1. Implementation of Chat Server using TCP

Source Code:

TCP Client:

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

int main()

{

int sock, bytes\_recieved;

char send\_data[1024],recv\_data[1024];

struct sockaddr\_in server\_addr;

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

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(9059);

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

bzero(&(server\_addr.sin\_zero),8);

connect(sock, (struct sockaddr \*)&server\_addr,

sizeof(struct sockaddr));

while(1)

{

printf("\nSEND (q or Q to quit) : ");

scanf("%s",send\_data);

if (strcmp(send\_data , "q") != 0 && strcmp(send\_data , "Q") != 0)

send(sock,send\_data,strlen(send\_data), 0);

else

{

send(sock,send\_data,strlen(send\_data), 0);

close(sock);

break;

}

bytes\_recieved=recv(sock,recv\_data,1024,0);

recv\_data[bytes\_recieved] = '\0';

if (strcmp(recv\_data , "q") == 0 || strcmp(recv\_data , "Q") == 0)

{

close(sock);

exit;

}

else

printf("\nRecieved data = %s " , recv\_data);

}

return 0;

}

**TCPServer:**

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

int main()

{

int sock, connected, bytes\_recieved , true = 1;

char send\_data [1024] , recv\_data[1024];

struct sockaddr\_in server\_addr,client\_addr;

int sin\_size;

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

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(9059);

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

bind(sock, (struct sockaddr \*)&server\_addr, sizeof(struct sockaddr));

listen(sock, 5);

printf("\nTCPServer Waiting for client ");

fflush(stdout);

while(1)

{

sin\_size = sizeof(struct sockaddr\_in);

connected = accept(sock, (struct sockaddr \*)&client\_addr,&sin\_size);

printf("\n I got a connection from (%s , %d)",

inet\_ntoa(client\_addr.sin\_addr),ntohs(client\_addr.sin\_port));

while (1)

{

bytes\_recieved = recv(connected,recv\_data,1024,0);

recv\_data[bytes\_recieved] = '\0';

if (strcmp(recv\_data , "q") == 0 || strcmp(recv\_data , "Q") == 0)

{

close(connected);

break;

}

else

printf("\n RECIEVED DATA = %s " , recv\_data);

printf("\n SEND (q or Q to quit) : ");

scanf("%s",send\_data);

if (strcmp(send\_data , "q") == 0 || strcmp(send\_data , "Q") == 0)

{

send(connected, send\_data,strlen(send\_data), 0);

close(connected);

break;

}

else

send(connected, send\_data,strlen(send\_data), 0);

fflush(stdout);

}

}

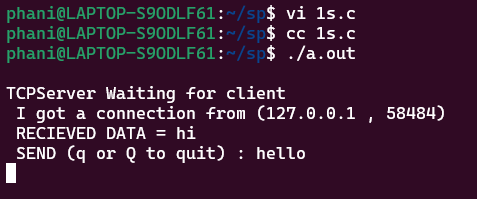
close(sock);

return 0;

}

Output:

**Server side:**



**Client side:**

A screen shot of a computer program

Description automatically generated

1. Implementation of Daytime Server using TCP.

Source Code:

DayTimeClient:

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

int main()

{

int sock,bytes;

char recv\_data[1024];

struct sockaddr\_in server\_addr;

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

server\_addr.sin\_family=AF\_INET;

server\_addr.sin\_port=htons(8523);

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

connect(sock,(struct sockaddr\*)&server\_addr,sizeof(struct sockaddr));

bytes=recv(sock,recv\_data,1024,0);

printf("Received data: %s",recv\_data);

close(sock);

return 0;

}

Daytime Server:

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<unistd.h>

#include<errno.h>

#include<time.h>

int main()

{

int sock,connected,bytes;

struct sockaddr\_in server\_addr,client\_addr;

int sin\_size;

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

server\_addr.sin\_family=AF\_INET;

server\_addr.sin\_port=htons(8523);

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

bind(sock,(struct sockaddr\*)&server\_addr,sizeof(struct sockaddr));

listen(sock,5);

printf("TCP Waiting fo Client");

sin\_size=sizeof(struct sockaddr\_in);

connected=accept(sock,(struct sockaddr\*)&client\_addr,&sin\_size);

time\_t t;

time(&t);

send(connected,ctime(&t),strlen(ctime(&t)),0);

