**Name-Ujjawal Mandhani**

**Batch-F6**

**Enroll no.-9918103237**

**ISL LAB WEEK #1 D**

**Question 1:**Write a client server program to transfer a text file from client to server. Client side

will first ask about name of file. Client side will open the file in read + write mode. Read contents

(maximum of 1024) transfer encrypted text to server. Server will receive filename first and then

encrypted data. Server will create file with same name, decrypt the received message and store it in

file, and terminate the connection. Use Caesar cipher, monoalphabetic cipher or hill cipher. You can

use any key of your choice.

**SERVER.c:**

**#include <stdio.h>**

#include <stdlib.h>

#include <string.h>

#include <arpa/inet.h>

#define SIZE 1024

void write\_file(int sockfd)

{

int n;

FILE \*fp;

char \*filename = "recv.txt";

char buffer[SIZE];

fp = fopen(filename, "w");

while (1)

{

n = recv(sockfd, buffer, SIZE, 0);

if (n <= 0)

{

break;

return;

}

printf("Data Recieved: %s \n", buffer);

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

{

if (buffer[i] == '\n')

{

break;

}

buffer[i] = buffer[i] - 3;

}

fprintf(fp, "%s", buffer);

printf("Decrypted data: %s \n", buffer);

//bzero(buffer, SIZE);

}

return;

}

int main()

{

char \*ip = "127.0.0.1";

int port = 8080;

int e;

int sockfd, new\_sock;

struct sockaddr\_in server\_addr, new\_addr;

socklen\_t addr\_size;

char buffer[SIZE];

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd < 0)

{

perror("[-]Error in socket");

exit(1);

}

printf("[+]Server socket created successfully.\n");

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = port;

server\_addr.sin\_addr.s\_addr = inet\_addr(ip);

e = bind(sockfd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr));

if (e < 0)

{

perror("[-]Error in bind");

exit(1);

}

printf("[+]Binding successfull.\n");

if (listen(sockfd, 10) == 0)

{

printf("[+]Listening....\n");

}

else

{

perror("[-]Error in listening");

exit(1);

}

addr\_size = sizeof(new\_addr);

new\_sock = accept(sockfd, (struct sockaddr \*)&new\_addr, &addr\_size);

write\_file(new\_sock);

printf("[+]Data written in the file successfully.\n");

return 0;

}

**CLIENT.c**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <arpa/inet.h>

#define SIZE 1024

void send\_file(FILE \*fp, int sockfd)

{

int n;

char data[SIZE] = {0};

while (fgets(data, SIZE, fp) != NULL)

{

printf(" file data: %s", data);

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

{

data[i] = (data[i] + 3);

}

if (send(sockfd, data, sizeof(data), 0) == -1)

{

perror("[-]Error in sending file.");

exit(1);

}

bzero(data, SIZE);

}

}

int main()

{

char \*ip = "127.0.0.1";

int port = 8080;

int e;

int sockfd;

struct sockaddr\_in server\_addr;

FILE \*fp;

char \*filename = "send.txt";

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd < 0)

{

perror("[-]Error in socket");

exit(1);

}

printf("[+]Server socket created successfully.\n");

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = port;

server\_addr.sin\_addr.s\_addr = inet\_addr(ip);

e = connect(sockfd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr));

if (e == -1)

{

perror("[-]Error in socket");

exit(1);

}

printf("[+]Connected to Server.\n");

fp = fopen(filename, "r");

if (fp == NULL)

{

perror("[-]Error in reading file.");

exit(1);

}

send\_file(fp, sockfd);

