Casey Burkhardt

Due: Monday, September 08, 2008

Lab 1

**Exercise 1.**

**Step 1.** Make sure you are in the directory csc2400/Lab1 of your home directory. Check this using the Unix command

 pwd

The directory listed should end in csc2400/Lab1.

**Step 2.** Invoke the pico editor using the command

 pico test1.c

and type in the following program. Use the menu listed at the bottom of the pico editor for help on text editting.

 #include <stdio.h>

 int main()

 {

 float x;

 int y;

 printf("\n Enter a float, then an int: \n");

 scanf("%f %d", &x, &y);

 printf("You have entered integer %d and float %f \n", y, x);

 return 0;

 }

Save and exit your program (CTRL-O, then CTRL-X). [Here](http://www.csc.villanova.edu/~mdamian/C/c-input-output.htm) is all you need to know about printf and scanf.

**Step 3.** Open another terminal on tanner (follow [these](http://www.csc.villanova.edu/~mdamian/unix/login.htm) login steps). In this new terminal window, compile your program from Step 2 using the gcc compiler

 gcc test1.c

What files are created?

 **No files were created.**

Run the command

 ls -l

to list all files and associated permissions ('r' stands for read, 'w' stands for write and 'x' stands for execute). Run the executable you find.

**Step 4.** Compile again your program test1.c using the command:

 gcc -o test1 test1.c

What new file is created? Run that file. ***What does the -o flag do?***

**The –o flag produces an output, or executable of the code.**

**Step 5.** Run your executable test1 again using

 test1 > junk

Type in the two numbers your program expects: an integer and a float, then hit ENTER.

What do you expect to happen?

 **Output to be piped to a file called “junk”**

What happened?

**Output was placed in a file called junk in the working directory.**

          ***What does the operator > do in Unix?***

**It pipes output to the provided location.**

The operator > is called ***redirection*** operator: redirect output to the file with the specified name.

**Exercise 2.**

Write a C program called test2.c that reads two integer values from the user and prints out their sum. Create an executable called test2 and run it.

Sample output:

 Enter the first integer value: 6

 Enter the second second integer value: 5

 The sum of 5 and 6 is 11

[Here](http://www.csc.villanova.edu/~mdamian/C/c-input-output.htm) is all you need to know about printf and scanf.

**include <stdio.h>**

**int main()**

**{**

 **int x, y, z;**

 **printf("\nEnter the first integer value:\n");**

 **scanf("%d", &x);**

 **printf("\nEnter the second integer value:\n");**

 **scanf("%d", &y);**

 **z = x + y;**

 **printf("\nThe sum of %d and %d is %d.\n", x, y, z);**

 **return 0;**

**}**

**Exercise 3.**

C, C++ and Java allow you to input values into your program through the command line. You will only need to declare the main function in a slightly different form. In C (and C++), this is

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

* argc is the number of command line arguments (including the name of the executable)
* argv is an array of the arguments

Type the following program in a file called test3.c:

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

 {

 int i;

 printf("\nYour program name is %s\n", argv[0]);

 printf("Your arguments are: \n");

 for(i = 1; i < argc; i++)

 printf("\t \t %s starts with %c \n", argv[i], argv[i][0]);

 return 0;

 }

Create an executable called test3 and run it using the command

 test3 let us see what this does

**Your program name is ./test3**

**Your arguments are:**

 **let starts with l**

 **us starts with u**

 **see starts with s**

 **what starts with w**

 **this starts with t**

 **does starts with d**