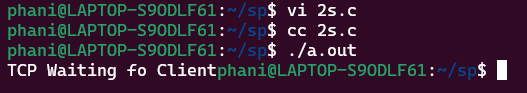
close(connected);

close(sock);

return 0;

}

Output:



A screenshot of a computer

Description automatically generated

1. Implementation of Concurrent Echo Server using TCP.

SourceCode:

Echo Client:

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

int main()

{

int sock, bytes\_recieved;

char send\_data[1024],recv\_data[1024];

struct sockaddr\_in server\_addr;

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

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(9934);

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

connect(sock, (struct sockaddr \*)&server\_addr,

sizeof(struct sockaddr));

while(1)

{

printf("\nSEND (q or Q to quit) : ");

scanf("%s",send\_data);

if (strcmp(send\_data , "q") != 0 && strcmp(send\_data , "Q") != 0)

send(sock,send\_data,strlen(send\_data), 0);

else

{

send(sock,send\_data,strlen(send\_data), 0);

close(sock);

break;

}

bytes\_recieved=recv(sock,recv\_data,1024,0);

recv\_data[bytes\_recieved] = '\0';

if (strcmp(recv\_data , "q") == 0 || strcmp(recv\_data , "Q") == 0)

{

close(sock);

exit;

}

else

printf("\nRecieved data = %s " , recv\_data);

}

return 0;

}

Echo Server:

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

int main()

{

int sock, connected, bytes\_recieved , true = 1;

char send\_data [1024] , recv\_data[1024];

struct sockaddr\_in server\_addr,client\_addr;

int sin\_size;

int pid;

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

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(9934);

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

bind(sock, (struct sockaddr \*)&server\_addr, sizeof(struct sockaddr));

listen(sock, 5);

printf("\nTCPServer Waiting for client ");

fflush(stdout);

while(1)

{

sin\_size = sizeof(struct sockaddr\_in);

connected = accept(sock, (struct sockaddr \*)&client\_addr,&sin\_size);

printf("\n I got a connection from (%s , %d)",

inet\_ntoa(client\_addr.sin\_addr),ntohs(client\_addr.sin\_port));

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

{

close(sock);

while (1)

{

bytes\_recieved = recv(connected,recv\_data,1024,0);

recv\_data[bytes\_recieved] = '\0';

if (strcmp(recv\_data , "q") == 0 || strcmp(recv\_data , "Q") == 0)

{

close(connected);

break;

}

else

printf("\n RECIEVED DATA = %s " , recv\_data);

strcpy(send\_data,recv\_data);

if (strcmp(send\_data , "q") == 0 || strcmp(send\_data , "Q") == 0)

{

send(connected, send\_data,strlen(send\_data), 0);

close(connected);

break;

}

else

send(connected, send\_data,strlen(send\_data), 0);

fflush(stdout);

}

exit(0);

}

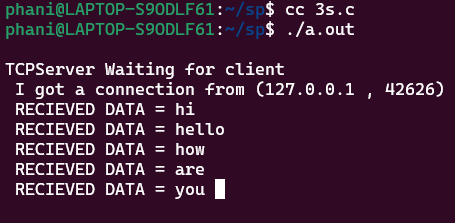
close(connected);

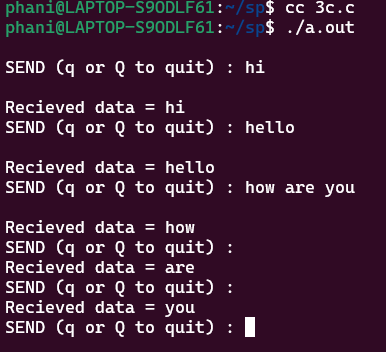
}

return 0;

}

**Output:**





1. Implementation of Computational Server using TCP.

SourceCode:

CSClient:

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

int main()

{

int sock,bytes;

char send\_data[1024],recv\_data[1024];

struct sockaddr\_in server\_addr;

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

server\_addr.sin\_family=AF\_INET;

server\_addr.sin\_port=htons(8563);

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

connect(sock,(struct sockaddr \*)&server\_addr,sizeof(struct sockaddr));

printf("Enter data\n");

scanf("%s",send\_data);

send(sock,send\_data,strlen(send\_data),0);

bytes=recv(sock,recv\_data,1024,0);

recv\_data[bytes]='\0';

close(sock);

printf("Received Data %s",recv\_data);

return 0;