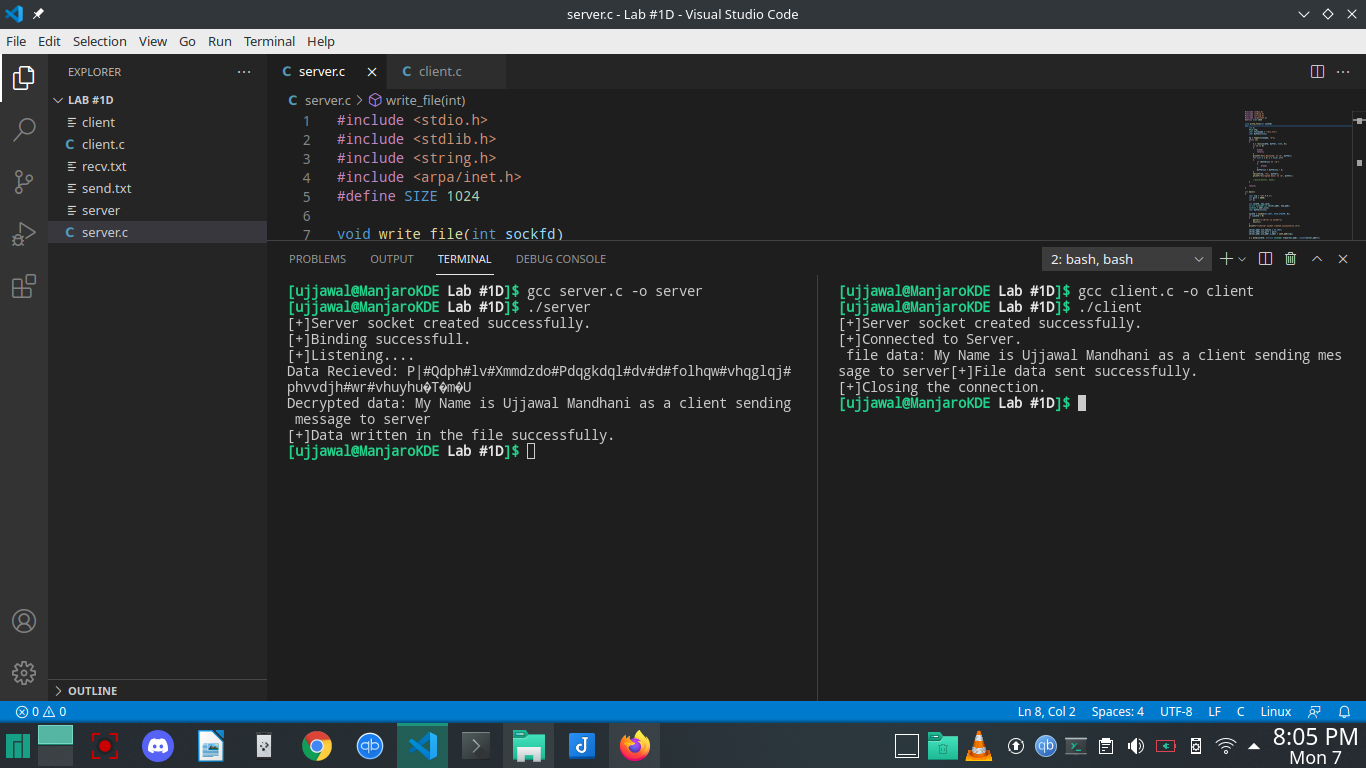
printf("[+]File data sent successfully.\n");

printf("[+]Closing the connection.\n");

close(sockfd);

return 0;

}



**Question 2:**Implement polyalphabetic cipher in C/C++ and compute the time taken to encrypt and

decrypt the following message “I AM JIIT STUDENT AND I AM PRACTISING INFORMATION

SECURITY”

#include <stdio.h>

#include <string.h>

void main()

{

char k[100], ct[100], pt[100];

int lenm, lenk, i, j;

char \*msg, \*key;

size\_t bufsize = 100;

printf("Enter Message : ");

getline(&msg, &bufsize, stdin);

printf("Enter Key : ");

getline(&key, &bufsize, stdin);

lenm = strlen(msg);

lenk = strlen(key);

for (i = 0; i < lenm; i++, j++)

{

if (j == lenk)

{

j = 0;

}

k[i] = key[j];

}

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

{

ct[i] = ((msg[i] + k[i]) % 26) + 'A';

}

ct[i] = '\0';

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

{

pt[i] = (((ct[i] - k[i]) + 26) % 26) + 'A';

}

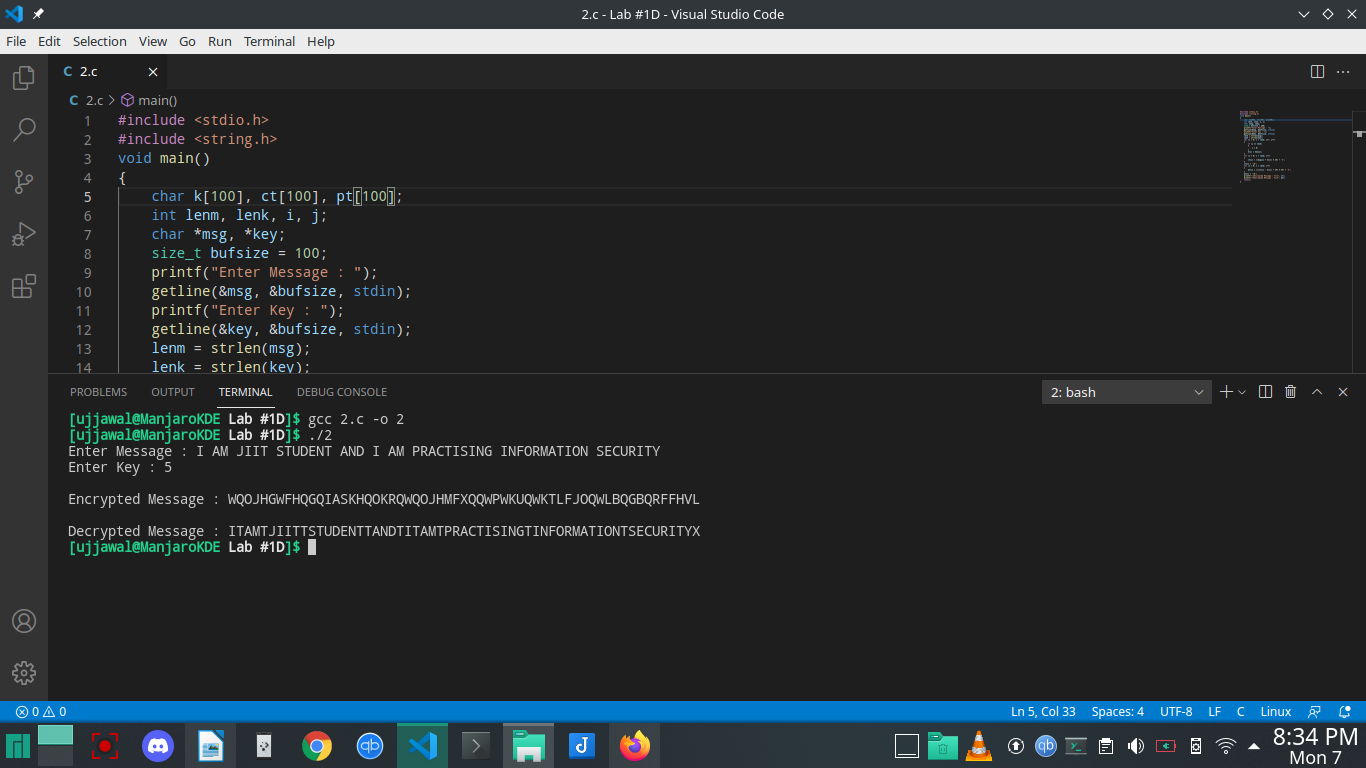
pt[i] = '\0';

printf("\nEncrypted Message : %s\n", ct);

printf("\nDecrypted Message : %s\n", pt);

return;

}

****