**MINOR ASSIGNMENT-04**

**UNIX Network Programming (CSE 4042)**

**Q1.**

#include<stdio.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<unistd.h>

int main()

{

int sockfd,count=0,i;

printf("PID=%ld\n",(long)getpid());

for(i=0;i<=20;i++){

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

count=count+1;

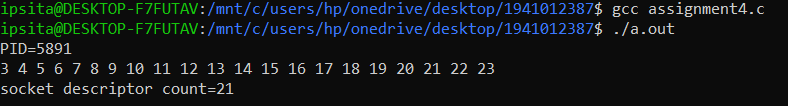
printf("%d ",sockfd);

}

printf("\nsocket descriptor count=%d\n",count);

while(1);

return 0;

}

**Q2.**

#include<stdio.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<unistd.h>

int main()

{

int sockfd,count=0,i;

printf("PID=%ld\n",(long)getpid());

for(i=0;i<=20;i++){

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

count=count+1;

if(sockfd%3==0){

printf("%d ",sockfd);

}

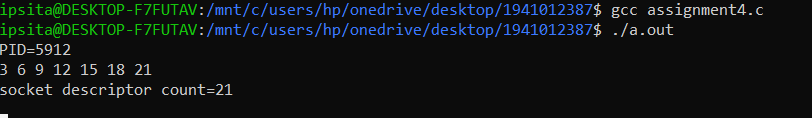
}

printf("\nsocket descriptor count=%d\n",count);

while(1);

return 0;

}



**Q3.**

#include<stdio.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<unistd.h>

int main()

{

int sockfd,count=0,i;

printf("PID=%ld\n",(long)getpid());

for(i=0;i<=20;i++){

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

count=count+1;

if(sockfd%3==0){

printf("%d ",sockfd);

}

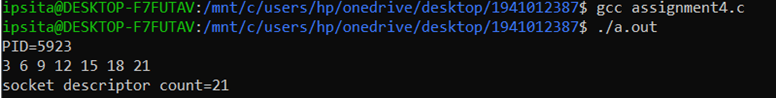
}

printf("\nsocket descriptor count=%d\n",count);

while(1);

return 0;

}



**Q4. For UDP-datagram socket**

#include<stdio.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<unistd.h>

int main()

{

int sockfd,count=0,i;

printf("PID=%ld\n",(long)getpid());

for(i=0;i<=20;i++){

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

count=count+1;

if(sockfd%3==0){

printf("%d ",sockfd);

}

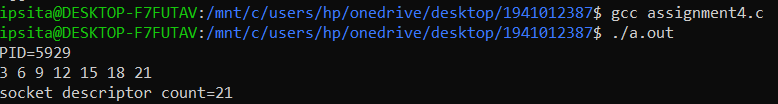
}

printf("\nsocket descriptor count=%d\n",count);

while(1);

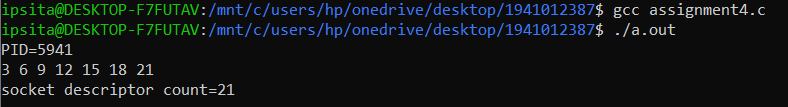
return 0;

}



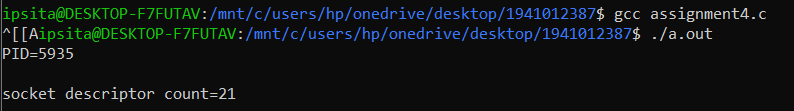
**For SCTP-sequenced packet socket**

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



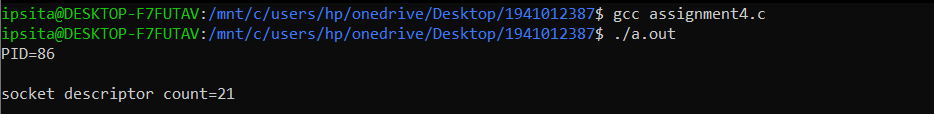
**For raw socket**

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



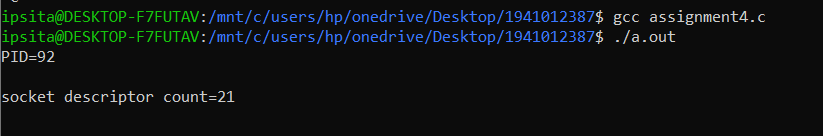
**For Unix domain socket**

sockfd=socket(AF\_LOCAL,SOCK\_SEQPACKET,IPPROTO\_TCP);