}

CSServer:

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

#include<math.h>

int main()

{

int sock,connected,bytes;

char send\_data[1024],recv\_data[1024];

struct sockaddr\_in server\_addr,client\_addr;

int sin\_size;

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

server\_addr.sin\_family=AF\_INET;

server\_addr.sin\_port=htons(8563);

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

bind(sock,(struct sockaddr \*)&server\_addr,sizeof(struct sockaddr));

listen(sock,5);

printf("TCPServer Waiting for client\n");

sin\_size=sizeof(struct sockaddr\_in);

connected=accept(sock,(struct sockaddr \*)&client\_addr,&sin\_size);

bytes=recv(connected,recv\_data,1024,0);

recv\_data[bytes]='\0';

int i=0,k=0;

char ch1[9];

int k1=0;

while(recv\_data[i]!='\0')

{

ch1[k]=recv\_data[i];

k=k+1;

i=i+1;

}

ch1[k]='\0';

int val=atoi(ch1);

printf("%s\n",ch1);

int res=val\*val;

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

while(res>0)

{

int d=res%10;

res=res/10;

send\_data[k1]=d+'0';

k1=k1+1;

}

send\_data[k1]='\0';

char send\_data2[1024];

int k2=0;

for(i=k1-1;i>=0;i--)

{

send\_data2[k2]=send\_data[i];

k2=k2+1;

}

send\_data2[k2]='\0';

printf("%s",send\_data2);

send(connected,send\_data2,strlen(send\_data2),0);

close(connected);

close(sock);

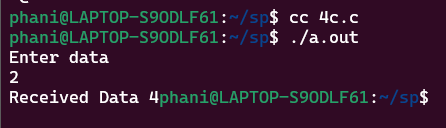
return 0;

}

Output:

A screenshot of a computer program

Description automatically generated



1. Implementation of DNS Server using TCP.

SourceCode:

DNSClient:

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<unistd.h>

#include<errno.h>

int main()

{

int sock,bytes;

char send\_data[1024],recv\_data[1024];

struct sockaddr\_in server\_addr;

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

server\_addr.sin\_family=AF\_INET;

server\_addr.sin\_port=htons(8563);

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

connect(sock,(struct sockaddr\*)&server\_addr,sizeof(struct sockaddr));

scanf("%s",send\_data);

send(sock,send\_data,strlen(send\_data),0);

bytes=recv(sock,recv\_data,1024,0);

recv\_data[bytes]='\0';

printf("%s",recv\_data);

return 0;

}

DNSServer:

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<errno.h>

#include<string.h>

int main()

{

int sock,connected,bytes;

char send\_data[1024],recv\_data[1024];

struct sockaddr\_in server\_addr,client\_addr;

int sin\_size;

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

server\_addr.sin\_family=AF\_INET;

server\_addr.sin\_port=htons(8563);

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

bind(sock,(struct sockaddr \*)&server\_addr,sizeof(struct sockaddr));

listen(sock,5);

printf("\nTCP Server Waiting for client");

sin\_size=sizeof(struct sockaddr\_in);

connected=accept(sock,(struct sockaddr \*)&client\_addr,&sin\_size);

bytes=recv(connected,recv\_data,1024,0);

recv\_data[bytes]='\0';

char dnsname[][100]={"google.com","facebook.com","amazon.com"};

char dnsip[][100]={"127.0.0.1","128.0.0.1","129.0.0.1"};

int i,index=-1;

for(i=0;i<3;i++)

{

if(strcmp(dnsname[i],recv\_data)==0)

{

index=i;

break;

}

}

if(index==-1)

{

send(connected,"-1",1,0);

close(connected);

close(sock);

}

else

{

strcpy(send\_data,dnsip[index]);

}

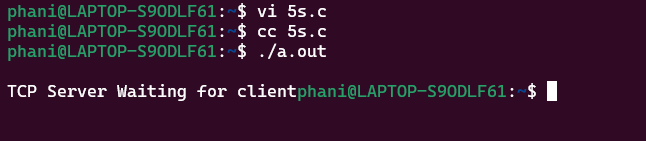
send(connected,send\_data,strlen(send\_data),0);

close(connected);

close(sock);

return 0;}

Output:



A screenshot of a computer

Description automatically generated

1. Implementation of Authentication Server using TCP.

SourceCode:

**Authclient:**

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<unistd.h>

#include<errno.h>

int main()

{

int sock,bytes;

char send\_data[1024],recv\_data[1024];

char password[1024];

struct sockaddr\_in server\_addr;

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

server\_addr.sin\_family=AF\_INET;

server\_addr.sin\_port=htons(8563);

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

connect(sock,(struct sockaddr\*)&server\_addr,sizeof(struct sockaddr));

printf("Enter Username");

scanf("%s",send\_data);

send(sock,send\_data,strlen(send\_data),0);

printf("Enter Password");

scanf("%s",password);

send(sock,password,strlen(password),0);

bytes=recv(sock,recv\_data,1024,0);

recv\_data[bytes]='\0';

printf("%s",recv\_data);

return 0;

}

**AuthServer:**

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<errno.h>

#include<string.h>

int main()

{

int sock,connected,bytes,bytes1;

char send\_data[1024],recv\_data[1024];

char pass[1024];

struct sockaddr\_in server\_addr,client\_addr;

int sin\_size;

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

server\_addr.sin\_family=AF\_INET;

server\_addr.sin\_port=htons(8563);

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

bind(sock,(struct sockaddr \*)&server\_addr,sizeof(struct sockaddr));

listen(sock,5);

printf("\nTCP Server Waiting for client");

while(1)

{

sin\_size=sizeof(struct sockaddr\_in);

connected=accept(sock,(struct sockaddr \*)&client\_addr,&sin\_size);

while(1)

{

bytes=recv(connected,recv\_data,1024,0); recv\_data[bytes]='\0';

if(strcmp(recv\_data,"q")==0)

{

close(connected);

break;

}

bytes1=recv(connected,pass,1024,0);

pass[bytes1]='\0';

char username[][100]={"user1","user2","user3"};

char password[][100]={"user1","user2","user3"};

int i,index=-1;

for(i=0;i<3;i++)

{

if(strcmp(username[i],recv\_data)==0)

{

index=i;

break;

}

}

if(index==-1)

{

strcpy(send\_data,"Wrong User");

send(connected,send\_data,1024,0);

}

else

{

if(strcmp(pass,password[i])==0) {

strcpy(send\_data,"Success");

}

else

{

strcpy(send\_data,"Error");

}

}

send(connected,send\_data,strlen(send\_data),0);

}

}

return 0;

}

Output:



A screenshot of a computer program

Description automatically generated

1. Implementation of FTP Using TCP

SourceCode:

FTPClient:

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

int main()

{

int sock, bytes\_recieved;

char send\_data[1024],recv\_data[1024];

struct sockaddr\_in server\_addr;

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

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(9934);

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

connect(sock, (struct sockaddr \*)&server\_addr,

sizeof(struct sockaddr));

bytes\_recieved=recv(sock,recv\_data,1024,0);

recv\_data[bytes\_recieved] = '\0';

printf("\nRecieved data = %s " , recv\_data);

return 0;

}

FTPServer:

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

int main()

{

int sock, connected, bytes\_recieved , true = 1;

char send\_data [1024] , recv\_data[1024];

struct sockaddr\_in server\_addr,client\_addr;

int sin\_size;

int pid;

FILE \*fp;

char ch;

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

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(9934);

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

bind(sock, (struct sockaddr \*)&server\_addr, sizeof(struct sockaddr));

listen(sock, 5);

printf("\nTCPServer Waiting for client ");

fflush(stdout);

sin\_size = sizeof(struct sockaddr\_in);

connected = accept(sock, (struct sockaddr \*)&client\_addr,&sin\_size);

printf("\n I got a connection from (%s , %d)",

inet\_ntoa(client\_addr.sin\_addr),ntohs(client\_addr.sin\_port));

int k=0;

fp=fopen("test.txt","r");

do

{

ch=fgetc(fp);

printf("%c\n",ch);

send\_data[k]=ch;

k=k+1;

}while(ch!=EOF);

send\_data[k]='\0';

send(connected, send\_data,strlen(send\_data), 0);

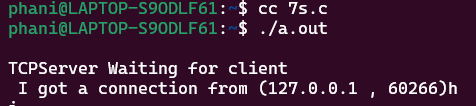
fflush(stdout);

