1. A. <https://www.javatpoint.com/kali-linux-installation>

B. <https://www.geeksforgeeks.org/how-to-install-virtual-box-in-kali-linux/>

1. <https://www.geeksforgeeks.org/linux-directory-structure/>
2. A. <https://www.javatpoint.com/linux-commands>

B. <https://www.redhat.com/sysadmin/introduction-vi-editor>

1. Theory related to 10 to 12 Networking devices(8 Network device write in 2.3 lecture in class). (If you know about create topology then 1 topology)[Cisco packet Tracer]
2. **Hello print program**:

#include <stdio.h>

#include <unistd.h>

#define MSGSIZE 16

char\* msg1 = "hello friends";

char\* msg2 = "hello, world #2";

char\* msg3 = "hello, world #3";

int main()

{

char inbuf[MSGSIZE];

int p[2], i;

if (pipe(p) < 0)

exit(1);

/\* continued \*/

/\* write pipe \*/

write(p[1], msg1, MSGSIZE);

write(p[1], msg2, MSGSIZE);

write(p[1], msg3, MSGSIZE);

for (i = 0; i < 3; i++) {

/\* read pipe \*/

read(p[0], inbuf, MSGSIZE);

printf("% s\n", inbuf);

}

return 0;

}

1. **Character count program in string:**

|  |  |
| --- | --- |
|  | #include <stdio.h>  #include <string.h>    int main()  {      char s[1000],c;      int i,count=0;        printf("Enter  the string : ");      gets(s);      printf("Enter character to be searched: ");      c=getchar();        for(i=0;s[i];i++)      {       if(s[i]==c)       {            count++;  }  }    printf("character '%c' occurs %d times \n ",c,count);            return 0;  } |

1. **Bits count program stuff :**

#include <stdio.h>

int countSetBits(int n) {

int count = 0;

while (n) {

count += n & 1;

n >>= 1;

}

return count;

}

int main() {

int num;

printf("Enter an integer: ");

scanf("%d", &num);

int result = countSetBits(num);

printf("Number of set bits in %d: %d\n", num, result);

return 0;

}

1. **Perform a GNU C program to generate frames from sender’s message by splitting message by given frame-length.**

#include <stdio.h>

#include <string.h>

#define MAX\_MESSAGE\_LENGTH 1000

void generateFrames(char \*message, int frameLength) {

int messageLength = strlen(message);

int numFrames = (messageLength + frameLength - 1) / frameLength; // Calculate the number of frames needed

int i, j;

printf("Frames:\n");

for (i = 0; i < numFrames; i++) {

printf("Frame %d: ", i + 1);

for (j = 0; j < frameLength && (i \* frameLength + j) < messageLength; j++) {

printf("%c", message[i \* frameLength + j]);

}

printf("\n");

}

}

int main() {

char message[MAX\_MESSAGE\_LENGTH];

int frameLength;

printf("Enter the message: ");

fgets(message, sizeof(message), stdin);

message[strcspn(message, "\n")] = '\0'; // Remove trailing newline

printf("Enter the frame length: ");

scanf("%d", &frameLength);

generateFrames(message, frameLength);

return 0;

}

1. **II) Character Stuffing Program :**

#include <stdio.h>

#include <string.h>

#define MAX\_FRAME\_SIZE 100

void characterStuffing(char\* input, char\* stuffed, char delimiter) {

int i, j = 0;

stuffed[j++] = delimiter; // Start and end delimiter

for (i = 0; i < strlen(input); i++) {

if (input[i] == delimiter) {

stuffed[j++] = delimiter; // Escape the delimiter

stuffed[j++] = delimiter; // Duplicate the delimiter

} else {

stuffed[j++] = input[i];

}

}

stuffed[j++] = delimiter; // End delimiter

stuffed[j] = '\0'; // Null terminator

}

int main() {

char input[MAX\_FRAME\_SIZE];

char stuffed[MAX\_FRAME\_SIZE \* 2]; // Maximum possible stuffed frame size

char delimiter;

printf("Enter the frame: ");

fgets(input, sizeof(input), stdin);

input[strcspn(input, "\n")] = 0; // Remove newline character

printf("Enter the delimiter character: ");

delimiter = getchar();

getchar(); // Consume newline character

characterStuffing(input, stuffed, delimiter);

printf("Stuffed frame: %s\n", stuffed);

return 0;

