



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CSE2005 - Operating System
Project Review
Topic : Socket Programming
Group : 23

November 14, 2018

Members:

17BCE0336 (Purvesh Badmera)
17BCE0360 (Ziaul Umair)
17BCE2211 (Simriti Koul)
17BCE0945 (Shivam Sharma)
17BCE2334 (Rakhi Kumari Hathi)

Acknowledgement

We would like to express our deepest appreciation to all those who provided us the possibility to complete this report. A special gratitude we give to our project manager, Prof. KALYANARAMAN P whose contribution in stimulating suggestions and encouragement, helped us to coordinate our project especially in writing this report.

Furthermore we would also like to acknowledge with much appreciation the crucial role of the sta of VIT University, who gave the permission to use all required equipment and the necessary materials to complete the project “Socket Programming”. We have to appreciate the guidance given by other supervisor as well as the panels especially in our project presentation that has improved our presentation skills thanks to their comment and advices.

Abstract:

The concept of socket programming is used here to transfer the file from server to client. Here, the server creates the socket, binds them and make them listen to the requests from client, while client also creates the socket and binds them and then the server and client gets connected. The server code helps in making the files accessible for its clients. It accepts the names of the files required by the client and successfully delivers a copy of the file to the client if such a file exists. It uses the concepts of Semaphores and Mutex Locks to manage client requests. The client code is used to establish a stable network with the server using sockets to retrieve data.

Introduction:

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket listen on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server. While running the client code we provide the IP address of the server which we obtain by using the “ifconfig” in the terminal. This project helps you transfer files from one computer to many other computers. There are many applications of socket programming you can create a chat application for the communication between the clients and server and the file transfer application and many other.

Program

Server Code:

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

#define Port 3000

int connectsocket()
{

    int sock;
    struct sockaddr_in cliaddr, servaddr;
    sock = socket (AF_INET, SOCK_STREAM, 0);
    if ( sock<0)
    {
        perror("Socket_creation_problem");
        exit(1);
    }

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(Port);

    bind (sock, (struct sockaddr_in *) &servaddr, sizeof(servaddr));

    listen (sock, 3);
    printf("%s\n", "Server_is_up.");
    return sock;
}

void* body(int *arg)
{
    struct sockaddr_in cliaddr;
    int addrlen;
    int i,l;
    int c = (int) arg;
    int b;
    while (1)
```

```

{
b = accept(c, &cliaddr, &addrlen);
if(b<0)
{
perror("Error");
exit(1);
}
printf("\n\nConnected\n");
file_manip(b);
close(b);
}
}

void file_manip(int d)
{
char buffer_rec[1000], buffer_send[1000];
char file_buffer[1000], f_buffer[1000];
int n;
FILE *f;
printf("\n\nRequesting_client_for_file_name:\n\n");
sprintf(buffer_send, "Enter_file_name: ");
send(d, buffer_send, strlen(buffer_send), 0);
printf("\n\nReceiving\n\n");
n=recv(d, buffer_rec, 1000, 0);
buffer_rec[n]='\0';
printf("%s\n", buffer_rec);
fflush(stdout);
printf("Checking_FILE:%s_exists_or_not\n", buffer_rec);
if((f = fopen(buffer_rec, "r"))==NULL)
{
sprintf(buffer_send, "File_not_found");
exit(0);
}
else
{
printf("\nFile_found\n");
sprintf(buffer_send, "File_found\n");
send(d, buffer_send, strlen(buffer_send), 0);
}
printf("Sending_file_to_client\n");
while(!feof(f))
{
fgets(f_buffer, 1000, f);
if (feof(f))
break;
strcat(file_buffer, f_buffer);

```

