Bits, Bytes, Bit Operations and image processing

15-123

Systems Skills in C and Unix

Midterm

- Thursday or Friday from 7-9 pm
- No class thursday
- Given in GHC 5205 cluster, linux machines, same as recitation
- Access to man pages, no internet
- Exam Format
  - Starter code given
  - Write few functions
  - Compile and Test with make files and testers provided.
- Exam Topic
  - hashtables

Structures used

```
typedef struct HASH_NODE {
  char *key;                /* pointer to the key for node */
  char *value;              /* pointer to the value for the node */
  struct HASH_NODE *next;   /* pointer to next node */
} hash_node;
```

```
typedef struct hashtable {
        hash_node **table;              /* head node of the list */
        int size;               /* number of cells*/
        int cellsused; /* cells used */
        int numnodes;  /* number of nodes in the table */
        double loadfactor; /* lf = numnodes/cellsused */
        int (*hashfn)(char*,int);               /* hash function */
        int (*equal)(const void*, const void*); /* equal function pointer */
        void (*free_key)(void*); /* takes a pointer to a key and frees it*/
        void (*free_value)(void*); /* takes a pointer to a value and frees it*/
} hashtable;
```

topics

- bits, bytes and words
- data and instructions
- representation of data using hexadecimal
- signed and unsigned ints
- Two's compliment and negative numbers
- Left shift (<<), right shift (>>)
- Bit operations: negation(~), xor(^), or(|) and (&)
- setbit and getbit
- Binary files
  - fread and fwrite
- Manipulating bitmaps

Representing Information

- Smallest Data unit is the *"bit"*
- Smallest addressable unit is the *"byte"*
- Each computer has a *"word"* size
  - Amount of memory transferred between CPU and RAM
  - Indicate the nominal size of integers and pointers
  - Most common size is 32-bits
  - How many addressable units are there then?

Question

- If a computer has 32-bit word size, what would be the range of virtual address space?

- What if the computer is a “64-bit” machine?

Data Sizes

- Here are the typical 32-bit allocation for data types (in bytes)
  - char (1), short int (2), int (4), long int (4)
    - In compaq alpha long int is 8
  - char* (4), float (4), double (8)
- The exact size of data allocation depends on both compiler and machine

Data value ranges

- <limits.h> library defines the range of values any data type can assume.
- Applications that are designed to be portable must use symbolic constants.
- **Some examples**
  - **INT_MIN**
    - Minimum value for a variable of type **int**.
    - –2147483647 – 1
  - **INT_MAX**
    - Maximum value for a variable of type **int**.
    - 2147483647
  - **UINT_MAX**
    - Maximum value for a variable of type **unsigned int**.
    - 4294967295 (oxffffffff)
  - **LONG_MIN**
    - Minimum value for a variable of type **long**.
    - –2147483647 – 1
  - **LONG_MAX**
    - Maximum value for a variable of type **long**.

Storage Classes

- auto
  - Typical variables defined inside functions
- static
  - Variables that retain values between function calls
- extern
  - Declared within a function, but specifications given else where
- register

Representation formats

- Binary
- Octal
- Decimal
- Hexadecimal

Addressing and byte ordering

- Little Endian
  - Least significant byte first (DEC, Intel)
- Big Endian
  - Most significant byte first (IBM, Motorola, SUN)
- Application programmers may not care about this ordering

When byte ordering becomes an issue

- Communication of binary data over a network between different machines
- Code written for networking applications must then do their own conversions between machines

Integer Representations

- Typical 32-bit machine uses
  - 32-bit representation for int and unsigned
    - Range:
  - Compaq alpha uses 64 bits for long int
    - Range:

Closer look at signed and unsigned integers

- Consider a n-bit integer representation of an unsigned integer

- Consider a n-bit integer representation of a signed integer

Representing negative numbers using 2's complement

- One's complement

  ~x

- Two's complement

  1 + ~x

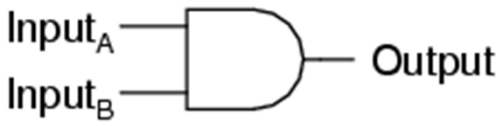
Signed and unsigned numbers

- By default all constant values are signed
  - int x = 20, y =0x45
- Can create an unsigned constant using
  - unsigned x = 0x123u (or U)

