MINOR ASSIGNMENT-5

UNIX Network Programming (CSE 4042)

1.

SOLN:-

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<stdlib.h>

#include<string.h>

#include<signal.h>

#define PORT 12262

#define MAXLINE 200

void dg\_echo(int sockfd,struct sockaddr\* pcliaddr,socklen\_t client){

int n;

socklen\_t len;

char buffer[MAXLINE];

memset(buffer,'\0', sizeof(buffer));

for(;;)

{

len = client;

n=recvfrom(sockfd,buffer,MAXLINE,0, pcliaddr,&len);

printf("message from client %s\n", buffer);

sendto(sockfd,buffer,n,0,pcliaddr,len);

}

}

int main()

{

int lisnfd, br;

socklen\_t clilen,len;

struct sockaddr\_in servaddr, cliaddr;

len=sizeof(servaddr);

servaddr.sin\_family=AF\_INET;

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

servaddr.sin\_port=htons(PORT);

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

if(lisnfd<0)

{

fprintf(stderr,"create error in socket\n");

return 1;

}

br=bind(lisnfd,(struct sockaddr \*)&servaddr,sizeof(servaddr));

if(br==0)

{

printf("Bind success: with return value=%d\n", br);

}

else{

printf("bind unsuccess:with return value=%d\n", br);

printf("Retry........" );

exit(2);

}

dg\_echo(lisnfd,(struct sockaddr\*)&cliaddr,sizeof(cliaddr));

printf("Connected client details.....\n");

printf("client port number=%d\n", ntohs(cliaddr.sin\_port));

printf("client IP details =%s\n", inet\_ntoa(cliaddr.sin\_addr));

return 0;

}

2.soln:-

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<stdlib.h>

#include<string.h>

#include<signal.h>

#define PORT 12262

#define MAXLINE 200

void dg\_echo(int sockfd,struct sockaddr\* pcliaddr,socklen\_t client){

int n;

socklen\_t len;

char buffer[MAXLINE];

memset(buffer,'\0', sizeof(buffer));

for(;;)

{

len = client;

n=recvfrom(sockfd,buffer,MAXLINE,0, pcliaddr,&len);

printf("message from client %s\n", buffer);

sendto(sockfd,buffer,n,0,pcliaddr,len);

}

}

void generate\_random(int l, int r, int count)

int i;

for(i=0; i<10; i++){

int rand\_num=(rand()%(r-l+1))+1;

printf(“%d”, rand\_num);

}

int main()

{

int lisnfd, br;

socklen\_t clilen,len;

struct sockaddr\_in servaddr, cliaddr;

len=sizeof(servaddr);

servaddr.sin\_family=AF\_INET;

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

servaddr.sin\_port=htons(PORT);

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

if(lisnfd<0)

{

fprintf(stderr,"create error in socket\n");

return 1;

}

br=bind(lisnfd,(struct sockaddr \*)&servaddr,sizeof(servaddr));

if(br==0)

{

printf("Bind success: with return value=%d\n", br);

}

else{

printf("bind unsuccess:with return value=%d\n", br);

printf("Retry........" );

exit(2);

}

dg\_echo(lisnfd,(struct sockaddr\*)&cliaddr,sizeof(cliaddr));

printf("Connected client details.....\n");

printf("client port number=%d\n", ntohs(cliaddr.sin\_port));

printf("client IP details =%s\n", inet\_ntoa(cliaddr.sin\_addr));

return 0;