}

1. **Byte Stuffing :**

#include <stdio.h>

#include <string.h>

int main() {

char frame[50][50], str[50][50];

char flag[10];

strcpy(flag, "flag");

char esc[10];

strcpy(esc, "esc");

int i, k = 0, n;

strcpy(frame[k++], flag);

printf("Enter length of String : \n");

scanf("%d", &n);

printf("Enter the String: ");

getchar(); // to clear the buffer

for (i = 0; i < n; i++) {

fgets(str[i], sizeof(str[i]), stdin);

str[i][strcspn(str[i], "\n")] = '\0'; // remove newline character

}

printf("\nYou entered :\n");

for (i = 0; i < n; i++) {

puts(str[i]);

}

printf("\n");

for (i = 0; i < n; i++) {

if (strcmp(str[i], flag) != 0 && strcmp(str[i], esc) != 0) {

strcpy(frame[k++], str[i]);

} else {

strcpy(frame[k++], esc);

strcpy(frame[k++], str[i]);

}

}

strcpy(frame[k++], flag);

printf("------------------------------\n\n");

printf("Byte stuffing at sender side:\n\n");

printf("------------------------------\n\n");

for (i = 0; i < k; i++) {

printf("%s\t", frame[i]);

}

return 0;

}

1. **Bit Stuffing Program:**

#include <stdio.h>

#include <string.h>

int main() {

char data[100], stuffedData[200];

int i, count = 0, j = 0;

printf("Enter the data: ");

scanf("%s", data);

for(i = 0; i < strlen(data); i++) {

if(data[i] == '1') {

count++;

stuffedData[j++] = data[i];

} else {

count = 0;

stuffedData[j++] = data[i];

}

if(count == 5) {

count = 0;

stuffedData[j++] = '0';

}

}

stuffedData[j] = '\0';

printf("Data after bit stuffing: %s\n", stuffedData);

return 0;

}

**Remaining Lab topics:**

LRC = 2D parity (Program No:11)

Checksum = Addition Method (Program No:11)

CRC = Division Method (Program No:12)

Hamming Code : (Program No:13)

Leaky Bucket : (Program No:14)

Token Bucket : (Program No:15)

**Program 11 LRC :**

#include <stdio.h>

// Function to calculate LRC

unsigned char calculateLRC(unsigned char \*data, int length) {

unsigned char lrc = 0;

for (int i = 0; i < length; i++) {

lrc += data[i];

}

// Take the one's complement of the sum

lrc = (~lrc) + 1;

return lrc;

}

// Function to print a byte in binary format

void printBinary(unsigned char byte) {

for (int i = 7; i >= 0; i--) {

printf("%d", (byte >> i) & 1);

}

}

int main() {

// Example data to be sent (replace this with your actual data)

unsigned char dataToSend[] = {0x41, 0x42, 0x43, 0x44}; // "ABCD" in ASCII

int dataLength = sizeof(dataToSend) / sizeof(dataToSend[0]);

// Calculate LRC for the data

unsigned char lrc = calculateLRC(dataToSend, dataLength);

// Append LRC to the data

dataToSend[dataLength] = lrc;

// Display the data with appended LRC in binary format

printf("Data with appended LRC (in binary):\n");

for (int i = 0; i < dataLength + 1; i++) {

printBinary(dataToSend[i]);

printf(" ");

}

printf("\n");

return 0;

}

**Program 11 Checksum:**

#include<stdio.h>

#include<math.h>

int sender(int arr[10],int n)

{

int checksum,sum=0,i;

printf("\n\*\*\*SENDER SIDE\*\n");

for(i=0;i<n;i++)

sum+=arr[i];

printf("SUM IS: %d",sum);

checksum=~sum; //1's complement of sum

printf("\nCHECKSUM IS:%d",checksum);

return checksum;

}

void receiver(int arr[10],int n,int sch)

