Name: Swaraj Shinde
Roll No : 164
Moodle ID: 23102070

Experiment: 08

Aim: Socket programming using TCP. Design TCP iterative Client and Server application to reverse the given input sentence.

Server.c:

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <sys/types.h>

#define MAXLINE 20
#define SERV_PORT 5777

int main(int argc, char *argv[]) {
    int i, j;
    ssize_t n;
    char line[MAXLINE], revline[MAXLINE];
    int listenfd, connfd, clilen;
    struct sockaddr_in servaddr, cliaddr;

    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    if (listenfd < 0) {
        perror("socket error");
        exit(1);
    }

    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(SERV_PORT);
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);

    if (bind(listenfd, (struct sockaddr*)&servaddr, sizeof(servaddr)) < 0) {
        perror("bind error");
        exit(1);
    }

    if (listen(listenfd, 1) < 0) {
        perror("listen error");
        exit(1);
    }

    printf("Server running... waiting for client\n");

    for (;;) {
```

```
        clilen = sizeof(cliaddr);
        connfd = accept(listenfd, (struct sockaddr*)&cliaddr, (socklen_t*)&clilen);
        if (connfd < 0) {
            perror("accept error");
            continue;
        }
        printf("Connected to client\n");

        while (1) {
            n = read(connfd, line, MAXLINE);
            if (n <= 0)
                break;

            line[n-1] = '\0';  // remove newline
            j = 0;
            for (i = n-2; i >= 0; i--)
                revline[j++] = line[i];
            revline[j] = '\0';

            write(connfd, revline, strlen(revline)+1);
        }
        close(connfd);
    }
}
```

client.c:

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <sys/types.h>

#define MAXLINE 20
#define SERV_PORT 5777

int main(int argc, char *argv[]) {
    char sendline[MAXLINE], revline[MAXLINE];
    int sockfd;
    struct sockaddr_in servaddr;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) {
        perror("socket error");
        exit(1);
    }

    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(SERV_PORT);
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
```

```
    if (connect(sockfd, (struct sockaddr*)&servaddr, sizeof(servaddr)) < 0) {
        perror("connect error");
        exit(1);
    }

    printf("Enter the data to be sent:\n");
    while (fgets(sendline, MAXLINE, stdin) != NULL) {
        write(sockfd, sendline, strlen(sendline));
        read(sockfd, revline, MAXLINE);
        printf("Reverse of the given sentence is: %s\n", revline);
    }

    close(sockfd);
    return 0;
}
```

FILE TRANSFER:
server.c:

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <fcntl.h>

#define PORT 5777
#define BUF_SIZE 1024

int main() {
    int listenfd, connfd;
    struct sockaddr_in servaddr, cliaddr;
    socklen_t clilen;
    char buffer[BUF_SIZE];
    ssize_t n;
    FILE *fp;

    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    if (listenfd < 0) {
        perror("socket error");
        exit(1);
    }

    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORT);
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);

    if (bind(listenfd, (struct sockaddr*)&servaddr, sizeof(servaddr)) < 0) {
        perror("bind error");
```

```
        exit(1);
    }

    if (listen(listenfd, 1) < 0) {
        perror("listen error");
        exit(1);
    }

    printf("Server ready. Waiting for client...\n");
    clilen = sizeof(cliaddr);
    connfd = accept(listenfd, (struct sockaddr*)&cliaddr, &clilen);
    if (connfd < 0) {
        perror("accept error");
        exit(1);
    }
    printf("Client connected. Receiving file...\n");

    fp = fopen("received_file.txt", "wb");
    if (fp == NULL) {
        perror("file open error");
        exit(1);
    }

    while ((n = read(connfd, buffer, BUF_SIZE)) > 0) {
        fwrite(buffer, 1, n, fp);
    }

    printf("File received successfully. Saved as 'received_file.txt'\n");
    fclose(fp);
    close(connfd);
    close(listenfd);

    return 0;
}
```

client.c:

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <fcntl.h>

#define PORT 5777
#define BUF_SIZE 1024

int main(int argc, char *argv[]) {
    int sockfd;
    struct sockaddr_in servaddr;
    char buffer[BUF_SIZE];
```

```
    ssize_t n;
    FILE *fp;

    if (argc != 2) {
        printf("Usage: %s <filename>\n", argv[0]);
        exit(1);
    }

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) {
        perror("socket error");
        exit(1);
    }

    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORT);
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY); // use 127.0.0.1 for local test

    if (connect(sockfd, (struct sockaddr*)&servaddr, sizeof(servaddr)) < 0) {
        perror("connect error");
        exit(1);
    }

    fp = fopen(argv[1], "rb");
    if (fp == NULL) {
        perror("file open error");
        exit(1);
    }

    printf("Sending file: %s\n", argv[1]);
    while ((n = fread(buffer, 1, BUF_SIZE, fp)) > 0) {
        write(sockfd, buffer, n);
    }

    printf("File sent successfully.\n");
    fclose(fp);
    close(sockfd);

    return 0;
}
```

OUTPUT:

```
apsit@apsit-HP-Pro-Tower-280-G9-E-PCI-Desktop-PC:~/Desktop/swaraj$ gcc server.c -o server
apsit@apsit-HP-Pro-Tower-280-G9-E-PCI-Desktop-PC:~/Desktop/swaraj$ ./server
Server running... waiting for client
Connected to client
```

```
apsit@apsit-HP-Pro-Tower-280-G9-E-PCI-Desktop-PC:~/Desktop/swaraj$ gcc client.c -o client
apsit@apsit-HP-Pro-Tower-280-G9-E-PCI-Desktop-PC:~/Desktop/swaraj$ ./client
Enter the data to be sent:
i love apsit
Reverse of the given sentence is: tispa evol i
```

```
apsit@apsit-HP-Pro-Tower-280-G9-E-PCI-Desktop-PC:~/Desktop/swaraj$ gcc server.c -o server
apsit@apsit-HP-Pro-Tower-280-G9-E-PCI-Desktop-PC:~/Desktop/swaraj$ ./server
Server ready. Waiting for client...
Client connected. Receiving file...
File received successfully. Saved as 'received_file.txt'
apsit@apsit-HP-Pro-Tower-280-G9-E-PCI-Desktop-PC:~/Desktop/swaraj$
```

```
apsit@apsit-HP-Pro-Tower-280-G9-E-PCI-Desktop-PC:~$ cd Desktop/swaraj
apsit@apsit-HP-Pro-Tower-280-G9-E-PCI-Desktop-PC:~/Desktop/swaraj$ gcc client.c -o client
apsit@apsit-HP-Pro-Tower-280-G9-E-PCI-Desktop-PC:~/Desktop/swaraj$ ./client file.txt
Sending file: file.txt
File sent successfully.
apsit@apsit-HP-Pro-Tower-280-G9-E-PCI-Desktop-PC:~/Desktop/swaraj$
```