}

# 3.soln:-

# Server program:-

# #include<stdio.h>

# #include<stdlib.h>

# #include<unistd.h>

# #include<sys/socket.h>

# #include<sys/types.h>

# #include<netinet/in.h>

# #include<arpa/inet.h>

# #include<string.h>

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

# int sockfd, newsockfd, clilen;

# char buffer[256];

# struct sockaddr\_in serv\_addr, cli\_adder;

# int n;

# if(argc<2)

# {

# fprintf(stderr,"error!! no port provided\n");

# exit(1);

# }

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

# if(sockfd<0){

# error("error opening socket");

# exit(1):

# }

# bzero((char \*)&serv\_addr, sizeof(serv\_addr));

# 

# serv\_addr.sin\_family=AF\_INET;

# serv\_addr.sin\_addr.s\_addr=INADDR\_ANY;

# serv\_addr.sin\_port=htons(atoi(arg[1]));

# 

# if(bind(sockfd, (struct sockaddr\*) &serv\_addr, sizeof(serv\_addr))<0){

# error("error on binding");

# exit(1);

# }

# 

# listen(sockfd,5);

# while(1){

# clilen=sizeof(cli\_addr);

# newsockfd= accept(sockfd,(struct sockaddr \*)&cli\_addr, &clilen);

# 

# if(newsockfd<0){

# error("error on accept");

# exit(1);

# }

# printf("new client conn\_ from port no %d and ip %s\n",ntohs(cli\_addr.sin\_port), inet\_ntoa(cli\_addr.sin\_addr));

# 

# bzero(buffer,256);

# n=read(newsockfd,buffer,256);

# 

# if(n<0){

# error("error reading from socket");

# exit(1);

# }

# 

# printf("here is the message:%s\n",buffer);

# n=write(newsockfd,"i got the message");

# 

# if(n<0){

# error("error writing to socket");

# exit(1);

# }}

# return 0;

# }

# Client program:-

# #include<stdio.h>

# #include<stdlib.h>

# #include<unistd.h>

# #include<sys/socket.h>

# #include<sys/types.h>

# #include<netinet/in.h>

# #include<arpa/inet.h>

# #include<string.h>

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

# int sockfd, newsockfd, clilen;

# char buffer[256];

# struct sockaddr\_in serv\_addr, cli\_adder;

# int n;

# if(argc<2)

# {

# fprintf(stderr,"error!! no port provided\n");

# exit(1);

# }

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

# if(sockfd<0){

# error("error opening socket");

# exit(1):

# }

# bzero((char \*)&serv\_addr, sizeof(serv\_addr));

# 

# serv\_addr.sin\_family=AF\_INET;

# serv\_addr.sin\_addr.s\_addr=INADDR\_ANY;

# serv\_addr.sin\_port=htons(atoi(arg[1]));

# 

# if(bind(sockfd, (struct sockaddr\*) &serv\_addr, sizeof(serv\_addr))<0){

# error("error on binding");

# exit(1);

# }

# 

# listen(sockfd,5);

# while(1){

# clilen=sizeof(cli\_addr);

# newsockfd= accept(sockfd,(struct sockaddr \*)&cli\_addr, &clilen);

# 

# if(newsockfd<0){

# error("error on accept");

# exit(1);

# }

# printf("new client conn\_ from port no %d and ip %s\n",ntohs(cli\_addr.sin\_port), inet\_ntoa(cli\_addr.sin\_addr));

# 

# bzero(buffer,256);

# n=read(newsockfd,buffer,256);

# 

# if(n<0){

# error("error reading from socket");

# exit(1);

# }

# 

# printf("here is the message:%s\n",buffer);

# n=write(newsockfd,"i got the message");

# 

# if(n<0){

# error("error writing to socket");

# exit(1);

# }}

# return 0;

# }

# 4.sol:-

# #include<stdio.h>

# #include<stdlib.h>

# #include<unistd.h>

# #include<sys/socket.h>

# #include<sys/types.h>

# #include<netinet/in.h>

# #include<arpa/inet.h>

# #include<string.h>

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

# int sockfd, newsockfd, clilen;

# char buffer[256];

# struct sockaddr\_in serv\_addr, cli\_adder;

# int n;

# if(argc<2)

# {

# fprintf(stderr,"error!! no port provided\n");

# exit(1);

# }

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

# if(sockfd<0){

# error("error opening socket");

# exit(1):

# }

# bzero((char \*)&serv\_addr, sizeof(serv\_addr));

# 

# serv\_addr.sin\_family=AF\_INET;

# serv\_addr.sin\_addr.s\_addr=INADDR\_ANY;

# serv\_addr.sin\_port=htons(atoi(arg[1]));

# 

# if(bind(sockfd, (struct sockaddr\*) &serv\_addr, sizeof(serv\_addr))<0){

# error("error on binding");

# exit(1);

# }

# 

# listen(sockfd,5);

# while(1){

# clilen=sizeof(cli\_addr);

# newsockfd= accept(sockfd,(struct sockaddr \*)&cli\_addr, &clilen);

# 

# if(newsockfd<0){

# error("error on accept");

# exit(1);

# }

# printf("new client conn\_ from port no %d and ip %s\n",ntohs(cli\_addr.sin\_port), inet\_ntoa(cli\_addr.sin\_addr));

# 

# bzero(buffer,256);

# n=read(newsockfd,buffer,256);

# x=read(inta[][]=new int[m][n]);

# 

# {

# }

# 

# if(n<0){

# error("error reading from socket");

# exit(1);

# }

# 

# printf("here is the message:%s\n",buffer);

# n=write(newsockfd,"Addition of MxN matrix");

# 

# if(n<0){

# error("error writing to socket");

# exit(1);

# }}

# return 0;

# }

# 5.soln:-

# #include<stdio.h>

# #include<unistd.h>

# #include<sys/socket.h>

# #include<sys/types.h>

# #include<netinet/in.h>

# #include<netdb.h>

# #define SERV\_TCP\_PORT 1718

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

# {

# int sockfd,newsockfd,clength;

# struct sockaddr\_in serv\_addr;

# struct hostent \*server;

# char buffer[4096];

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

# serv\_addr.sin\_family=AF\_INET;

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

# serv\_addr.sin\_port=htons("SERV\_TCP\_PORT");

# printf("\nREADY TO SEND DATA");

# connect(sockfd,(struct sockaddr\*)&serv\_addr,sizeof(serv\_addr));

# printf("\nENTER THE MESSAGE TO SEND");

# r=0;

# n1=n;

# while(n>0){

# r=r\*10+n%10;

# n=n/10;

# }

# If(r==n1)

# printf(“palindrome\n”);

# else

# print(“not pallindrome”);

# printf("\nClient:-");

# fgets(buffer,4096,stdin);

# write(sockfd,buffer,4096);

# printf("\n");

# close(sockfd);

# return 0;

# }