{

int checksum,sum=0,i;

printf("\n\n\*\*\*RECEIVER SIDE\*\n");

for(i=0;i<n;i++)

sum+=arr[i];

printf("SUM IS:%d",sum);

sum=sum+sch;

checksum=~sum; //1's complement of sum

printf("\nCHECKSUM IS:%d",checksum);

}

void main()

{

int n,sch,rch;

printf("\nENTER SIZE OF THE STRING:");

scanf("%d",&n);

int arr[n];

printf("ENTER THE ELEMENTS OF THE ARRAY TO CALCULATE CHECKSUM:\n");

for(int i=0;i<n;i++)

{

scanf("%d",&arr[i]);

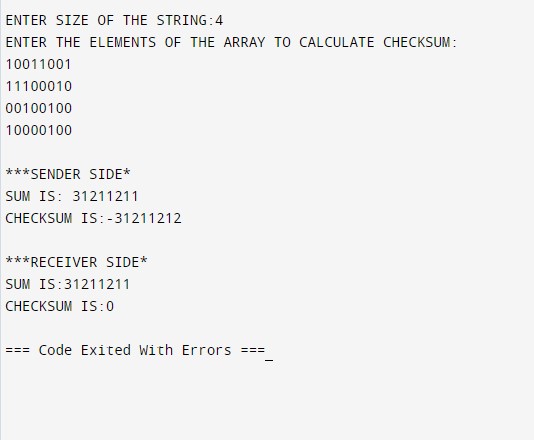
}

sch=sender(arr,n);

receiver(arr,n,sch);

}

**Output:**



**Program 12 CRC:**

#include<stdio.h>

#include<string.h>

// length of the generator polynomial

#define N strlen(gen\_poly)

// data to be transmitted and received

char data[28];

// CRC value

char check\_value[28];

// generator polynomial

char gen\_poly[10];

// variables

int data\_length,i,j;

// function that performs XOR operation

void XOR(){

// if both bits are the same, the output is 0

// if the bits are different the output is 1

for(j = 1;j < N; j++)

check\_value[j] = (( check\_value[j] == gen\_poly[j])?'0':'1');

}

// Function to check for errors on the receiver side

void receiver(){

// get the received data

printf("Enter the received data: ");

scanf("%s", data);

printf("\n-----------------------------\n");

printf("Data received: %s", data);

// Cyclic Redundancy Check

crc();

// Check if the remainder is zero to find the error

for(i=0;(i<N-1) && (check\_value[i]!='1');i++);

if(i<N-1)

printf("\nError detected\n\n");

else

printf("\nNo error detected\n\n");

}

void crc(){

// initializing check\_value

for(i=0;i<N;i++)

check\_value[i]=data[i];

do{

// check if the first bit is 1 and calls XOR function

if(check\_value[0]=='1')

XOR();

// Move the bits by 1 position for the next computation

for(j=0;j<N-1;j++)

check\_value[j]=check\_value[j+1];

// appending a bit from data

check\_value[j]=data[i++];

}while(i<=data\_length+N-1);

// loop until the data ends

}

int main()

{

// get the data to be transmitted

printf("\nEnter data to be transmitted: ");

scanf("%s",data);

printf("\n Enter the Generating polynomial: ");

// get the generator polynomial

scanf("%s",gen\_poly);

// find the length of data

data\_length=strlen(data);

// appending n-1 zeros to the data

for(i=data\_length;i<data\_length+N-1;i++)

data[i]='0';

printf("\n----------------------------------------");

// print the data with padded zeros

printf("\n Data padded with n-1 zeros : %s",data);

printf("\n----------------------------------------");

// Cyclic Redundancy Check

crc();

// print the computed check value

printf("\nCRC or Check value is : %s",check\_value);

// Append data with check\_value(CRC)

for(i=data\_length;i<data\_length+N-1;i++)

data[i]=check\_value[i-data\_length];

printf("\n----------------------------------------");

// printing the final data to be sent

printf("\n Final data to be sent : %s",data);

printf("\n----------------------------------------\n");

