Manish J Bangera

4nm18cs090

B section

Simple Chat Application

Problem : One Server and single client text chatting app .

Here server and client are created using unix based operating system socket APIs.
We are communicating through internet domain and socket streams ie through
TCP (Transmission Control Protocol).

Server : In serverSetup function socket gets created , then a child process is created to read data from socket and display to the terminal using function (readFromSocket) for receiving message from client. In parent process input is taken from the terminal and written to the socket using function (writeToSocket) for sending message to client. Here child process is created because to send and receive message virtually simultaneously.

Client: In clientSetup function socket gets created , then a child process is created to read data from socket and display to the terminal using function (readFromSocket) for receiving message from server. In parent process input is taken from the terminal and written to the socket using function (writeToSocket) for sending message to server.

In the End when application gets closed through ctrl + c .

How to get started ?

First compile the server code,
cc -o server tcp_server.c

then compile client code
cc -o client tcp_client.c

then execute the server with port number as agruement ,for example 3000
./server 3000

then execute the client with hostname and port number as agruement , port number should be what the server is listening to , for example : hostname : localhost and port number 3000,

./client localhost 3000

Then start Chating.

Server code:

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

int gsockfd,gnewsockfd;
void error(char *msg)
{
perror(msg);
exit(1);
}
/* A simple server in the internet domain using TCP
The port number is passed as an argument */
void serverSetup(int *sockfd,int *newsockfd,int *portno,int
*clilen,char *argv){
struct sockaddr_in serv_addr, cli_addr;
*sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0)
error("ERROR opening socket");
bzero((char *) &serv_addr, sizeof(serv_addr));
*portno = atoi(argv);
serv_addr.sin_family = AF_INET;

serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = htons(*portno);
if (bind(*sockfd, (struct sockaddr *) &serv_addr,
sizeof(serv_addr)) < 0)
```

```
error("ERROR on binding");
listen(*sockfd,5);
*clilen = sizeof(cli_addr);
*newsockfd = accept(*sockfd, (struct sockaddr *) &cli_addr,
clilen);
if (newsockfd < 0)
error("ERROR on accept");
printf("listing to PORT: %d \n",*portno);
gsockfd = *sockfd;
gnewsockfd = *newsockfd;
}

void writeToSocket(char *input , int newsockfd){
ssize_t n = -1;
bzero(input,256);
fgets(input, 256, stdin);
n = write(newsockfd,(char *)input,strlen(input));
if(n == -1){
error("erro while writting to socket");
}
bzero(input,256);
}

void readFromSocket(char *output,int newsockfd){
ssize_t n = -1;
bzero(output,256);
while(n = read(newsockfd,(char*)output,256) > 0){
printf("client-> %s",output);
bzero(output,256);
}
if(n == -1){
error("error while reading from socket");
}
}

void handleParentSigInt(int sig){
close(gsockfd);
close(gnewsockfd);
printf("closing the application sucessfully\n");
exit(0);
}

void handleChildSigInt(int sig){
close(gsockfd);
close(gnewsockfd);
```

```
exit(0);
}

int main(int argc, char *argv[])
{
int sockfd, newsockfd, portno, clilen;
if (argc < 2) {
fprintf(stderr,"ERROR, no port provided\n");
exit(1);
}
serverSetup(&sockfd,&newsockfd,&portno,&clilen,argv[1]);
signal(SIGINT,handleParentSigInt);
pid_t pid = fork();
if(pid < 0){
printf("error couldn't fork \n");
exit(1);
}else if(pid == 0){
char output[257];
signal(SIGINT,handleChildSigInt);
while(1){
readFromSocket(output,newsockfd);
}

}else{
char input[256];
while(1){
writeToSocket(input,newsockfd);
}
}
return 0;
}
```

Client Code:

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

```c
#include <fcntl.h>
#include <sys/stat.h>
#include<signal.h>
int gsockfd,gcpid;
void error(char *msg)
{
perror(msg);
exit(0);
}

void handleParentSigInt(int sig){
close(gsockfd);
printf("closing the application sucessfully\n");
exit(0);
}

void handleChildSigInt(int sig){
close(gsockfd);
exit(0);
}

void clientSetup(int *sockfd,char *host,char *port){
int portno;
struct sockaddr_in serv_addr;
struct hostent *server;

portno = atoi(port);
*sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (*sockfd < 0)
error("ERROR opening socket");
server = gethostbyname(host);
if (server == NULL) {
fprintf(stderr,"ERROR, no such host\n");
exit(0);
}
bzero((char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
bcopy((char *)server->h_addr,
(char *)&serv_addr.sin_addr.s_addr,
server->h_length);
serv_addr.sin_port = htons(portno);
if (connect(*sockfd,(struct sockaddr
*)&serv_addr,sizeof(serv_addr)) < 0)
error("ERROR connecting");
printf("connection established\n");
```

```
gsockfd = *sockfd;
}

void writeToSocket(char *input , int sockfd){
ssize_t n = -1;
bzero(input,256);
fgets(input, 256, stdin);
n = write(sockfd,(char *)input,strlen(input));
if(n == -1){
error("erro while writting to socket");
}
bzero(input,256);
}

void readFromSocket(char *output,int sockfd){
ssize_t n = -1;
bzero(output,256);
while(n = read(sockfd,(char*)output,256) > 0){
printf("server-> %s",output);
bzero(output,256);
}
if(n == -1){
error("error while reading from socket");
}
}
int main(int argc, char *argv[])
{
int sockfd, portno;
if (argc < 3) {
fprintf(stderr,"usage %s hostname port\n", argv[0]);
exit(0);
}
clientSetup(&sockfd,argv[1],argv[2]);
signal(SIGINT,handleParentSigInt);
pid_t pid = fork();
if(pid < 0){
printf("error couldn't fork \n");
exit(1);
}else if(pid == 0){
signal(SIGINT,handleChildSigInt);
char output[256];
while(1){
readFromSocket(output,sockfd);
}
}else{
```

```
char input[256];
while(1){
writeToSocket(input,sockfd);
}
}
return 0;
}
```

Output: