1. **Implementation of Daytime Server using TCP.**

**Client**

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<errno.h>

#include<unistd.h>

int main()

{

int sock,bytes\_recieved;

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(7474);

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

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

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

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

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

printf("\nDate and Time in Server = %s \n" , recv\_data);

return 0;

}

**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>

#include<time.h>

int main()

{

int sock,connected,true=1;

char send\_data[1024];

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

int sin\_size;

struct tm\* ptr;

time\_t t;

time(&t);

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

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

server\_addr.sin\_port = htons(7474);

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

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

listen(sock, 5);

fflush(stdout);

printf("Date Time of server is sent\n");

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

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

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

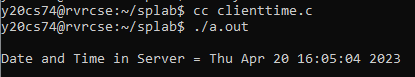
close(connected);

fflush(stdout);

close(sock);

return 0;

}





1. Implementation of Echo Server using TCP.

Echo client

**#include<stdio.h>**

**#include<sys/socket.h>**

**#include<sys/types.h>**

**#include<netinet/in.h>**

**#include<string.h>**

**#include <stdlib.h>**

**#include <unistd.h>**

**#include <errno.h>**

**int main()**

**{**

**int sock, bytes\_recieved,port;**

**printf("Enter port :");**

**scanf("%d",&port);**

**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(port);**

**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;**

**}**

**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,port;**

**char recv\_data[1024];**

**printf("Enter port :");**

**scanf("%d",&port);**

**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(port);**

**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);**

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

**}**

**fflush(stdout);**

**}**

**}**

**close(sock);**

**return 0;**

**}**

**Echo concurrent at server side**

**#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,port;**

**char recv\_data[1024];**

**pid\_t pid;**

**printf("Enter port :");**

**scanf("%d",&port);**

**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(port);**

**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);**

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

**}**

**fflush(stdout);**

**}**

**exit(0);**

**}**

**close(connected);**

**}**

**return 0;**

**}**

1. Implementation of Computational Server using TCP.
2. Implementation of DNS Server using TCP.
3. Implementation of Authentication Server using TCP.
4. Implementation of Concurrent Chat server using TCP.
5. Implementation of Concurrent FTP Server using TCP.
6. Implementation of Caesar cipher using UDP.
7. Implementation of daytime Server as a daemon.
8. Implementation of TCP echo server using threads.