permissions and changing them
 - process monitoring and manipulation

C vs. Java

- C is procedural, not object oriented
 - C has no objects, interfaces or packages
 - A program only consists of functions and data
- C is compiled, not interpreted
 - Translated directly to assembly language
 - Faster, less portable and very hard to debug.
- C has no array bounds, null pointer or cast checks
 - You have to detect and handle all problems yourself
- C has no garbage collector
 - You have to do all of the memory management yourself

3

C vs. Java (cont.)

- C has pointers
 - Similar to Java references but...
 - ...they can be used in calculations (pointer arithmetic)
 - Allows you to use the location of data in computations (not just the value)
 - Useful, powerful and a debugging nightmare!
- Compared to Java, C is a low-level language
 - You can and must do everything yourself
 - Suitable for low-level software like device-drivers, communication libraries, operating systems, etc.
- You can implement anything in C!
 - No limits!

4

C vs. Java (cont.)

- A Java program consists of:
 - Several classes, one class per file.
 - A main method in one of these classes.
 - External class libraries (jar files).
- A C program consists of:
 - Several functions in any number of files.
 - A main function in one of these files.
 - Possibly some header files.
 - External libraries with their own header files.

5

C can be quite complex

- This program computes and prints the first 800 decimals of PI:

```
---
#include <stdio.h>
long a=10000,b,c=2800,d,e,f[2801],g;
int main(){
    for(;b-c;)f[++b]=a/5;
    for(;d=0,g=c*2;c-=14,printf("%.4d",e+d/a),e=d%a)
        for(b=c;d+=f[b]*a,f[b]=d%--g,d/=g--,--b;d*=b);
}
---
```

6

Basic C Program

```
main()
{
}

```

7

Basic C Program: Print to stdout

```
#include <stdio.h>

main()
{
    printf("Hello, CSC4304 Class!\n");
}

```

8

Basic C Program: Print to stdout

```
#include <stdio.h>

main()
{
    printf("Hello, CSC4304 Class!\n");
}

---

gcc prog1.c          ==> a.out
gcc prog1.c -o prog1 ==> prog1
make prog1           ==> prog1
```

9

Header Files

- The C compiler works in 3 phases:
 - ① Pre-process source files
 - ② Compile source files into object files
 - ③ Link object files into an executable
- `#include <stdio.h>` means "include the contents of standard file `stdio.h` here"
 - ① Standard files are usually located in directory `/usr/include`
 - ② `/usr/include/stdio.h` may contain `#include` statements itself...
- You can use `#include` to include your own files into each other:
 - ▶ `#include "myfile.h"` means: "include file `myfile.h` (from the current directory) here"
 - ▶ Included files usually have extension `".h"` (header)

10

Read argument and print

```
#include <stdio.h>

// take arguments from stdin
main(int argc, char* argv[])
{
    printf("Hello, %s!\n", argv[1]);
}
```

11

Read argument and print

```
#include <stdio.h>

main(int argc, char* argv[])
{
    if (argc < 2){
        printf("Usage: %s <your name>\n", argv[0]);
    }
    else{
        printf("Hello, %s!\n", argv[1]);
    }
}
```

12

Read from stdin and print

```
#include <stdio.h>

main()
{
    char name[64];
    printf("What's your name?");
    scanf("%s", name);
    printf("Hello, %s!\n", name);
}
```

13

Basic Data Types

- Basic Types
 - char : character - 1 byte
 - short: short integer - 2 bytes
 - int: integer - 4 bytes
 - long: long integer - 4 bytes
 - float: floating point - 4 bytes
 - double - double precision floating point - 8 bytes
- Formatting Template
 - %d: integers
 - %f: floating point
 - %c: characters
 - %s: string
 - %x: hexadecimal
 - %u: unsigned int

14

Test Size of Data Types

```
#include <stdio.h>

main()
{
    printf("sizeof(char): %d\n", sizeof(char));
    printf("sizeof(short): %d\n", sizeof(short));
    printf("sizeof(int): %d\n", sizeof(int));
    printf("sizeof(long): %d\n", sizeof(long));
    printf("sizeof(float): %d\n", sizeof(float));
    printf("sizeof(double): %d\n", sizeof(double));
}
```

15

Formatting

```
#include <stdio.h>

main()
{
    char var1;
    float f;

    printf(" Enter a character:");
    scanf("%c", &var1);
    printf("You have entered character:%c \n ASCII value=%d \n\n"
           "Address=%x\n", var1, var1, &var1);

    printf(" And its float value would be: %.2f\n", (float)var1);
}
```

16

Formatting (cont.)

<pre>#include <stdio.h> int main(void) { int val = 5; char c = 'a'; char str[] = "world"; printf("Hello world\n"); printf("Hello %d World\n", val); printf("%d %c World\n", val, c); printf("Hello %s\n", str); printf("Hello %d\n", str); return 0; }</pre>	<pre> Hello world Hello 5 World 5 a World Hello world ** wrong! ** </pre>
--	---

17

Arrays

- Defining an array is easy:

```
int a[3];    /* a is an array of 3 integers */
```

- Array indexes go from 0 to n-1:

```
a[0] = 2; a[1] = 4; a[2] = a[0] + a[1];
int x = a[a[0]];    /* what is the value of x? */
```

- ▶ **Beware:** in this example a[3] does not exist, but your compiler will not complain if you use it!

- ★ But your program may have a very strange behavior...

- You can create multidimensional arrays:

```
int matrix[3][2];
matrix[0][1] = 42;
```

18

Strings

- A string is an array of characters:

```
char hello[15]="Hello, world!\n";
```

- Unlike in Java, you must decide in advance how many characters can be stored in a string.
 - ▶ You cannot change the size of the array afterwards
- Beware: strings are always terminated by a NULL character: '\0'
 - ▶ For example, "Hello" is string of 6 characters:

H	e	l	l	o	\0
---	---	---	---	---	----

19

Manipulating Arrays

- You cannot copy an array into another directly
 - ▶ You must copy each element one at a time

```
int a[3] = {12,24,36};
int b[3];

b = a;    /* This will NOT work! */

b[0]=a[0];
b[1]=a[1];
b[2]=a[2]; /* This will work */
```

20

Manipulating Strings

- There are standard function to manipulate strings:
 - ▶ `strcpy(destination, source)` will copy string **source** into string **destination**:

```
char a[15] = "Hello, world!\n";  
char b[15];  
strcpy(b,a);
```

- ⚠ Attention: `strcpy` does **not** check that **destination** is large enough to accomodate **source**.

```
char c[10];  
strcpy(c,a); /* This will get you in BIG trouble */
```

21

Manipulating Strings (cont.)

- Instead of `strcpy` it is **always better to use** `strncpy`:
 - ▶ `strncpy` takes one more parameter to indicate the maximum number of characters to copy:

```
char a[15] = "Hello, world!";  
char c[10];  
strncpy(c,a,9); /* Why 9 instead of 10? */
```

22

Comparison Operators

- The following operators are defined for basic data types:

```
if (a == b) { ... }
if (a != b) { ... }
if (a < b) { ... }
if (a <= b) { ... }
if (a > b) { ... }
if (a >= b) { ... }
if ((a==b) && (c>d)) {...} /* logical AND */
if ((a==b) || (c>d)) {...} /* logical OR */
```

- There is no boolean type in C. We use integers instead:

- ▶ 0 means FALSE
- ▶ Any other value means TRUE

```
int x;
if (x) {...}          /* Equivalent to: if (x!=0) {...} */
if (!x) {...}         /* Equivalent to: if (x==0) {...} */
```

23

Example

```
#include <stdio.h>

main()
{
    int x = 5;
    int y = 3;

    if (x==y){
        printf("x is equal to y, x=%d, y=%d\n", x, y);
    }
    else{
        printf("x is not equal to y, x=%d, y=%d\n", x, y);
    }
}
```

24

Classical Bugs

- Do not confuse '=' and '=='!

```
if (x=y) { ... }    /* This is correct C but it means something different */  
if (x=3) { /* always executed */ }  
if (x=0) { /* never executed */ }
```

- Do not confuse '&' and '&&'!

```
if (x&y) { ... }    /* This is correct C but it means something different */  
if (x|y) { ... }
```

Exercise:

- (7 & 8) vs (7 && 8)
- (7 | 8) vs (7 || 8)

25

Loops

```
while (x>0){  
  ...  
}
```

```
do{  
  ...  
} while (x>0);
```

```
for (x=0; X<3;x++) {...}
```

26

Functions

- In C, functions can be defined in two ways:

```
int foo() {                                /* function foo returns an int */
    ...
    return 123;
}

void bar(int p1, double p2) {              /* function bar returns nothing */
    ...
}
```

- Calling a function is easy:

```
int i = foo(); /* call function foo() */
bar(2, -4.321); /* call function bar() */
```

27

Exercises

1. Write a program which defines an integer, a float, a character and a string, then displays their values and their sizes on screen. /*use the sizeof() function*/

2. Write a program which computes and displays fib(n), where n is a parameter taken from command line:

fib(0) = 0, fib(1) = 1

If $n > 1$ then $\text{fib}(n) = \text{fib}(n - 1) + \text{fib}(n - 2)$

28

Summary

- C Basics
 - C vs Java
 - Writing to stdout
 - Taking arguments
 - Reading from stdio
 - Basic data types
 - Formatting
 - Arrays and Strings
 - Comparison Operators
 - Loops
 - Functions



29

Acknowledgments

- Advanced Programming in the Unix Environment by R. Stevens
- The C Programming Language by B. Kernighan and D. Ritchie
- Understanding Unix/Linux Programming by B. Molay
- Lecture notes from B. Molay (Harvard), T. Kuo (UT-Austin), G. Pierre (Vrije), M. Matthews (SC), and B. Knicki (WPI).

30