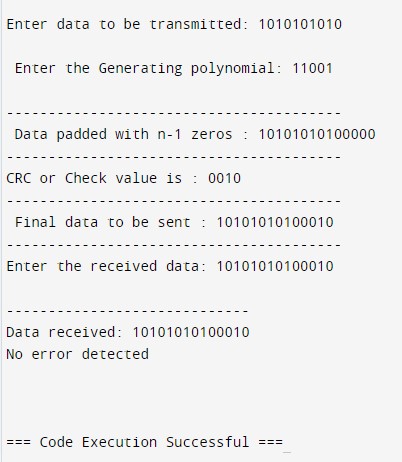
// Calling the receiver function to check errors

receiver();

return 0;

}

**Output:**



**Practical No. 13 Hamming Code :**

#include <stdio.h>

#include <math.h>

int input[32];

int code[32];

int ham\_calc(int,int);

void main()

{

int n,i,p\_n = 0,c\_l,j,k;

printf("Please enter the length of the Data Word: ");

scanf("%d",&n);

printf("Please enter the Data Word:\n");

for(i=0;i<n;i++)

{

scanf("%d",&input[i]);

}

i=0;

while(n>(int)pow(2,i)-(i+1))

{

p\_n++;

i++;

}

c\_l = p\_n + n;

j=k=0;

for(i=0;i<c\_l;i++)

{

if(i==((int)pow(2,k)-1))

{

code[i]=0;

k++;

}

else

{

code[i]=input[j];

j++;

}

}

for(i=0;i<p\_n;i++)

{

int position = (int)pow(2,i);

int value = ham\_calc(position,c\_l);

code[position-1]=value;

}

printf("\nThe calculated Code Word is: ");

for(i=0;i<c\_l;i++)

printf("%d",code[i]);

printf("\n");

printf("Please enter the received Code Word:\n");

for(i=0;i<c\_l;i++)

scanf("%d",&code[i]);

int error\_pos = 0;

for(i=0;i<p\_n;i++)

{

int position = (int)pow(2,i);

int value = ham\_calc(position,c\_l);

if(value != 0)

error\_pos+=position;

}

if(error\_pos == 1)

printf("The received Code Word is correct.\n");

else

printf("Error at bit position: %d\n",error\_pos);

}

int ham\_calc(int position,int c\_l)

{

int count=0,i,j;

i=position-1;

while(i<c\_l)

{

for(j=i;j<i+position;j++)

{

if(code[j] == 1)

count++;

}

i=i+2\*position;

}

if(count%2 == 0)

return 0;

else

return 1;

}

**OUTPUT :**

****

**Program 14 Leaky Bucket :**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h> // For sleep function

int main() {

int i, packets[10], content = 0, newcontent, time, clk, bucket\_size, output\_rate;

// Generate random packet sizes

for (i = 0; i < 5; i++) {

packets[i] = rand() % 10;

if (packets[i] == 0)

i--; // Regenerate if packet size is 0

}

printf("\nEnter output rate of the bucket: ");

scanf("%d", &output\_rate);

printf("\nEnter Bucket size: ");

scanf("%d", &bucket\_size);

for (i = 0; i < 5; ++i) {

if ((packets[i] + content) > bucket\_size) {

if (packets[i] > bucket\_size)

printf("\nIncoming packet size %d greater than the size of the bucket\n", packets[i]);

else

printf("\nBucket size exceeded\n");

} else {

newcontent = packets[i];

content += newcontent;

printf("\nIncoming Packet: %d\n", newcontent);

printf("Transmission left: %d\n", content);

time = rand() % 10;

printf("Next packet will come at: %d\n", time);

for (clk = 0; clk < time && content > 0; ++clk) {

printf("\nLeft time: %d", (time - clk));

sleep(1);

if (content > 0) {

printf("\nTransmitted\n");

if (content < output\_rate)

content = 0;

else

content -= output\_rate;

printf("Bytes remaining: %d\n", content);

} else {

printf("\nNo packets to send\n");