****

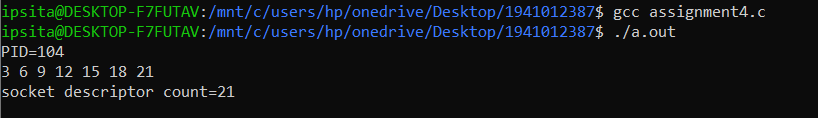
**For Routing socket**

sockfd=socket(AF\_ROUTE,SOCK\_SEQPACKET,IPPROTO\_TCP);



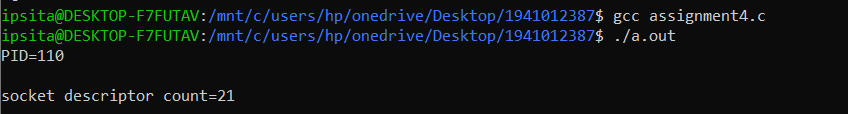
**For IPV6 stream socket**

sockfd=socket(AF\_INET6 ,SOCK\_STREAM,IPPROTO\_TCP);

****

**For Key management socket**

sockfd=socket(AF\_KEY ,SOCK\_STREAM,IPPROTO\_TCP);

****

**Q5.**

**SERVER---**

#include<stdio.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<stdlib.h>

#include<string.h>

#include<time.h>

#include<unistd.h>

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

{

int listenfd,connfd,len;

struct sockaddr\_in servaddr,clientaddr;

char buff[1024];

time\_t ticks;

len=sizeof(struct sockaddr\_in);

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

servaddr.sin\_family=AF\_INET;

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

servaddr.sin\_port=htons(0);

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

getsockname(listenfd, (struct sockaddr \*)&servaddr, &len);

printf("After bind ephemeral port=%d\n",(int)ntohs(servaddr.sin\_port));

listen(listenfd,5);

connfd=accept(listenfd, (struct sockaddr \*)&clientaddr,&len);

ticks=time(NULL);

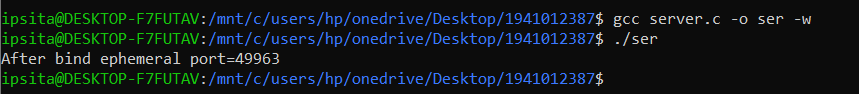
snprintf(buff,sizeof(buff),"%s\r\n",ctime(&ticks));

write(connfd,buff,strlen(buff));

write(connfd,"ITER",4);

close(connfd);

}



**CLIENT---**

#include<stdio.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<stdlib.h>

#include<string.h>

#include<time.h>

#include<arpa/inet.h>

#include<unistd.h>

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

{

int sockfd,n,conn,len;

char recvline[1024];

struct sockaddr\_in servaddr;

len=sizeof(struct sockaddr\_in);

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

servaddr.sin\_family=AF\_INET;

servaddr.sin\_addr.s\_addr=inet\_addr(argv[1]);//get ip from server

servaddr.sin\_port=htons(atoi(argv[2])); // Get the port from the server

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

n=read(sockfd,recvline,1024);

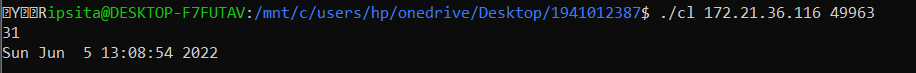
printf("%d\n",n);

recvline[n]=0;

printf("%s",recvline);

close(sockfd);

}



**Q6.**

**For Server---**

#include <stdio.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<stdlib.h>

#include<string.h>

#include<time.h>

#define SERVERPORT 33456

#define MAXLINE 100

#define MAXCLIENT 150

void data\_transmission(int connFD,char buffer[1024],struct sockaddr\_in clientAddr){

int n=read(connFD,buffer,1024);

buffer[n]=0;

printf("Client address; %s:%d Received : %s\n",inet\_ntoa(clientAddr.sin\_addr),ntohs(clientAddr.sin\_port),buffer);

for(int i=0;i<30;i++)

{

printf("%s || %d\n",buffer,i);

sleep(1);

}

}

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

struct sockaddr\_in sa,clientAddr;

char buffer[1024];

pid\_t chpid;

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

int len=sizeof(sa);

sa.sin\_family=AF\_INET;

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

sa.sin\_port=htons(SERVERPORT);

int bindstatus=bind(socketFd,(struct sockaddr\*)&sa,len);

if(bindstatus==-1){

perror("Bind");

exit(EXIT\_FAILURE);

}

//getsockname(sockFd,(struct sockaddr\*)&sa,&len);

printf("Chosen port=%d\n",ntohs(sa.sin\_port));

listen(socketFd,MAXCLIENT);

for(int i=0;i<MAXCLIENT;i++)

{

int connFD=accept(socketFd,(struct sockaddr\*)&clientAddr,&len);

if((chpid=fork())==0)

{

close(socketFd);

data\_transmission(connFD,buffer,clientAddr);

close(connFD);

exit(EXIT\_SUCCESS);

}

close(connFD);

}

}

**For Client---**

#include <stdio.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<stdlib.h>

#include<string.h>

#include<time.h>

#define SERVERPORT 33456

#define MAXLINE 100

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

{

struct sockaddr\_in sa;

if(argc!=3){

printf("Invalid C.L.A format,must need 3 args \n");

exit(EXIT\_FAILURE);

}

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

sa.sin\_family=AF\_INET;

sa.sin\_addr.s\_addr=inet\_addr(argv[1]);

sa.sin\_port=htons(SERVERPORT);

int len=sizeof(sa);

if(connect(sockFd,(struct sockaddr\*)&sa,len)==-1){

perror("Connect");

exit(EXIT\_FAILURE);

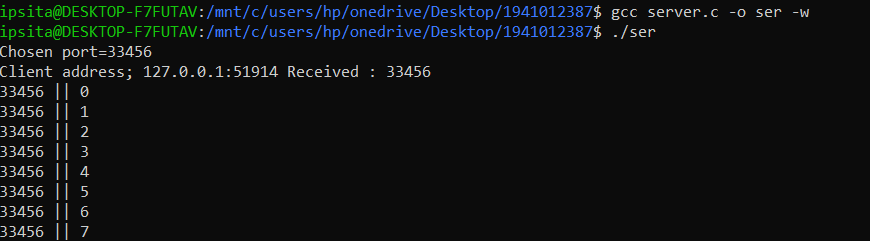
}

write(sockFd,argv[2],strlen(argv[2]));

}

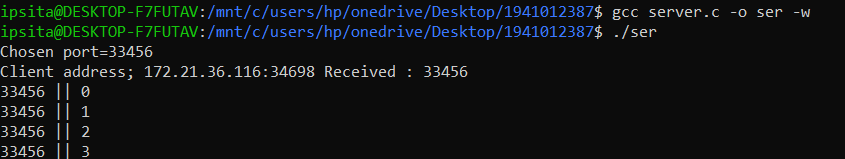
**(a) $./client 127.0.0.1 specify the local host (127.0.0.1), which is running the server(loopback address).**

****

****

**(b) $./client xxx.xxx.xxx.xxx specify the ethernet interface of the server, which is running the server(IP address).**

****

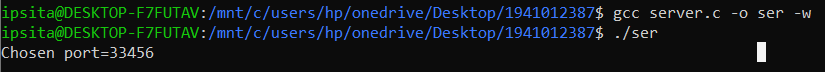
****

**(c) $./client 172.18.152.120 specify a host that is not running your server.**

****

o/p-Got blank output

**(d) $./client 172.18.152.102 specify an IP address that is on the local subnet but the host ID is nonexistent**

****

****

**Q8.**