Umang Srivastava
2019101090

# OSN Assignment 6
# Socket Programming

## Overview

In this assignment, we are asked to implement socket programming. The client sends requests to download files which are in server directory. Error cases for missing files or any scenario are handled.

## Directory Structure

- /server/
  - server.c
  - makefile
  - file.txt (test file)
  - test (test file)
- /client/
  - client.c
  - makefile
- README.pdf

## Execution

1. cd server
2. make
3. ./server
4. cd client
5. make
6. ./client

## Code Description for client

The code for the client side starts with establishing the connection to the socket and checking for any errors.

```
struct sockaddr_in address;
int sock = 0, valread;
struct sockaddr_in serv_addr;
char buffer[1024] = {0};
if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
{
    printf("\n Socket creation error \n");
    return -1;
}

memset(&serv_addr, '0', sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(PORT);

if(inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr)<=0)
{
    printf("\nInvalid address/ Address not supported \n");
    return -1;
}

if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
{
    printf("\nConnection Failed \n");
    return -1;
}
```

Now , the connection is established and we send request to server for its directory address.

```
printf("Connected to Server...\n\n");
char server_wd[1024];
valread = read( sock , server_wd, 1024);
```

Now we go into the main loop of the program which runs till we get exit instruction.

```
if(cnt == 1 && strcmp(arr[0],"exit") == 0)
{
    send(sock, "-1", sizeof("-1"), 0);
    printf("GoodBye..!!\n\n");
    break;
}
```

We take input from user and store it in character array inp. Next, we separate the words and store all words in 2D character array arr.

```
printf("client> ");
fgets(inp,10000,stdin);
lli l = strlen(inp);
l--;
lli cnt=0,j=0;
for(lli i=0;i<=l;i++)
{
    if(inp[i]==' ' || i==l)
    {
        cnt++;
        j=0;
    }
    else
    {
        arr[cnt][j]=inp[i];
        j++;
    }
}
```

Next we each filename, we check for any sort of error, if encountered, appropriate error message is displayed.

Next we open the file, if present then we proceed forward else error message is displayed.

```
int fl = open(source, O_RDONLY);
if(fl < 0)
{
        printf("File Not Found: %s\n\n",source);
        continue;
}
```

Then we proceed to the writing part of the file where we receive data packets from socket and write them to the destination file. The progress is displayed in percentage.

```
void receivefile(int sockfd, int fp, lli deno)
{
    ssize_t n=1;
    char buff[N];
    memset(buff,0,N);
    lli cnt2=1;
    while (n > 0)
    {
        n = recv(sockfd, buff, N, 0);
        if(n<=0 || strcmp(buff,"-1")==0)
            break;
        if(strcmp(buff,"-1")==0)
            break;
        double ans = cnt2*100.00 / deno;
        if(ans>100)
            ans=100;
        printf("\r%0.2f %% downloaded....",ans);
        cnt2++;
        if (n == -1)
        {
            perror("Receive File Error");
            exit(1);
        }

        if (write(fp, buff, n) != n)
        {
            perror("Write File Error");
            exit(1);
        }
        memset(buff, 0, N);
    }
}
```

After writing the file, we close the file descriptors and print the “Downloaded” message and proceed to download the next file.

```
printf("\n%s downloaded\n\n",arr[j]);
close(fl2);
close(file);
```

## Code Description for server

The code for the server side starts with establishing the connection using socket and checking for any errors.