}

}

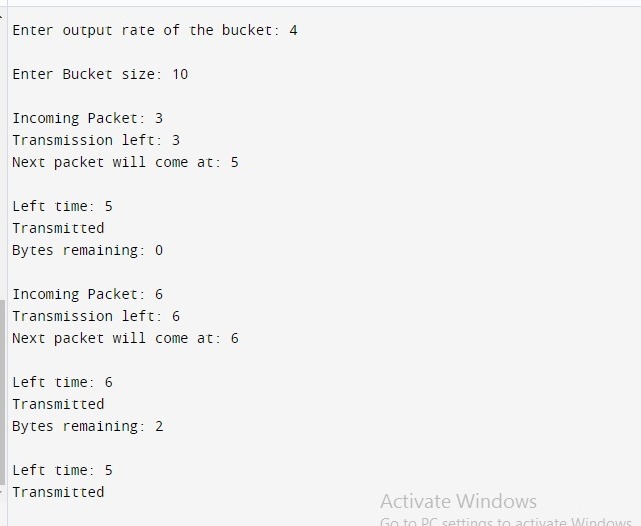
}

}

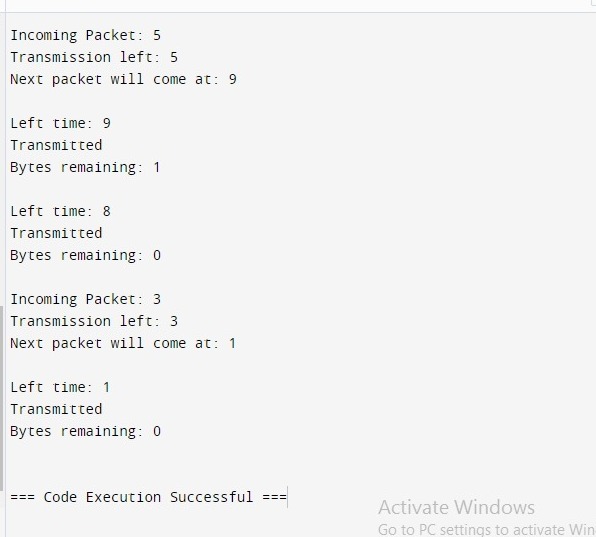
return 0;

}

**Output :**

****

****

****

**Program 15 Token Bucket :**

#include <stdio.h>

#include <stdbool.h>

#include <unistd.h> // for usleep function

int main() {

int bucket\_size, output\_rate;

// User input for bucket size and output rate

printf("Enter the bucket size: ");

scanf("%d", &bucket\_size);

printf("Enter the output rate of the bucket: ");

scanf("%d", &output\_rate);

int bucket = 0; // Current size of the bucket

while (true) {

// Generate some data, e.g., incoming packets

int incoming\_packets;

printf("Enter the number of incoming packets: ");

scanf("%d", &incoming\_packets);

// Add incoming packets to the bucket

if (bucket + incoming\_packets <= bucket\_size) {

bucket += incoming\_packets;

} else {

printf("Bucket overflow! Dropping %d packets.\n", incoming\_packets + bucket - bucket\_size);

bucket = bucket\_size; // Bucket is full

}

// Transmit data from the bucket

if (bucket >= output\_rate) {

printf("%d packets transmitted.\n", output\_rate);

bucket -= output\_rate;

} else {

printf("Bucket empty.\n");

}

// Wait for a second before the next iteration

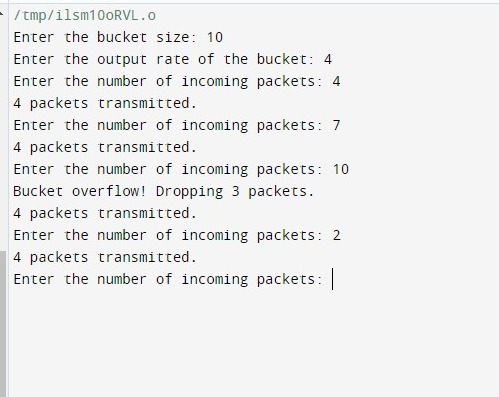
usleep(1000000); // Sleep for 1 second (1 million microseconds)

}

return 0;

}

**Output :**

****