**Program\_05**

**chatclient.c**

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<sys/stat.h>

#include<unistd.h>

#include<stdlib.h>

#include<stdio.h>

#include<fcntl.h>

#include<arpa/inet.h>

void str\_cli(FILE \*fp, int sockfd)

{

int bufsize=2048,cont;

char \*buffer=malloc(bufsize);

while(fgets(buffer,bufsize,fp)!=NULL)

{

send(sockfd, buffer, sizeof(buffer),0);

if((cont=recv(sockfd,buffer,bufsize,0))>0)

{

fputs(buffer,stdout);

}

}

printf("\nEOF\n");

}

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

{

int create\_socket;

struct sockaddr\_in address;

if((create\_socket=socket(AF\_INET,SOCK\_STREAM,0))>0)

printf("The socket was created.\n");

address.sin\_family=AF\_INET;

address.sin\_port=htons(18000);

inet\_pton(AF\_INET, argv[1], &address.sin\_addr);

if(connect(create\_socket,(struct sockaddr\*)&address, sizeof(address))==0)

printf("The connection was accepted with the server %s...\n",argv[1]);

else

printf("Error in connect()\n");

str\_cli(stdin, create\_socket);

return close(create\_socket);

}

**chatclient2.c**

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<sys/stat.h>

#include<unistd.h>

#include<stdlib.h>

#include<stdio.h>

#include<fcntl.h>

#include<arpa/inet.h>

void str\_cli(FILE \*fp, int sockfd)

{

int bufsize=2048,cont;

char \*buffer=malloc(bufsize);

while(fgets(buffer,bufsize,fp)!=NULL)

{

send(sockfd, buffer, sizeof(buffer),0);

if((cont=recv(sockfd,buffer,bufsize,0))>0)

{

fputs(buffer,stdout);

}

}

printf("\nEOF\n");

}

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

{

int create\_socket;

struct sockaddr\_in address;

if((create\_socket=socket(AF\_INET,SOCK\_STREAM,0))>0)

printf("The socket was created.\n");

address.sin\_family=AF\_INET;

address.sin\_port=htons(18000);

inet\_pton(AF\_INET, argv[1], &address.sin\_addr);

if(connect(create\_socket,(struct sockaddr\*)&address, sizeof(address))==0)

printf("The connection was accepted with the server %s...\n",argv[1]);

else

printf("Error in connect()\n");

str\_cli(stdin, create\_socket);

return close(create\_socket);

}

**chatserver.c**

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <sys/stat.h>

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

#include <fcntl.h>

#include <arpa/inet.h>

void str\_echo(int connfd, FILE \*fp) {

int n;

int bufsize =2048;

char \*buffer = malloc(bufsize);

again:

while ((n = recv(connfd, buffer, bufsize, 0)) > 0) {

fputs(buffer,stdout);

if(fgets(buffer,bufsize,fp)!=NULL)

send(connfd, buffer, n, 0);

//printf("Port: %d\n", port);

//system(buffer);

}

if (n < 0)

goto again;

}

int main() {

int cont, listenfd, connfd, addrlen, addrlen2, fd, pid, addrlen3;

struct sockaddr\_in address, cli\_address;

if ((listenfd = socket(AF\_INET, SOCK\_STREAM, 0)) > 0) {

printf("The socket was created\n");

}

address.sin\_family = AF\_INET;

address.sin\_addr.s\_addr = INADDR\_ANY;

address.sin\_port = htons(18000);

printf("The address before bind %s...\n", inet\_ntoa(address.sin\_addr));

if (bind(listenfd, (struct sockaddr \*)& address, sizeof(address)) == 0) {

printf("Binding Socket\n");

