Server-side Program:EchoServer.c

/\*Required Headers\*/

#include <sys/types.h>

#include <sys/socket.h>

#include <netdb.h>

#include <stdio.h>

#include<string.h>

int main()

{

char str[100];

int listen\_fd, comm\_fd;

struct sockaddr\_in servaddr;

listen\_fd = socket(AF\_INET, SOCK\_STREAM, 0);

bzero( &servaddr, sizeof(servaddr));

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

servaddr.sin\_port = htons(22000);

bind(listen\_fd, (struct sockaddr \*) &servaddr, sizeof(servaddr));

listen(listen\_fd, 10);

while(1)

{

comm\_fd = accept(listen\_fd, (struct sockaddr\*) NULL, NULL);

bzero( str, 100);

recv(comm\_fd,str,100,0);

printf("Echoing back - %s",str);

send(comm\_fd,str,strlen(str),0);

close(comm\_fd);

}

}

Client Side Program: EchoClient.c

#include <sys/types.h>

#include <sys/socket.h>

#include <netdb.h>

#include <stdio.h>

#include<string.h>

int main(int argc,char \*\*argv)

{

int sockfd,n;

char sendline[100];

char recvline[100];

struct sockaddr\_in servaddr;

sockfd=socket(AF\_INET,SOCK\_STREAM,0);

bzero(&servaddr,sizeof servaddr);

servaddr.sin\_family=AF\_INET;

servaddr.sin\_port=htons(22000);

servaddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

//inet\_pton(AF\_INET,"127.0.0.1",&(servaddr.sin\_addr));

connect(sockfd,(struct sockaddr \*)&servaddr,sizeof(servaddr));

while(1)

{

bzero( sendline, 100);

bzero( recvline, 100);

fgets(sendline,100,stdin); /\*stdin = 0 , for standard input \*/

send(sockfd,sendline,strlen(sendline),0);

recv(sockfd,recvline,100,0);

printf("%s",recvline);

}

}

-----X----

LAB 3

#include <stdio.h>

#include <signal.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <netdb.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

// Function designed for chat between client and server.

void func(int sockfd, short cport)

{

char buff[MAX];

int n;

// infinite loop for chat

for (;;) {

bzero(buff, MAX); // set buff with null values

// read the message from client and copy it in buffer

read(sockfd, buff, sizeof(buff));

//recvfrom

// print buffer which contains the client contents

printf("From client(%d): %s\nTo client(%d) : ",cport, buff,cport);

bzero(buff, MAX);

n = 0;

// copy server message in the buffer

while ((buff[n++] = getchar()) != '\n')

;

// and send that buffer to client

write(sockfd, buff, sizeof(buff));

//sendto

// if msg contains "Exit" then server exit and chat ended.

if (strncmp("exit", buff, 4) == 0) {

printf("Server Exit...\n");

break;

}

}

}

// Driver function

int main()

{

int sockfd, connfd, len;

struct sockaddr\_in servaddr, cli;

pid\_t pid;

//void sig\_chld(int);

// socket create and verification

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd == -1) {

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

bzero(&servaddr, sizeof(servaddr));

// assign IP, PORT

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

servaddr.sin\_port = htons(PORT);

// Binding newly created socket to given IP and verification

if ((bind(sockfd, (SA\*)&servaddr, sizeof(servaddr))) != 0) {

printf("socket bind failed...\n");

exit(0);

}

else

printf("Socket successfully binded..\n");

// Now server is ready to listen and verification

if ((listen(sockfd, 5)) != 0) {

printf("Listen failed...\n");

exit(0);

}

else

printf("Server listening..\n");

len = sizeof(cli);

while(1){

// Accept the data packet from client and verification

connfd = accept(sockfd, (SA\*)&cli, &len);

if (connfd < 0) {

printf("server acccept failed...\n");

exit(0);

}

else{

printf("server acccept the client...\n");

short c\_port= ntohs(cli.sin\_port);

// Function for chatting between client and server

func(connfd,c\_port);

}//end else

}//end while(1)

close(sockfd);// After chatting close the socket

}

--x---

#include<sys/socket.h>

#include<netdb.h>

#include<string.h>

#include<stdlib.h>

#include<stdio.h>

int main()

{

char buff[80];

int sockfd,len,n;

struct sockaddr\_in servaddr;

sockfd=socket(AF\_INET,SOCK\_DGRAM,0);

if(sockfd==-1)

{

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

bzero(&servaddr,sizeof(len));

servaddr.sin\_family=AF\_INET;

servaddr.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

servaddr.sin\_port=htons(43454);

len=sizeof(servaddr);

printf("\nEnter string : ");

n=0;

while((buff[n++]=getchar())!='\n')

;

sendto(sockfd,buff,sizeof(buff),0,(struct sockaddr \*)&servaddr,len);

bzero(buff,sizeof(buff));

recvfrom(sockfd,buff,sizeof(buff),0,(struct sockaddr \*)&servaddr,&len);

printf("From Server : %s\n",buff);

close(sockfd);

}

-----X----

LAB 4

cc

#include <netdb.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

void func(int sockfd)

{

char buff[MAX];

int n;

for (;;) {

bzero(buff, sizeof(buff));

printf("Enter the string : ");

n = 0;

while ((buff[n++] = getchar()) != '\n')

;

write(sockfd, buff, sizeof(buff));//send the message to server

bzero(buff, sizeof(buff)); //nullifies the buffer content

read(sockfd, buff, sizeof(buff)); //reading the message recived from server

printf("From Server : %s", buff);

if ((strncmp(buff, "exit", 4)) == 0) {

printf("Client Exit...\n");

close(sockfd);

break;

}

}

}

int main()

{

int sockfd, connfd;

struct sockaddr\_in servaddr, cli;

// socket create and varification

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if (sockfd == -1) {

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

bzero(&servaddr, sizeof(servaddr));

// assign IP, PORT

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

servaddr.sin\_port = htons(PORT);

// connect the client socket to server socket

if (connect(sockfd, (SA\*)&servaddr, sizeof(servaddr)) != 0) {

printf("connection with the server failed...\n");

exit(0);

}

else

printf("connected to the server..\n");

// function for chat

func(sockfd);

// close the socket

close(sockfd);

}