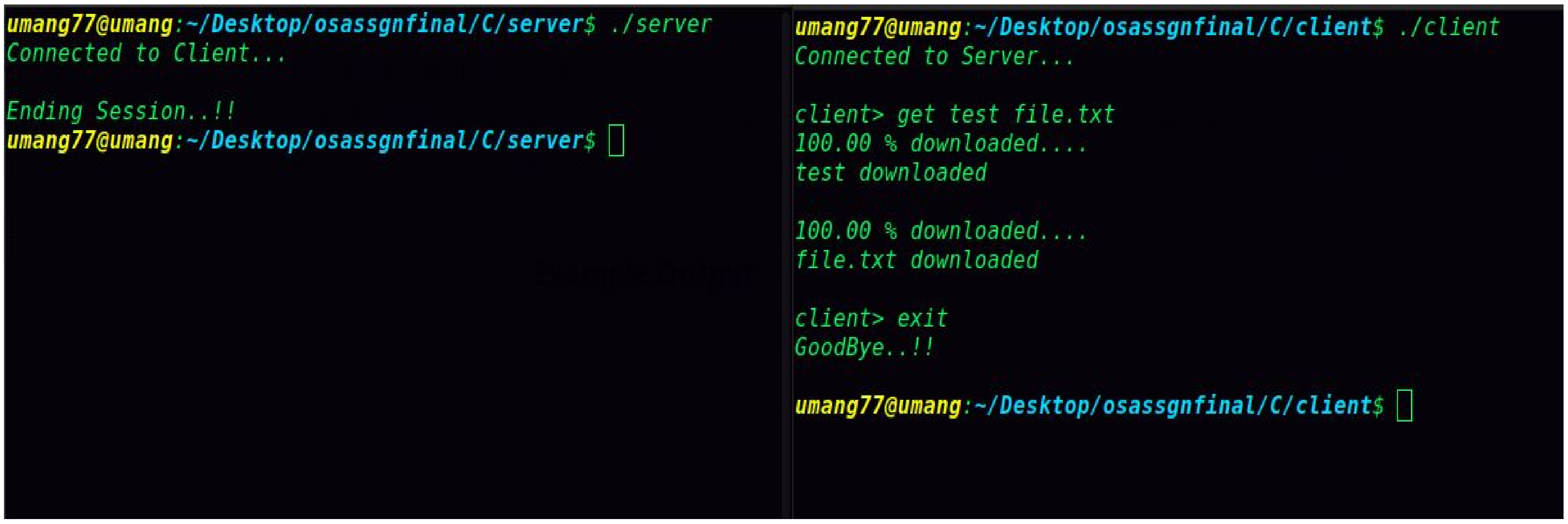
```
if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
{
        perror("socket failed");
        exit(EXIT_FAILURE);
}
if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,&opt,
sizeof(opt)))
{
        perror("setsockopt");
        exit(EXIT_FAILURE);
}
address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons( PORT );
if (bind(server_fd, (struct sockaddr *)&address,sizeof(address))<0)
{
        perror("bind failed");
        exit(EXIT_FAILURE);
}
if (listen(server_fd, 3) < 0)
{
        perror("listen");
        exit(EXIT_FAILURE);
}
if ((new_socket = accept(server_fd, (struct sockaddr *) &address,
(socklen_t*) &addrlen))<0)
{
        perror("accept");
        exit(EXIT_FAILURE);
}
```

Now , the connection is established and we send server's directory address to the client.
Then for the file request received, we open that file, read the data and send it through socket.Upon receiving the exit request from client side, we break the loop

and exit the program.

```c
void sendfile(int fp, int sockfd)
{
    lli n;
    char sendline[N];
    memset(sendline, 0, N);
    while ((n = read(fp, sendline, N)) > 0)
    {
        if (n == -1)
        {
            perror("Read File Error");
            exit(1);
        }
        if (send(sockfd, sendline, n, 0) == -1)
        {
            perror("Can't send file");
            exit(1);
        }
        memset(sendline, 0, N);
    }
    sleep(1);
    send(sockfd, "-1", sizeof("-1"), 0);
}
```

## Example Output



```
umang77@umang:~/Desktop/osassgnfinal/C/server$ ./server
Connected to Client...

Ending Session..!!
umang77@umang:~/Desktop/osassgnfinal/C/server$ 
```

```
umang77@umang:~/Desktop/osassgnfinal/C/client$ ./client
Connected to Server...

client> get test file.txt
100.00 % downloaded....
test downloaded

100.00 % downloaded....
file.txt downloaded

client> exit
GoodBye..!!

umang77@umang:~/Desktop/osassgnfinal/C/client$ 
```