}

printf("The address after bind %s...\n", inet\_ntoa(address.sin\_addr));

listen(listenfd, 3);

printf("Server is listening\n");

getsockname(listenfd, (struct sockaddr \*)&address, &addrlen3);

printf("The server's local address %s... and port %d\n", inet\_ntoa(address.sin\_addr), htons(address.sin\_port));

while (1) {

addrlen = sizeof(struct sockaddr\_in);

connfd = accept(listenfd, (struct sockaddr\*)& cli\_address, &addrlen);

addrlen2 = sizeof(struct sockaddr\_in);

int i = getpeername(connfd, (struct sockaddr \*)& cli\_address, &addrlen);

printf("The Client %s is connected on port %d\n", inet\_ntoa(cli\_address.sin\_addr), htons(cli\_address.sin\_port));

if ((pid = fork()) == 0) {

printf("Inside child\n");

close(listenfd);

//str\_echo(connfd, htons(cli\_address.sin\_port));

str\_echo(connfd,stdin);

exit(0);

}

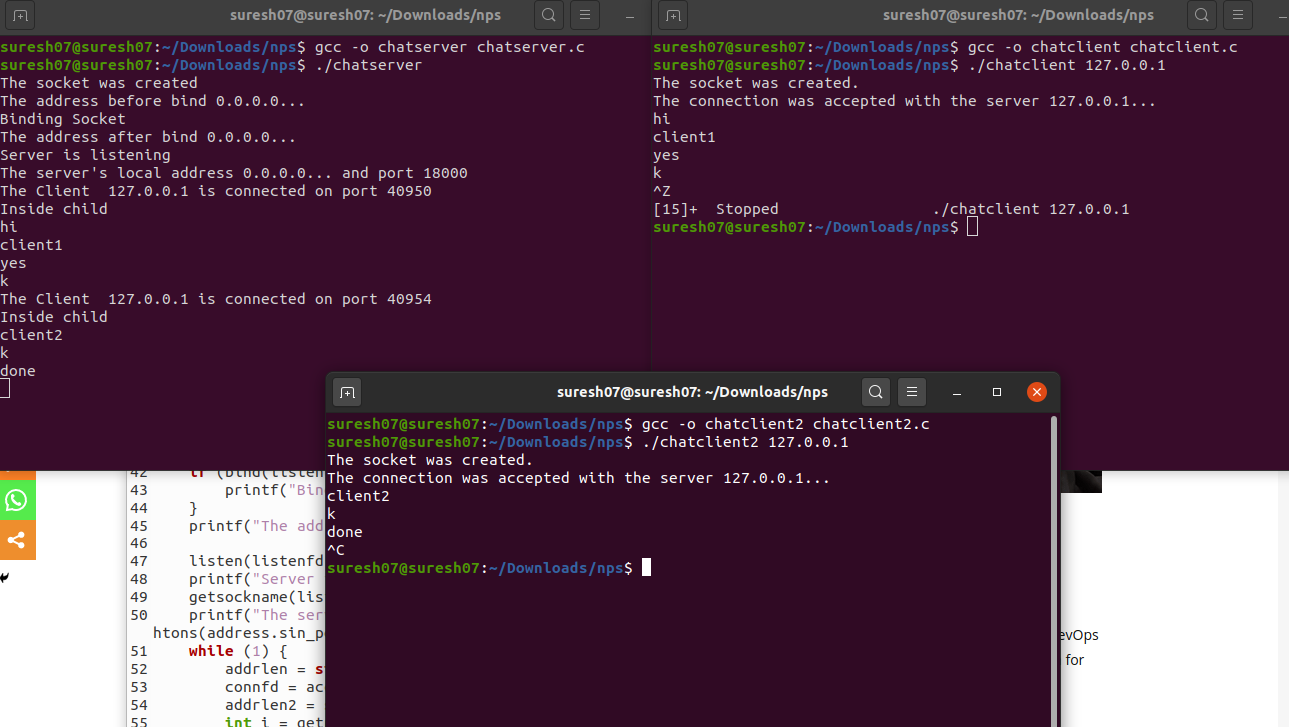
close(connfd);

}

return 0 ;

}

**OUTPUT**

****