

1. Using TCP/IP Socket programming, implement a program to transfer the contents of a requested file from server to the client using TCP/IP Sockets.

// client.c

```
#include <stdio.h>

#include <arpa/inet.h>

#include <unistd.h>

int main() {

    // Create a Socket: Use the socket() system call to create a new socket //
    // PF_INET: Specifies the protocol family (IPv4)//
    // SOCK_STREAM: Specifies the socket type (TCP for stream-oriented communication)//
    // 0: Specifies the protocol (usually 0 for default TCP)//

    int soc = socket(PF_INET, SOCK_STREAM, 0);

    if (soc < 0) return 1; // Exit if socket creation fails


    /*Connect to Server: Initialize a sockaddr_in structure with the server's IP address and port,
    then use connect() to establish a connection with the server*/

    struct sockaddr_in addr = {

        .sin_family = AF_INET,

        .sin_port = htons(7891),

        .sin_addr.s_addr = inet_addr("127.0.0.1")

    };

    if (connect(soc, (struct sockaddr *) &addr, sizeof(addr)) == 0) {

        printf("Connected to server\nEnter file name: ");

        char fname[50], buffer[1024];

        scanf("%s", fname);
```

*/*Communicate: Use send() and recv() (or write() and read()) to exchange data with the connected client */*

```
send(soc, fname, sizeof(fname), 0);
```

```
printf("Response:\n");
```

```
while (recv(soc, buffer, sizeof(buffer), 0) > 0)
```

```
    printf("%s", buffer);
```

```
    } else {
```

```
        perror("Connection failed");
```

```
    }
```

*/*Close Connections: Close the client socket using close() when communication is complete, and eventually close the server socket*/*

```
close(soc);
```

```
return 0;
```

```
}
```

//server.c

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

```
#include <arpa/inet.h>
```

```
#include <fcntl.h>
```

```
#include <unistd.h>
```

```
int main() {
```

```
// Create a Socket: Use the socket() system call to create a new socket //
```

```
// PF_INET: Specifies the protocol family (IPv4)//
```

```
// SOCK_STREAM: Specifies the socket type (TCP for stream-oriented communication)//
```

//0: Specifies the protocol (usually 0 for default TCP)//

```
int welcome = socket(PF_INET, SOCK_STREAM, 0);
```

/ Bind the Socket: Initialize a sockaddr_in structure with the server's IP address and port number, and then use bind() to associate the socket with this address */*

```
struct sockaddr_in addr = { .sin_family = AF_INET, .sin_port = htons(7891), .sin_addr.s_addr = inet_addr("127.0.0.1") };
```

```
bind(welcome, (struct sockaddr*)&addr, sizeof(addr));
```

```
printf("Server is online, waiting for connections...\n");
```

*/*Listen for Connections: Use listen() to put the server socket in a passive mode, waiting for incoming client connections*/*

```
listen(welcome, 5);
```

*/*Accept Connections: Use accept() to accept an incoming client connection*/*

```
int new_soc = accept(welcome, NULL, NULL);
```

```
char fname[50], buffer[1024];
```

*/*Communicate: Use send() and recv() (or write() and read()) to exchange data with the connected client */*

```
recv(new_soc, fname, sizeof(fname), 0);
```

```
printf("Request received for file: %s\n", fname);
```

```
int fd = open(fname, O_RDONLY);
```

```
if (fd < 0) {
```

```
    send(new_soc, "File not found\n", 15, 0);
```

```
    printf("File not found, notified client.\n");
```

```
} else {
```

```
    printf("File found, sending contents to client...\n");
```

```
    int n;
```

```
while ((n = read(fd, buffer, sizeof(buffer))) > 0)
    send(new_soc, buffer, n, 0);
}

/*Close Connections: Close the client socket using close() when communication is complete, and
eventually close the server socket*/

close(fd);

close(new_soc);

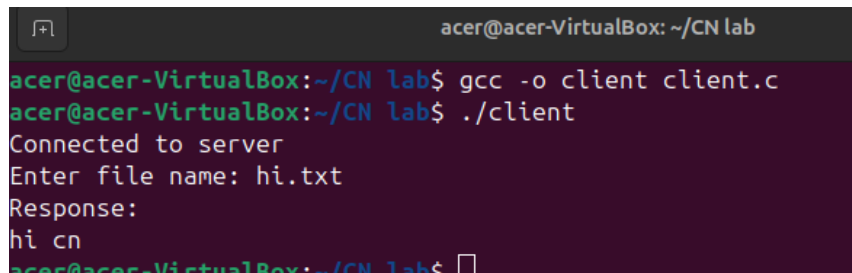
close(welcome);

printf("Request completed.\n");

return 0;
}
```

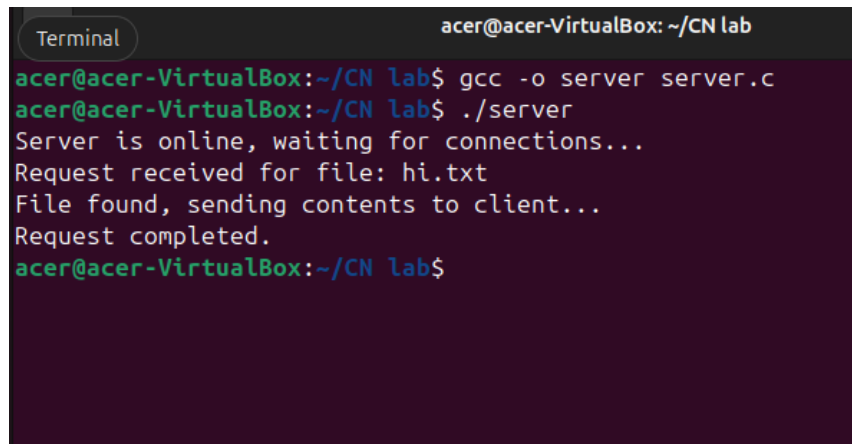
OUTPUT

Client. C



```
acer@acer-VirtualBox: ~/CN lab
acer@acer-VirtualBox:~/CN lab$ gcc -o client client.c
acer@acer-VirtualBox:~/CN lab$ ./client
Connected to server
Enter file name: hi.txt
Response:
hi cn
acer@acer-VirtualBox:~/CN lab$
```

SERVER.C



```
acer@acer-VirtualBox: ~/CN lab
acer@acer-VirtualBox:~/CN lab$ gcc -o server server.c
acer@acer-VirtualBox:~/CN lab$ ./server
Server is online, waiting for connections...
Request received for file: hi.txt
File found, sending contents to client...
Request completed.
acer@acer-VirtualBox:~/CN lab$
```