UNIVERSITY AT ALBANY
State University of New York

COLLEGE OF ENGINEERING AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

ICSI333 System Fundamentals

Note: Students are expected to start the activities as soon as the description is available and seek feedback as needed. Although some activities are not graded for credit, they are contiguous study of the lecture or used as stepping-stones for the projects. Skipping any activities would impact the learning significantly.

Objectives:

- Practice bitwise operations

Reading:

- Lecture notes
- Reading materials

Submission (5 points):

- All required C programs and explanations must be submitted on Duifene on time.

Instructions:

The following shows the list of bitwise operations in C.

Operator	Name	Description
~	Complement (NOT)	All 0 bits are set to 1, and all 1 bits are set to 0. This is often called toggling the bits.
&	bitwise AND	Compares its two operands bit by bit. The bits in the result are set to 1 if the corresponding bits in the two operands are **both** 1.
|	bitwise inclusive OR	Compares its two operands bit by bit. The bits in the result are set to 1 if **at least one** of the corresponding bits in the two operands is 1.
^	bitwise exclusive OR (also known as bitwise XOR)	Compares its two operands bit by bit. The bits in the result are set to 1 if the corresponding bits in the two operands are different.
<<	left shift	Shifts the bits of the first operand left by the number of bits specified by the second operand; **fill from the right with 0 bits**.
>>	right shift	Shifts the bits of the first operand right by the number of bits specified by the second operand; **the method of filling from the left is machine-dependent** when the left operand is **negative**. For unsigned numbers, the vacated bits are zero-filled. For signed numbers, the result is **machine-dependent**. In most cases, the sign bit 1 is used to fill the vacated bit positions.

Task #1. Study and discuss the following examples.

Example 1: The result of right-shifting of some numbers is implementation-defined or undefined. Execute the following program containing the bitwise expressions and explain the results.

- -1 << 1
- 1 << -1
- -1 >> 1
- 1 >> 1
- 1 << 33
- 1 << 64

```
#include <stdio.h>
```

```
int main(void){
      printf("%d %d %d %d %d %d \n",
                  -1 << 1, 1 << -1, -1 >> 1, 1 >> 1, 1 << 33, 1 << 64);
}
```

Example 2: A single right shift divides a number by 2; a single left shift multiplies a number by 2. Execute the following program and explain the results.

```
#include<stdio.h>

int main(){
      int x = 19;
      printf ("x << 1 = %d\n", x << 1);
      printf ("x >> 1 = %d\n", x >> 1);

      return 0;
}
```

Example 3: The bitwise operators should not be used in place of logical operators. The result of logical operators (`&&`, `||` and `!`) is either `0` or not `0`(usually `1`), but bitwise operators may return an integer value between `0` and $2^n - 1$, where `n` is a size of machine word in bits. . Execute the following program and explain the results.

```
#include<stdio.h>
int main(){

      int x = 2, y = 5;
      (x & y)? printf("True ") : printf("False ");
      (x && y)? printf("True ") : printf("False ");

      return 0;
}
// Output: False True
```

Example 4: The `&` operator can be used to check if a number is odd or even. The value of the expression (`x & 1`) would be non-zero only if x is odd, zero otherwise. Execute the following program and explain the results.

```
#include<stdio.h>
int main(){

      int x = 19;
      (x & 1)? printf("Odd"): printf("Even");

      return 0;
}
// Output: Odd
```

Task #2: Write a program to count the number of bits 1 in a binary expansion of an integer.
The program must prompt the users to enter an integer as a decimal expansion, and then, use the bitwise operators (Hint: use & and >>.) to count the number of bits 1 in the binary expansion of the integer.

For example: when executing the object file of `myProgram` with command-line argument `48`, the following output should be produced.

```
./myProgram 48
Your number was 48
In 48, there are 2 bits set to 1.
```

If needed, study the following examples for command-line arguments.

```
#include<stdio.h>

int main(int argc, char** argv) {
      printf("This program has %d arguments\n",argc);

      for(int i = 0; i<=argc; i++)
            printf("argv [ %d ] = %s\n",i, argv[i]);

      return 0;
}
```

Task #3. Study the following examples and explain the results.

Example 1: Given an array in which all elements occur even number of times except one element, find the element that occurs odd number of times using XOR operator.

```
#include<stdio.h>
// Function to return the only odd occurring element
int findOdd(int arr[], int n){
      int res = 0, i;

      for (i = 0; i < n; i++)
            res ^= arr[i];
      return res;
}

int main(void){
      int arr[] = {12, 12, 14, 90, 14, 14, 14};
      int n = sizeof(arr)/sizeof(arr[0]);

      printf ("The odd occurring element is %d ", findOdd(arr, n));

      return 0;
}
// Output: The odd occurring element is 90
```

Example 2: A function that uses >> and & operators to find and display the binary expansion of a given number.

```
#include<stdio.h>

void binary(unsigned int);

void main(){
      unsigned int num;

      printf("Enter Decimal Number : ");
      scanf("%u",&num);
      binary(num); // Function Call
      getchar();
}

void binary(unsigned int num){
      unsigned int mask=32768;          //mask = 1000 0000 0000 0000
      printf("Binary Eqivalent : ");
      while(mask > 0){
            if((num & mask) == 0 )
                  printf("0");
            else
                  printf("1");
      mask = mask >> 1;               //Right Shift
     }
}
```