```

}
fclose(f);
send(d, file_buffer , strlen( file_buffer ), 0);
close(d);
printf("Connection_closed.\n");
}

int main (int argc , char **argv)
{
int c;
c=connectsocket();

pthread_t t;
struct sockaddr_in clieaddr;
socklen_t addrlen;
addrlen = sizeof(clieaddr);
printf ("%s\n", "thread_created_for_client_requests");
pthread_create(&t, 0, body, (void *)c);
pause();
}

```

Client Code:

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

#define Port 3000

int main(int argc, char **argv)
{
    int sock, a;
    struct sockaddr_in servaddr;
    char send_line[1000], receive_line[1000];

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

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr(argv[1]);
    servaddr.sin_port = htons(Port);

    a = connect(sock, (struct sockaddr *) &servaddr, sizeof(servaddr));
    if (a < 0)
    {
        perror("Server_is_not_getting_connected");
        exit(1);
    }
    else
        printf("\nConnected\n");

    int n;
    char buffer_rec[1000];
    char buffer_send[1000];
    char *filename;
    char file_buffer[1000];
    FILE *f;
```

```

n = recv(sock , buffer_rec , sizeof(buffer_rec) , 0);
buffer_rec [n]='\0';
printf("%s",buffer_rec);

printf("\n\nRequired_file_name:\n\n");
scanf("%s",buffer_send);
send(sock ,buffer_send ,strlen(buffer_send),0);

printf("\n\nreceiving_the_string\n"File_found_or_not"\n\n");
n = recv(sock , buffer_rec , sizeof(buffer_rec) , 0);
buffer_rec [n]='\0';
printf("%s",buffer_rec);
fflush(stdout);

printf("\n\nReceiving\n\n");
n=recv(sock , file_buffer , sizeof(file_buffer) , 0);
file_buffer [n]='\0';
fflush(stdout);

f = fopen("received_file.txt","w");
fputs(file_buffer ,f);
fclose(f);
close(sock);

if (argc !=2)
{
perror("Error");
exit(1);
}
return 0;
}

```


Output

Screenshots:

```
In file included from s.c:4:8:
/usr/include/x86_64-linux-gnu/sys/socket.h:112:12: note: expected 'const struct sockaddr *' but argument is of type 'struct sockaddr_inr *'
extern int bind(int __fd, __CONST_SOCKADDR_ARG __addr, socklen_t __len)
          ^
s.c: In function 'body':
s.c:40:9: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
    int c = (int) arg;
          ^
s.c:44:15: warning: passing argument 2 of 'accept' from incompatible pointer type [-Wincompatible-pointer-types]
    b = accept(c, &cliaddr, &addrlen);
              ^
In file included from s.c:4:8:
/usr/include/x86_64-linux-gnu/sys/socket.h:232:12: note: expected 'struct sockaddr * restrict' but argument is of type 'struct sockaddr_in *'
extern int accept(int __fd, __SOCKADDR_ARG __addr,
          ^
s.c:51:11: warning: implicit declaration of function 'file_manip'; did you mean 'fileno'? [-Wimplicit-function-declaration]
    file_manip(b);
    ^
~~~~~
fileno
s.c: At top level:
s.c:58:8: warning: conflicting types for 'file_manip'
void file_manip(int d)
     ^
~~~~~
s.c:51:11: note: previous implicit declaration of 'file_manip' was here
    file_manip(b);
    ^
~~~~~
s.c: In function 'main':
s.c:106:11: warning: implicit declaration of function 'pthread_create' [-Wimplicit-function-declaration]
    pthread_create(&t,0,body,(void *)c);
    ^
~~~~~
s.c:106:26: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
    pthread_create(&t,0,body,(void *)c);
                         ^
purvesh@Purvesh-Inspiron-5567:~/Desktop$ ./s
Server is up.
thread created for client requests

Connected

Requesting client for file name:

Receiving

file.txt
Checking FILE:file.txt exists or not

File found
Sending file to client
Connection closed.
```

Figure 1: Server output.

```
purvesh@Purvesh-Inspiron-5567:~/Desktop$ cd Desktop
purvesh@Purvesh-Inspiron-5567:~/Desktop$ gcc -o c c.c
c.c: In function 'main':
c.c:66:11: warning: implicit declaration of function 'close'; did you mean 'pclose'? [-Wimplicit-function-declaration]
    close(sock);
    ^
~~~~~
close
purvesh@Purvesh-Inspiron-5567:~/Desktop$ ./c 127.0.0.1

