COMSW 1003-1

Introduction to Computer Programming in C

Lecture 17

Spring 2011

Instructor: Michele Merler

http://www1.cs.columbia.edu/~mmerler/comsw1003-1.html

Review - Arrays of strings

- An array Arr of 3 strings of variable length

```
char *Arr[3]={ "Hello", "World", "Wonderful" };

Arr[2] = Arr+2 // "Wondeful"
```

- An array Arr of 3 strings of maximum length = 15

```
char Arr2[3][15] = { "Hello2", "World2", "Wonderful2" };

Arr2[0] = Arr2 // "Hello2"

Arr2[1] = Arr2+1 // "World2"
```

Program's Inputs

- When we run a program, sometimes we want to pass some input arguments to it
- This can be done by writing them in the command line, immediately after the program name
- The program's inputs must be **separated by spaces**

Example

The program sumTwoNumbers sums two numbers.

We can pass the two input numbers directly when we invoke the program's executable (instead of the usual I/O operations, such as printing to command line the message "please insert two numbers:", followed by fgets() etc.)

```
./sumTwoNumbers  3  5
```

Clarguments.c

Command Line Arguments

- Input parameters of the function main()
- `argc, argv`

```
int main( int argc, char* argv[] )
```

`argc`
- Integer
- Specifies the **number** of arguments on the command line (including the program name)

`argv`
- Array of strings
- Contains the actual **arguments** on the command line
- First element is the name of the program

Command line arguments

It is a good habit, especially when a program takes input arguments, to specify in a **header** on the top of the main file:

- Program name and purpose
- Program usage: syntax to use to invoke (run) the program with input arguments
- Description of input arguments
- Description of output from the program

It is common to add a **–help** option to print the relevant information about program usage and input arguments

calculator.c

Command line arguments

Example

Program calculator, reads two numbers, the operator, and prints the result

Linux Wildcard Characters

Linux has a series of wildcard characters * ? [ ]

* Represents strings of arbitrary length containing any possible character

***** all items (directories and files) - with or without a suffix

r* items beginning with the letter "r"

boot* items beginning with "boot"

mem all items contain "mem" anywhere in the name

***.png** items having the suffix of ".png" - that end in ".png"

We must be very careful when we use wildcard characters as input, because argc and argv recognize them!

Linux Wildcard Characters

Linux has a series of wildcard characters * ? [ ]

? Represents one single character which has any possible value

?.txt items starting with only one character and ending in ".txt"
Examples: b.txt and 3.txt

memo?.sxw items beginning with "memo", having a single character after "memo", and having the suffix of ".sxw"
Examples: memo1.sxw and memoh.sxw - not memo23.sxw

memo??.sxw items beginning with "memo", having a two characters (only) after "memo", and having the suffix of ".sxw"
Examples: memo21.sxw and memok9.sxw - not memos.sxw

We must be very careful when we use wildcard characters as input, because argc and argv recognize them!

Linux Wildcard Characters

Linux has a series of wildcard characters * ? [ ]

[ ] Represents intervals of characters values

[a-z]* items that begin with any lower case letter and end in any other characters

[A-Z]-list.dat items that begin with any upper case letter and end in "-list.dat"

[a-zA-Z]report.sxc items that begin with any lower case or upper case letter and end in "report.sxc“

[e-t].c items that begin with any lower case letter between ‘e’ and ‘t’ and end in “.c”

We must be very careful when we use wildcard characters as input, because argc and argv recognize them!

Homework 3 Solution