Bit Operations in C

- Bitwise AND (&)
  - 0x75 & 0x 96

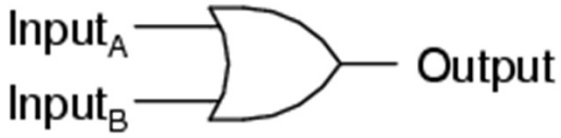
AND gate



A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

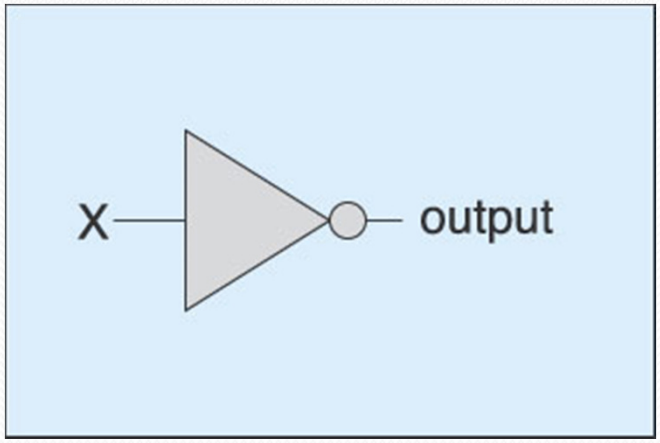
Bitwise OR (|)

2-input OR gate



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

Bitwise negation (~)



XOR (^)

Exclusive-OR gate

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

Logic for adding bit by bit

- Si = (Ai ^ Bi) ^ Cin
- Cout = (Ai & Bi) | ((Ai ^ Bi) & Cin)

Bit adder

- Exercise: Given two unsigned char's write a bit-by-bit adder using above formulas. How would you recognize an overflow situation?

Logical Operations in C are different

- Logical AND (&&)
  - 0x75 && 0x 96
- Logical OR (||)
- Logical Not (!)

Shifting in C

- Left Shift (<<)

- Right shift (>>)

Counting number of 1's

- Let C(n) be the number of 1's in int n
- We want to know number of 1's in the binary representation of n. Why?
  - Eg: if the answer is 1, then we know one of two things
- Find an iterative solution
  - Shift 32 times and & with 1
- To find a recursive solution
  - What is the relation between C(n) and C(n/2)?
    - When n is even
    - When n is odd

getbit function

```
#define MASK(j) (1<<j)
int getBit(char w, unsigned j){
   return (( w & MASK(j)) == 0) ? 0 : 1;
}
```

- What is an alternative way to write this?

printBinary

- Complete the function printBinary

```
void printBinary(char w){




}
```

setbit function

```
#define MASK(j) (1<<j)
int setBit(char w, unsigned j, short value){
    if (value == 0) return (w & ~MASK(j));
    else if (value == 1) return w | MASK(j);
    else return w;
}
```

Masking

- Masking is a technique to extract bits from a value
- Eg: Determine if the number is even or odd

Exercise

- Complete the function bitReverse

```
/* reverse the bit pattern of the *ptr*/
void bitReverse(char* ptr, int numbits){




}
```

Bitmap format

- Developed by Microsoft
- Each pixel is represented by RGB
  - 3 bytes per pixel
- Each byte value vary from 0-255
  - 0- darker, 1-lighter
- Each bmp file has a header
  - 54 bytes

Header info
first 14 bytes

```
typedef struct {
    unsigned short int type;    /* BMP type identifier */
    unsigned int size;          /* size of the file in bytes*/
    unsigned short int reserved1, reserved2;
    unsigned int offset;        /* starting address of the byte */
} HEADER;
```

Binary Files

Binary Files

- Any file is a collection of bytes
- File can be read one byte at a time
  - fread
- Data can be written to a file one byte at a time
  - fwrite

NAME

fread, fwrite - binary stream input/output

SYNOPSIS

```
#include <stdio.h>

size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);

size_t fwrite(const void *ptr, size_t size, size_t nmemb,
              FILE *stream);
```

DESCRIPTION

The function fread() reads nmemb elements of data, each size bytes long, from the stream pointed to by stream, storing them at the location given by ptr.

The function fwrite() writes nmemb elements of data, each size bytes long, to the stream pointed to by stream, obtaining them from the location given by ptr.

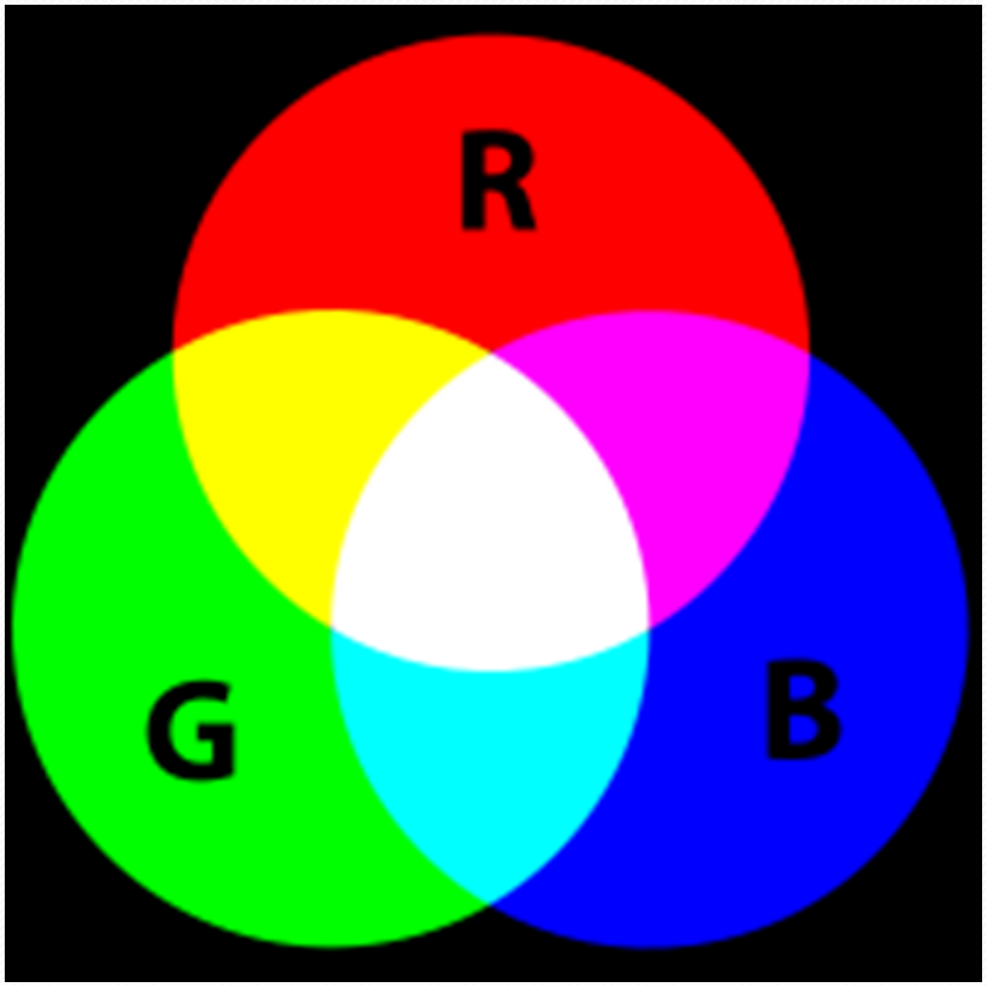
For non-locking counterparts, see unlocked_stdio(3).

RETURN VALUE

fread() and fwrite() return the number of items successfully read or written (i.e., not the number of characters). If an error occurs, or the end-of-file is reached, the return value is a short item count (or zero).

Image processing

RGB Color Scheme



Source: wikipedia

Exercises

- Read a BMP image and find its file size

Header Info
next 40 bytes

The next 40 bytes are reserved for a structure as follows.

```
typedef struct {
    unsigned int size;              /* Header size in bytes */
    int width,height;               /* Width and height in pixels */
    unsigned short int planes; /* Number of color planes */
    unsigned short int bits;    /* Bits per pixel */
    unsigned int compression; /* Compression type */
    unsigned int imagesize;   /* Image size in bytes */
    int xresolution,yresolution;            /* Pixels per meter */
    unsigned int ncolors;       /* Number of colors */
    unsigned int importantcolors; /* Important colors */
} INFOHEADER;
```

Exercises

- Read a BMP image and find its length and width

Application
Image Processing

512 x 512 image

Application
Dealing with Byte alignments

361x315 image
File size = 342228

Exercises

- Remove red color altogether from an image

- Make a color RGB image BW
  - hard

Bit packing

```
struct{
    unsigned leading : 3;
    unsigned flag1 : 1;
    unsigned flag2 : 1;
    trailing : 11;
} flags;
```

- fields within the struct are not variables
  - cannot be used with & the address operator

- printf("The leading field is %d \n", flags.leading);

Code Examples