Connected
Enter file name:

Required file name:

file.txt

receiving the string "File found or not"

File found

Receiving
purvesh@Purvesh-Inspiron-5567:~/Desktop$
```

Figure 2: Client output.

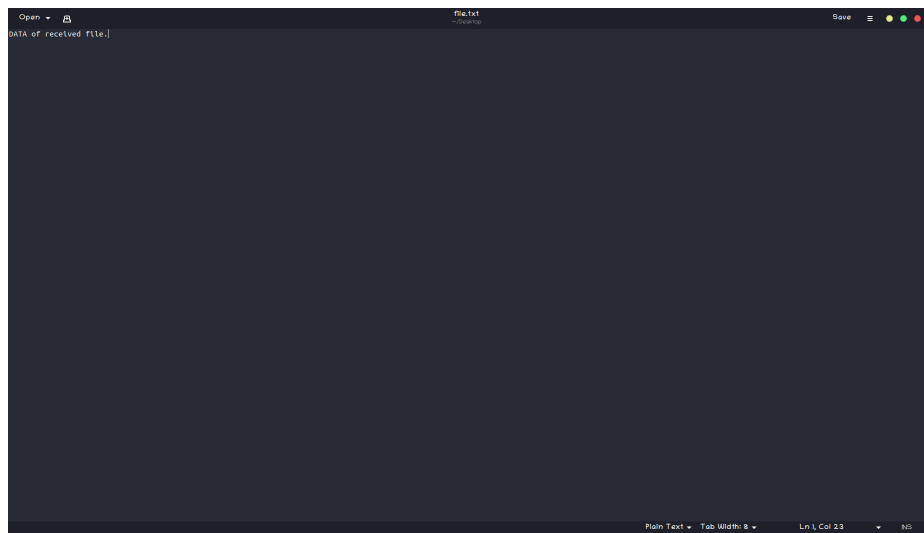


Figure 3: File.

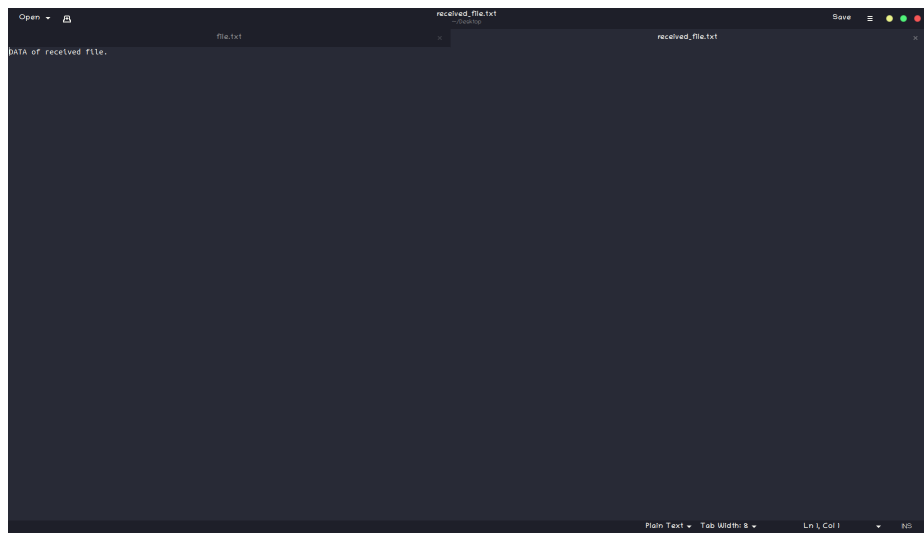


Figure 4: Received File.

References:

- <https://www.geeksforgeeks.org/socket-programming-cc/>
- <https://stackoverflow.com/questions/21191749/file-sending-from-multiple-client-to-single-server-in-java-socket-programming>
- <http://www.cs.unc.edu/~dewan/242/s07/notes/ipc/node27.html>
- Book - Advanced Concepts in Operating Systems - Mukesh Singhal, Niranjan G.Shivaratri

The End