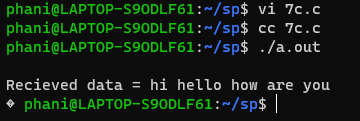
close(connected);

return 0;

}

Output:





8.Implementation Of Caeser Cipher Using UDP

Soruce Code:

#include<stdio.h>

#include<stdlib.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<netdb.h>

#include<string.h>

#include<unistd.h>

int main()

{

int sock,port,len,key=2;

char chars[26]={'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'};

struct sockaddr\_in serveraddr;

char send\_data[1024],recv\_data[1024];

printf("Enter Port");

scanf("%d",&port);

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

serveraddr.sin\_family=AF\_INET;

serveraddr.sin\_port=htons(port);

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

printf("Enter the String");

scanf("%s",send\_data);

char caeser[1024];

int i,j;

for(i=0;i<strlen(send\_data);i++)

{

for(j=0;j<26;j++){

if(send\_data[i]==chars[j])

{

int k=j+key;

if(k>=25)

{

k=k%26;

}

caeser[i]=chars[k];

}

}

}

caeser[strlen(send\_data)]='\0';

printf("Caeser Text %s",caeser);

sendto(sock,caeser,strlen(caeser),0,(struct sockaddr\*)&serveraddr,sizeof(serveraddr));

close(sock);

}

CaeserServer:

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

#include<netdb.h>

int main()

{

int sock,connect,bytes,port,len;

struct sockaddr\_in serveraddr,clientaddr;

char recv\_data[1024],chars[26]={'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'};

printf("Enter Port");

scanf("%d",&port);

int sin\_size;

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

if(sock<0)

{

printf("Sock Error");

}

serveraddr.sin\_family=AF\_INET;

serveraddr.sin\_port=htons(port);

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

bind(sock,(struct sockaddr \*)&serveraddr,sizeof(struct sockaddr));

listen(sock,5);

printf("UDP Server Waiting For Client\n"); sin\_size=sizeof(struct sockaddr\_in);

bytes=recvfrom(sock,recv\_data,1024,0,(struct sockaddr\*)&clientaddr,sin\_size);

recv\_data[bytes]='\0';

char str[1024];

int i,j;

for(i=0;i<strlen(recv\_data);i++)

{

for(j=0;j<26;j++)

{

int key=2;

int k=j-key;

if(recv\_data[i]==chars[j])

{

if(k<0)

{

k=k+26;

}

recv\_data[i]=chars[k];

}

}

}

recv\_data[bytes]='\0';

printf("%s",recv\_data);

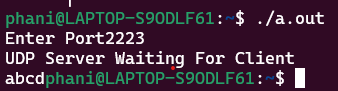
sendto(sock,recv\_data,strlen(recv\_data),0,(struct sockaddr\*)&clientaddr,sin\_size);

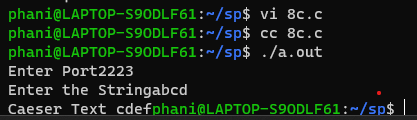
close(connect);

close(sock);

}

Output:





1. Implementation of daytime Server as a daemon.

SourceCode:

Daemonclient:

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<netdb.h>

#include<stdio.h>

#include<string.h>

#include<unistd.h>

int main()

{

int sock,port,n;

char recv\_data[1024+1];

struct sockaddr\_in serveraddr;

printf("Enter Port");

scanf("%d",&port);

if((sock=socket(AF\_INET,SOCK\_STREAM,0))<0)

{

printf("Socket Error");

}

serveraddr.sin\_family=AF\_INET;

serveraddr.sin\_port=htons(port);

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

if(connect(sock,(struct sockaddr\*)&serveraddr,sizeof(struct sockaddr))<0)

{

printf("Connect Error");

}

n=recv(sock,recv\_data,1024,0);

recv\_data[n]='\0';

printf("%s",recv\_data);

close(sock);

return 0;

}

DaemonServer:

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<string.h>

#include<signal.h>

#include<syslog.h>

#include<time.h>

#include<sys/stat.h>

#include<fcntl.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#define MAXFD 64

int daemon\_init(const char \*pname,int facility)

{

int i;

pid\_t pid;

if((pid=fork())<0)

{

return (-1);

}

else if(pid)

{

\_exit(0);

}

if(setsid()<0)

{

return -1;

}

signal(SIGHUP,SIG\_IGN);

if((pid=fork())<0)

