Lab 11: *Bitwise Operators*

Due: 11:59 PM, Friday, October 12, 2018

Description:

In this lab you will write a program that will contain two functions, *setlsbs*() and *getlsbs*(). These functions will use bitwise operators to embed and extract "hidden" bits in a character array.

Background Preparation:

- Review bitwise operators (Section 2.9 in textbook).

Specifications:

Your first function:

void setlsbs(unsigned char *p, unsigned char b0)

will take as parameters, an array *p* of eight bytes (unsigned char) and a byte *byte0*. It will replace the least significant bits (LSBs) of *p* by the bits of *byte0*. In other words, if the binary representation of *byte0* is$b_7b_6b_5b_4b_3b_2b_1b_0$, you should replace the LSB of *p*[0] by b_0, the LSB of *p*[1] by b_1, ... , and the LSB of *p*[7] by b_7.

Your second function:

unsigned char getlsbs(unsigned char *p)

will take an array *p* of eight bytes (unsigned char) and return a byte *byte0* which is created by combining the LSBs of *p*. That is,

your function should combine the least significant bits b_i of $p[i]$ to return a byte$b_7b_6b_5b_4b_3b_2b_1b_0$.

Write a program to test your functions as follows:

- Obtain a random number seed from the command line of your program using command line arguments.
- Initialize an array p of 8 unsigned char with random numbers from 0 to 255
- Initialize a separate unsigned character *byte0* with a random number.
- Print the values in the array *p* as well as the value for *byte0*.
    - Print the values in decimal format as well as binary format (use macros defined below)
- Call setlsbs() using *p* and *byte0* as parameters
- After the call to setlsbs() is completed, print the modified values of the array *p*.
    - Print the values in decimal format as well as binary format (use macros defined below)
- Use the modified array *p* as a parameter to getlsbs()
- Print the return value of the call to getlsbs(). The returned value should match the original value for *byte0*
    - Print the value in decimal format as well as binary format (use macros defined below)

You will create a Makefile to compile and link your programs. Your program filename should be Lab6_<*username*>_<*labsection*>.c

Macros:

You may use the following macros to print the binary representation of unsigned character variables:

```
#define BYTETOBINARYPATTERN "%d%d%d%d%d%d%d%d"

#define BYTETOBINARY(byte)  \
  (byte & 0x80 ? 1 : 0), \
  (byte & 0x40 ? 1 : 0), \
  (byte & 0x20 ? 1 : 0), \
  (byte & 0x10 ? 1 : 0), \
  (byte & 0x08 ? 1 : 0), \
  (byte & 0x04 ? 1 : 0), \
  (byte & 0x02 ? 1 : 0), \
  (byte & 0x01 ? 1 : 0)

#define PRINTBIN(x) printf(BYTETOBINARYPATTERN, BYTETOBINARY(x));
```

You can use the macros in a manner similar to the code below:

unsigned char num =173;

PRINTBIN(num); printf("\n");

Submission:

Log into zeus and copy your lab's files into a directory named *Lab6_<username>_<labsection>*. Start a typescript session in this directory. Type "uname -a" to show that you are on zeus. Use the *ls* and *cat*commands to list all your source files and Makefile. Compile your program using the *make* command. Run your program using 10 different random seeds to show that it executes correctly, then exit the typescript session. Verify that your script file was created correctly by using the *more* (or *less*) command (your choice) to see the contents of the file. As always, information on these commands can be obtained by using the manual pages ("man more" or "man less").

Change back to the directory immediately above *Lab6_<username>_<labsection>*, and use

the *tar* command to create an archive of all the files in the *Lab6_<username>_<labsection>* directory. Name this archive *Lab6_<username>_<labsection>.tar*. Re-read the man page if necessary to discover the proper options to use.

Once you are sure the tarfile is correct (check it by extracting it in another directory), copy it back to your local computer and submit it to Blackboard as Lab 6.