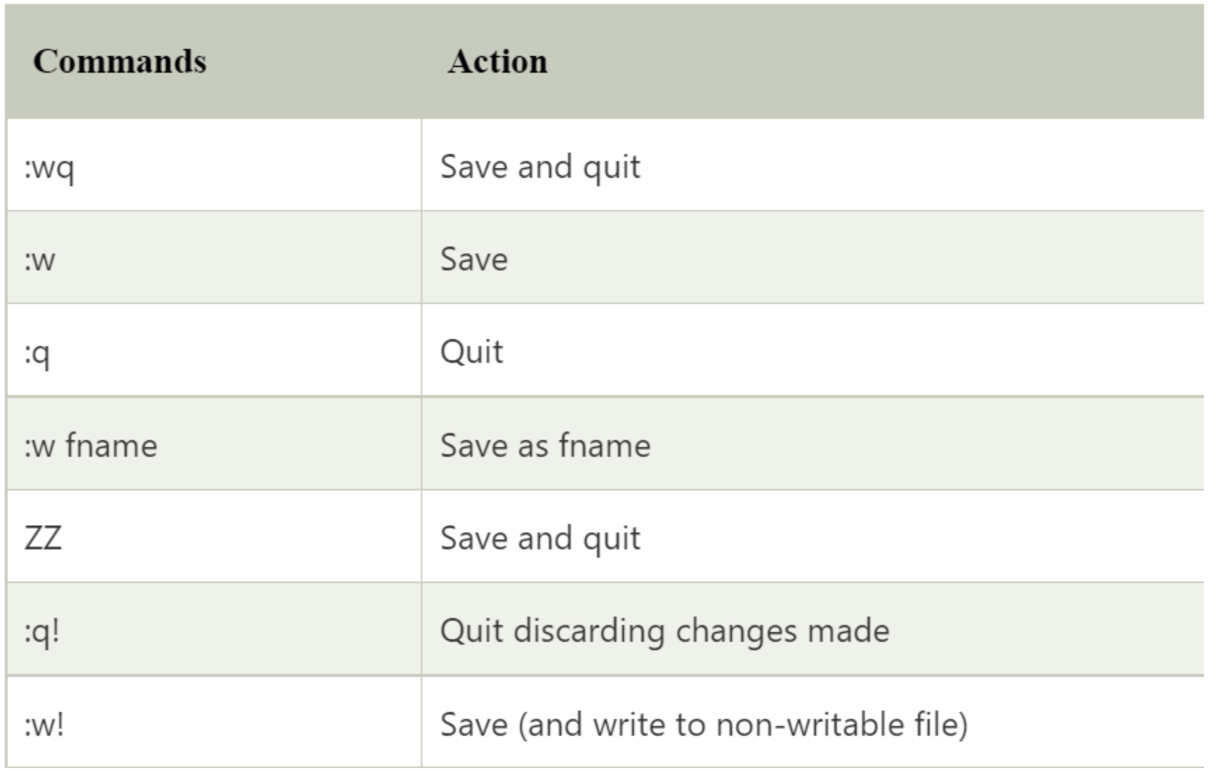
Linux Basic Commands

`ls [option(s)] [file(s)]`	If you run ls without any additional parameters, the program will list the contents of the current directory in short form. `-l` detailed list `-a` displays hidden files
`cp [option(s)] sourcefile targetfile`	Copies sourcefile to targetfile. `-I` waits for confirmation, if necessary, before an existing targetfile is overwritten `-r` copies recursively (includes subdirectories)
`mv [option(s)] sourcefile targetfile`	Copies sourcefile to targetfile then deletes the original sourcefile. `-b` creates a backup copy of the sourcefile before moving. `-I` waits for confirmation, if necessary, before an existing targetfile is overwritten.
`rm [option(s)] file(s)`	Removes the specified files from the file system. Directories are not removed by rm unless the option `-r` is used. `-r` deletes any existing subdirectories `-I` waits for confirmation before deleting each file
`cd [options(s)] [directory]`	Changes the current directory. cd without any parameters changes to the user's home directory.
`mkdir [option(s)] directoryname`	Creates a new directory.
`rmdir [option(s)] directoryname`	Deletes the specified directory, provided it is already empty.
`cat [option(s)] file(s)`	The cat command displays the contents of a file, printing the entire contents to the screen without interruption. `-n` numbers the output on the left margin
`cal`	Displays the calendar of the current month.
`date`	Displays current time and date.
`whoami`	Reveals the user who is currently logged in.
`whatis`	Gives a one-line description about the command. It can be used as a quick reference for any command.

`man`	Manual Pages, for more detailed information, Linux provides man pages and info pages. To see a command's manual page, man command is used.
`pwd`	Prints the absolute path to current working directory.
`vi`	A text editor for Linux operating system.
`gedit`	The default GUI text editor in the Ubuntu operating system.
`emacs`	Another text editor for Linux operating system.

Vi commands:

To save and quit:



Commands	Action
:wq	Save and quit
:w	Save
:q	Quit
:w fname	Save as fname
ZZ	Save and quit
:q!	Quit discarding changes made
:w!	Save (and write to non-writable file)

More commands:

Copy-pasting within the terminal: Ctrl+Shift+C/V

Resources and Credits:

- Teaching materials from Professor Kuperman at UAlbany
- Vi reference: https://www.ele.uri.edu/faculty/vetter/Other-stuff/vi/vi-quick-ref.pdf