{

return (-1);

}

else if(pid)

{

\_exit(0);

}

chdir("/");

for(i=0;i<MAXFD;i++)

{

close(i);

}

open("/dev/null",444);

open("/dev/null",666);

open("/dev/null",O\_RDWR);

openlog(pname,LOG\_PID,facility);

return (0);

}

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

{

int listenfd,connect;

socklen\_t addrlen,len;

struct sockaddr\_in serveraddr,clientaddr;

char buff[1024];

time\_t ticks;

int port;

printf("ENter port");

scanf("%d",&port);

daemon\_init(argv[0],0);

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

serveraddr.sin\_family=AF\_INET;

serveraddr.sin\_port=htons(port);

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

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

listen(listenfd,5);

while(1)

{

int sin\_size=sizeof(struct sockaddr\_in);

connect=accept(listenfd,(struct sockaddr\*)&clientaddr,&sin\_size);

time(&ticks);

send(connect,ctime(&ticks),strlen(ctime(&ticks)),0);

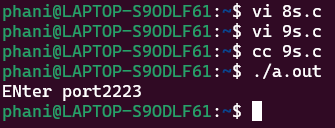
close(connect);

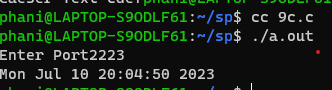
}

return 0;

}

Output:





10. Implementation of TCP echo server using threads.

SourceCode:

Threadclient:

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<netdb.h>

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<unistd.h>

#include<errno.h>

#include<pthread.h>

void str\_cli(int);

int sockfd;

void \*copyto(void \*);

main( )

{

int port;

struct sockaddr\_in servaddr;

printf("enter port number:");

scanf("%d",&port);

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

bzero(&servaddr, sizeof(servaddr));

servaddr.sin\_family = AF\_INET;

servaddr.sin\_port = htons(port);

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

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

str\_cli(sockfd); /\* do it all \*/

exit(0);

}

void str\_cli(int sockfd)

{

char recv\_data[1024];

pthread\_t tid;

int bytes\_received,i;

for( ; ; )

{

pthread\_create(&tid,NULL,copyto,NULL);

bytes\_received=recv(sockfd, recv\_data, 1024,0);

recv\_data[bytes\_received]='\0';

printf("received data=%s",recv\_data);

}

//fflush(stdout);

close(sockfd);

return;

}

void \*copyto(void \*arg)

{

char send\_data[1024];

char recv\_data[1024];

int i,bytes\_received;

printf("\nEnter data to send");

scanf(" %s",send\_data);

send(sockfd,send\_data,strlen(send\_data),0);

}

ThreadServer:

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <netdb.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <time.h>

#include <sys/wait.h>

#include<signal.h>

#include<pthread.h>

void str\_echo(int);

static void \*doit(void \*);

main( )

{

int listenfd, connfd,port,\*iptr;

socklen\_t clilen;

pthread\_t tid;

struct sockaddr\_in cliaddr, servaddr;

printf("enter port nu:");

scanf("%d",&port);

listenfd = 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(port);

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

listen(listenfd, 5);

fflush(stdout);

for ( ; ; )

{

clilen = sizeof(cliaddr);

iptr=malloc(sizeof(int));

\*iptr=accept(listenfd,(struct sockaddr \*)&cliaddr,&clilen);

pthread\_create(&tid,NULL,&doit,iptr);

/\* parent closes connected socket \*/

}

exit(0);

}

static void \*doit(void \*arg)

{

int connfd;

connfd = \*((int \*) arg);

free(arg);

pthread\_detach(pthread\_self());

str\_echo(connfd); /\* same function as before \*/

close(connfd); /\* done with connected socket \*/

}

void str\_echo(int sockfd)

{

char recv\_data[1024],send\_data[1024];

int i,bytes\_received;

while(1)

{

again:

for( ; ;)

{

bytes\_received = recv(sockfd, recv\_data, sizeof(recv\_data),0);

recv\_data[bytes\_received]='\0';

if (bytes\_received<0&&errno == EINTR)

goto again;

else if (bytes\_received<0)

printf("str\_echo: read error");

printf("\n received data is %s",recv\_data);

send(sockfd, recv\_data, strlen(recv\_data),0);

}

fflush(stdout);

return;

}

}

Output:

