UNBC Computer Science Program
Term: Fall 2017
Class time: T 8:30 - 11:20
Classroom: LIB 5-405
Lab time: F 11:30 - 12:50
Lab space: LAB 8-361

Instructor: Jernej Polajnar
Office: TAL 10-2044
Phone: 250-960-6624
E-mail: polajnar@unbc.ca
Office hours: TBA
& by appointment
Lab Instructor: Curtis Bowden
E-mail: curtis.bowden@unbc.ca

CPSC 321-3 Computer Operating Systems

Lab 2: C language, include, pointers, references

Includes:

The **include** directive in C is similar to using an import in Java. You are already familiar with including system libraries such as **<stdio.h>**. You can include your own libraries with a simliar syntax **include "fooheader.h"**. Using quotations tells the compiler to look in the current directory '.' for a file called "fooheader.h" in this case. For lab2 you will write a program called lab2.c and include the file "square.h" so that you can use the functions contained in "square.c" within your main program. Most if not all C compilers recognize files with a .h extension as a header file and will look for function declarations within.

Function declarations are in square.h:

```
/*
        CPSC444 Lab2 C language, include, pointers, references
        square.h
        2017-09-21
*/

int square_value(int );

int square_reference(int *);
```

```
/*
        CPSC444 Lab2 C language, include, pointers, references
        square.c
        2017-09-21
*/


//When a function is call by value a copy of the variable is made.
int square_value(int x)
{

        int p = x * x;

        return p;

}


//When a function is call by reference the memory address of the variable is passed.
int square_reference(int *x_ptr)
{

        //Multiply the value at *x_ptr by the value at *x_ptr
        int p = *x_ptr * *x_ptr;

        return p;

}
```

Makefiles

You can use a Makefile to save yourself a lot of typing of compiling and linking commands. When you have multiple .c and .h files you must compile each program individually and link them together, or compile and link them in one long command which can be difficult and inpractical when there are more than a couple files in your project. The following is an example of a make file that can be used for todays lab, and modified for future labs. An example of what the output from make looks like is at the end of this document with the example program output.

```
###### _._ ########################################################################
#     /_ _`.
#     (.X.)|    MakeFile
#     |\_/'|   ~~~~~~~~~~~~~~~
#     )____`\     copyright   : (C) 2003 by Curtis Bowden
#    //_V _\ \    email       : bowdenc@unbc.ca
#   ((  |  `(_)
#  / \> '   / \
#  \  \.__./  /
#   `-'    `-'
###################################################################################

BIN      = lab2
COMPILER = gcc
CPPFLAGS =
TARGETS  = lab2.o square.o
SOURCES = lab2.c square.c Makefile

$(BIN) : $(TARGETS)
        $(COMPILER) $(CPPFLAGS) -o $(BIN) $(TARGETS)

%.o: %.c
        $(COMPILER) $(CPPFLAGS) -c $<

script:
        cat $(SOURCES)

clean :
        rm -f $(TARGETS) $(BIN) core *.bck


###################################################################################
```

Main Program

lab2.c includes **square.h** and utilizes two different versions of the square function; One is call by reference and one is call by value. The program then demonstrates how to move a pointer to traverse an array and see the memory address of the pointer as it is incremented.

```
/*
        CPSC444 Lab2 C language, include, pointers, references
        lab2.c
        2017-09-21
*/
```

```

//Include a system library files
#include <stdio.h>

//Include square.h so you can use your square functions.
#include "square.h"

//Include your own function here.


int main()
{

        printf("Enter a number to square:");

        int number = 0;

        int *number_ptr = &number;

        // Read a number from the keyboard.
        scanf("%d", &number);
        printf("\n");

        // Call by value
        printf("Call by value result:%d\n", square_value(number));
        printf("\n");

        // Call by reference
        printf("Call by reference result:%d\n", square_reference(&number));
        printf("\n");

        printf("The address number_ptr is pointing to: %p\n", number_ptr);
        printf("\n");

        printf("The value of number_ptr is: %d\n", *number_ptr);
        printf("\n");

        int num_array[4] = {10, 20, 30, 40};

        //Set a pointer to the first address of the array num_array[4]
        int *array_ptr = num_array;

        //Count from zero to three and move the pointer
        //The purpose of this loop is to show how to iterate over an array using a pointer!
        for( int i=0; i<=3; ++i, ++array_ptr)
        {
                //This line increases the value stored where the pointer is pointing.
                ++*array_ptr;

                printf("num_array[%d] address:%p value:%d\n", i, array_ptr, *array_ptr);

        }

        return 0;

}
```

Example output.

```
curtis$ make
gcc  -c lab2.c
gcc  -c square.c
gcc  -o lab2 lab2.o square.o
curtis$
curtis$ ./lab2
Enter a number to square:5

Call by value result:25

Call by reference result:25

The address number_ptr is pointing to: 0x7fff5ea42a48

The value of number_ptr is: 5

num_array[0] address:0x7fff5ea42a50 value:11
num_array[1] address:0x7fff5ea42a54 value:21
num_array[2] address:0x7fff5ea42a58 value:31
num_array[3] address:0x7fff5ea42a5c value:41
```

Tasks

1. Read through the example programs paying close attention to the comments.
2. Create, compile and run the eample programs.
3. Create a swap function that takes two integers and swaps them. Hint: You will want to do this by reference, not value.
4. Challenge: make an integer variable and fork the program to see if you can have two processes modifying the variable.

Please submit your code files for review by Wednesday 2017-09-27 to *curtis.bowden@unbc.ca*