TCP_SERVER

```
#include<stdio.h>
#include<string.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<unistd.h>
int main(void){
   int socket_desc,client_sock,client_size;
   struct sockaddr_in server_addr,client_addr;
   char server_message[2000],client_message[2000];

   //clean buffers;
   memset(server_message,'\0',sizeof(server_message));
   memset(client_message,'\0',sizeof(client_message));

   //create socket:
   socket_desc=socket(AF_INET,SOCK_STREAM,0);
   if(socket_desc<0){
      printf("Error while creating a socket\n");
      return -1;
   }
   printf("Socket created successfully\n");

   //set port and IP:
   server_addr.sin_family = AF_INET;
   server_addr.sin_port = htons(2000);
   server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
   //Bind to the set port and IP
   if(bind(socket_desc, (struct sockaddr*)&server_addr,sizeof(server_addr))<0){
      printf("Couldn't bind to the port\n");
      return -1;
   }
   printf("Done with Binding\n");
   //Listen for clients:
   if(listen(socket_desc,1)<0){
      printf("Error while Listening\n");
      return -1;
   }
   printf("\nListening for incoming connections.    \n");

   //Accepting an incoming connection:
   client_size = sizeof(client_addr);
```

```
    client_sock = accept(socket_desc, (struct sockaddr*)&client_addr,&client_size);
    if(client_sock < 0)
    {
        printf("Can't accept\n");
        return -1;
    }
    printf("Client connected to an IP : %s and port :
%i\n",inet_ntoa(client_addr.sin_addr),ntohs(client_addr.sin_port));

    //Receive client's message:
    if(recv(client_sock, client_message, sizeof(client_message), 0) < 0){
        printf("Couldn't connect\n");
        return -1;
    }
    printf("Msg from client : %s\n",client_message);
    //Respond to client
    strcpy(server_message,"This is the server's message.");
    if(send(client_sock,server_message, strlen(server_message), 0) < 0){
        printf("Can't Send\n");
        return -1;
    }

    //closing the socket:
    close(client_sock);
    close(socket_desc);
    return 0;
}
```

TCP_CLIENT

```
#include<stdio.h>
#include<string.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<unistd.h>
int main(void){
   int socket_desc;
   struct sockaddr_in server_addr;
   char server_message[2000], client_message[2000];

   //clean buffers;
   memset(server_message,'\0',sizeof(server_message));
   memset(client_message,'\0',sizeof(client_message));

   //create socket:
   socket_desc=socket(AF_INET,SOCK_STREAM,0);
   if(socket_desc<0){
      printf("Error while creating a socket\n");
      return -1;
   }
   printf("Socket created successfully\n");

   //set port and IP:
   server_addr.sin_family = AF_INET;
   server_addr.sin_port = htons(2000);
   server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

   //send connection request to server:
   if(connect(socket_desc, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0){
      printf("Unable to connect\n");
      return -1;
   }
   printf("Connected with server successfully\n");

   //Get input from the user:
   printf("Enter message: ");
   gets(client_message);

   //send message to server:
   if(send(socket_desc,client_message, strlen(client_message), 0) < 0){
      printf("Unable to send message\n");
```

```
        return -1;
    }

    //Receive the server's response:
    if(recv(socket_desc,server_message,sizeof(server_message), 0) < 0){
        printf("Error while receiving servers message\n");
        return -1;
    }

    printf("Server's response : %s\n",server_message);

    //close the packet:
    close(socket_desc);

    return 0;
}
```

```
aront@Dell-Vostro:~/lab$ gcc server_TCP.c -o tcp_s
aront@Dell-Vostro:~/lab$ ./tcp_s
Socket created successfully
Done with Binding

Listening for incoming connections....
Client connected to an IP : 127.0.0.1 and port : 48354
Msg from client : Hi
```

```
aront@Dell-Vostro:~/lab$ gcc client_TCP.c -o tcp_c
aront@Dell-Vostro:~/lab$ ./tcp_c
Socket created successfully
Connected with server successfully
Enter message: Hi
Server's response : This is the server's message.
```

UDP_SERVER

```
#include <stdio.h>
#include <sys/socket.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <string.h>
int server_fd,new_socket,client_size;
struct sockaddr_in client_addr,server_addr;
char server_message[100],client_message[100];
int main(){

   memset(server_message, '\0',sizeof(server_message));
   memset(client_message, '\0',sizeof(client_message));

   //create a socket for client, store socket file descriptor to client_fd
   if((server_fd=socket(AF_INET,SOCK_DGRAM,IPPROTO_UDP))<0){
      printf("Error while creating socket\n");
      return -1;
   }
   printf("Socket created successfully!\n");

   //set port no, ip address, and family of addresses
   server_addr.sin_family=AF_INET;
   server_addr.sin_port=htons(2001);            //htons - host byte order to network byte order conversion
   server_addr.sin_addr.s_addr=inet_addr("127.0.0.1");

   //bind ip address and port to server socket
   if(bind(server_fd,(struct sockaddr*)&server_addr,sizeof(server_addr))<0){
      printf("Couldn't bind to the port\n");
      return -1;
   }
   printf("Binding done successfully!\n");

   client_size=sizeof(client_addr);
   //receive message from client
   if(recvfrom(server_fd,client_message,sizeof(client_message),0,(struct
sockaddr*)&client_addr,&client_size)<0){
      printf("Could'nt receive message from client\n");
```

```
        return -1;
    }
    printf("Message from client: %s\n",client_message);

    strcpy(server_message,"Recieved!");

    //send response to client
    if(sendto(server_fd,server_message,sizeof(server_message),0,(struct
sockaddr*)&client_addr,client_size)<0){
        printf("Could'nt send response to client\n");
        return -1;
    }
    printf("Response sent!\n");

    close(new_socket);
    close(server_fd);
    }
```

UDP_CLIENT

```
#include <stdio.h>
#include <sys/socket.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <string.h>
int client_fd,new_socket,client_size;
struct sockaddr_in client_addr,server_addr;
char server_message[100],client_message[100];
int main(){
   memset(server_message, '\0',sizeof(server_message));
   memset(client_message, '\0',sizeof(client_message));

   //create a socket for client, store socket file descriptor to client_fd
   if((client_fd=socket(AF_INET,SOCK_DGRAM,IPPROTO_UDP))<0){
      printf("Error while creating socket\n");
      return -1;
   }
   printf("Socket created successfully!\n");

   //set port no, ip address, and family of addresses
   server_addr.sin_family=AF_INET;
   server_addr.sin_port=htons(2001);            //htons - host byte order to network byte order conversion
   server_addr.sin_addr.s_addr=inet_addr("127.0.0.1");

   //connect to server
   if(connect(client_fd,(struct sockaddr*)&server_addr,sizeof(server_addr))<0){
      printf("Could'nt connect to server!\n");
      return -1;
   }
   printf("Connected to server!\n");

   strcpy(client_message,"Hello!");
   client_size=sizeof(client_addr);

   //send message to server
   if(sendto(client_fd,client_message,sizeof(client_message),0,(struct
sockaddr*)&server_addr,client_size)<0){
```

```
    printf("Could'nt send message to server!\n");
    return -1;
  }
  //receive message from server
  if(recvfrom(client_fd,server_message,sizeof(server_message),0,(struct
sockaddr*)&server_addr,&client_size)<0){
    printf("Could'nt receive message from server\n");
    return -1;
  }
  printf("Message from server: %s\n",server_message);

  close(client_fd);
  }
```

```
aront@Dell-Vostro:~/lab$ gcc server_udp.c -o udp_s
.aront@Dell-Vostro:~/lab$ ./udp_s
Socket created successfully!
Binding done successfully!
Message from client: Hello!
Response sent!
```

```
aront@Dell-Vostro:~/lab$ gcc client_udp.c -o udp_c
aront@Dell-Vostro:~/lab$ ./udp_c
Socket created successfully!
Connected to server!
Message from server: Recieved!
```

STOP AND WAIT SENDER

```
#include <sys/socket.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
int socket_desc,new_socket;
struct sockaddr_in receiver_addr,sender_addr;
char buffer[100];
int main(){
   int k=5,m=1,p;
   if((socket_desc=socket(AF_INET,SOCK_STREAM,0))<0){
      printf("Error while creating socket\n");
      return -1;
   }
   printf("Socket created successfully!\n");

   receiver_addr.sin_family=AF_INET;
   receiver_addr.sin_port=htons(2001);
   receiver_addr.sin_addr.s_addr=inet_addr("127.0.0.1");

   if(connect(socket_desc,(struct sockaddr*)&receiver_addr,sizeof(receiver_addr))<0){
      printf("Could'nt connect to reciever!\n");
      return -1;
   }
   printf("Connected to reciever!\n");

   while(k>0){
      //send frame to reciever only if m is divisible by 2
      if(m<=5){
         printf("Sending frame %d\n",m);
      }
      if(m%2==0){
         strcpy(buffer,"frame");
      }
```

```
    //if m not divisible, assume frame is lost in simulation and resend frame after 3 seconds
    else{
        strcpy(buffer,"error");
        printf("Packet loss!\n");
        for(p=1;p<=3;p++){
            printf("Waiting for %d seconds\n",p);
        }
        printf("Retransmitting...\n");
        strcpy(buffer,"frame");
        sleep(3);
    }
    int y= send(socket_desc,buffer,20,0);
    if(y<0){
        printf("Error sending message to reciever!\n");
        return -1;
    }
    else{
        printf("Sent frame: %d\n",m);
    }
    //check if ack is recieved from sender
    int z=recv(socket_desc,buffer,1024,0);
    if(z<0){
        printf("Error in receiving ack from sender");
        return -1;
    }
    if(strncmp(buffer,"ack",3)==0){
        printf("Received ACK for frame %d \n",m);
    }
    else{
        printf("ACK %d not recieved\n",m);
    }
    k--;
    m++;
}
close(socket_desc);
}
```

STOP AND WAIT RECEIVER

```
#include <stdio.h>
#include <sys/socket.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <string.h>
int socket_desc,new_socket,client_size;
struct sockaddr_in sender_addr,reciever_addr;
char buffer[100];
int main(){
   int k=5,m=1,p;
   if((socket_desc=socket(AF_INET,SOCK_STREAM,0))<0){
      printf("Error while creating socket\n");
      return -1;
   }
   printf("Socket created successfully!\n");

   reciever_addr.sin_family=AF_INET;
   reciever_addr.sin_port=htons(2001);
   reciever_addr.sin_addr.s_addr=inet_addr("127.0.0.1");

   if(bind(socket_desc,(struct sockaddr*)&reciever_addr,sizeof(reciever_addr))<0){
      printf("Couldn't bind to the port\n");
      return -1;
   }
   printf("Binding done successfully!\n");

   if(listen(socket_desc,1)<0){
      printf("Error while listening\n");
      return -1;
   }
   printf("Listening for clients\n");

   client_size=sizeof(reciever_addr);
```

```
    if((new_socket=accept(socket_desc,(struct sockaddr*)&sender_addr,&client_size))<0){
        printf("Acception failed!\n");
        return -1;
    }
    printf("Client connected at IP =: %s and port:
%i\n",inet_ntoa(sender_addr.sin_addr),ntohs(sender_addr.sin_port));

    while(k>0){
        //check if frame is recieved from sender
        int y=recv(new_socket,buffer,1024,0);
        if(y<0){
            printf("Error in reveiving message from sender!\n");
        }
        if(strncmp(buffer,"frame",5)==0){
            printf("Received frame %d successfully!\n",m);
        }
        else{
            printf("Frame %d not received!",m);
        }
        //send corresponding ack. WE ASSUME THAT ACK IS ALWAYS SENT. NO LOSS OF ACK
        strcpy(buffer,"ack");
        printf("Sending ack %d\n",m);
        int z= send(new_socket,buffer,20,0);
        if(z<0){
            printf("Error sending ack to reciever!\n");
            return -1;
        }
        k--;
        m++;
    }

    close(new_socket);
    close(socket_desc);
    }
```

```
aront@Dell-Vostro:~/lab$ gcc stop_r.c -o stop_r
aront@Dell-Vostro:~/lab$ ./stop_r
Socket created successfully!
Binding done successfully!
Listening for clients
Client connected at IP =: 127.0.0.1 and port: 47658
Received frame 1 successfully!
Sending ack 1
Received frame 2 successfully!
Sending ack 2
Received frame 3 successfully!
Sending ack 3
Received frame 4 successfully!
Sending ack 4
Received frame 5 successfully!
Sending ack 5
```

```
aront@Dell-Vostro:~/lab$ gcc stop_s.c -o stop_s
aront@Dell-Vostro:~/lab$ ./stop_s
Socket created successfully!
Connected to reciever!
Sending frame 1
Packet loss!
Waiting for 1 seconds
Waiting for 2 seconds
Waiting for 3 seconds
Retransmitting...
Sent frame: 1
Received ACK for frame 1
Sending frame 2
Sent frame: 2
Received ACK for frame 2
Sending frame 3
Packet loss!
Waiting for 1 seconds
Waiting for 2 seconds
Waiting for 3 seconds
Retransmitting...
Sent frame: 3
Received ACK for frame 3
Sending frame 4
Sent frame: 4
Received ACK for frame 4
Sending frame 5
Packet loss!
Waiting for 1 seconds
Waiting for 2 seconds
Waiting for 3 seconds
Retransmitting...
Sent frame: 5
Received ACK for frame 5
```

GO_BACK_N_SERVER

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <sys/time.h>
#include <unistd.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
struct timeval timeout;
void func(int connfd)
{
   char buff[MAX];
   int f, c, ack, next = 0;
   while (1)
   {
      sleep(1);
      bzero(buff, MAX);
      recv(connfd, buff, MAX, 0);
      if (strcmp("Exit", buff) == 0)
      {
         printf("Exit\n");
         break;
      }
      f = atoi(buff);
      if (f != next)
      {
         printf("Frame %d discarded\n", f); bzero(buff, MAX);
         snprintf(buff, sizeof(buff), "%d", ack);
         send(connfd, buff, sizeof(buff), 0);
         continue;
      }
      c = rand() % 3;
      switch (c)
```

```
      {
         case 0:
            // printf("Frame %d not received\n",f);
            break;
         case 1:
            ack = f;
            sleep(2)
            ;
            printf("Frame %d received\nAcknowledement sent: %d\n", f, ack);
            bzero(buff, MAX);
            snprintf(buff, sizeof(buff), "%d", ack);
            send(connfd, buff, sizeof(buff), 0);
            next = ack + 1;
            break;
         case 2:
            ack = f;
            printf("Frame %d received\nAcknowledement sent: %d\n", f, ack);
            bzero(buff, MAX);
            snprintf(buff, sizeof(buff), "%d", ack);
            send(connfd, buff, sizeof(buff), 0);
            next = ack + 1;
            break;
      }
   }
}
void main()
{
   int sockfd, connfd, len;
   struct sockaddr_in servaddr, cli;
   sockfd = socket(AF_INET, SOCK_STREAM, 0);
   if (sockfd == -1)
   {
      printf("Socket creation failed\n");
      exit(0);
   }
   else
      printf("Socket successfully created\n");
   bzero(&servaddr, sizeof(servaddr));
   servaddr.sin_family = AF_INET;
   servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
   servaddr.sin_port = htons(PORT);
   if ((bind(sockfd, (SA *)&servaddr, sizeof(servaddr))) != 0)
   {
      printf("socket bind failed\n");
      exit(0);
```

```
    }
    else
        printf("Socket successfully binded\n");
    if ((listen(sockfd, 5)) != 0)
    {
        printf("Listen failed\n");
        exit(0);
    }
    else
        printf("Server listening\n");
    len = sizeof(cli);
    connfd = accept(sockfd, (SA *)&cli, &len);
    if (connfd < 0)
    {
        printf("Server accept failed\n");
        exit(0);
    }
    else
        printf("Server accept the client\n");
    func(connfd);
    close(sockfd);
}
```

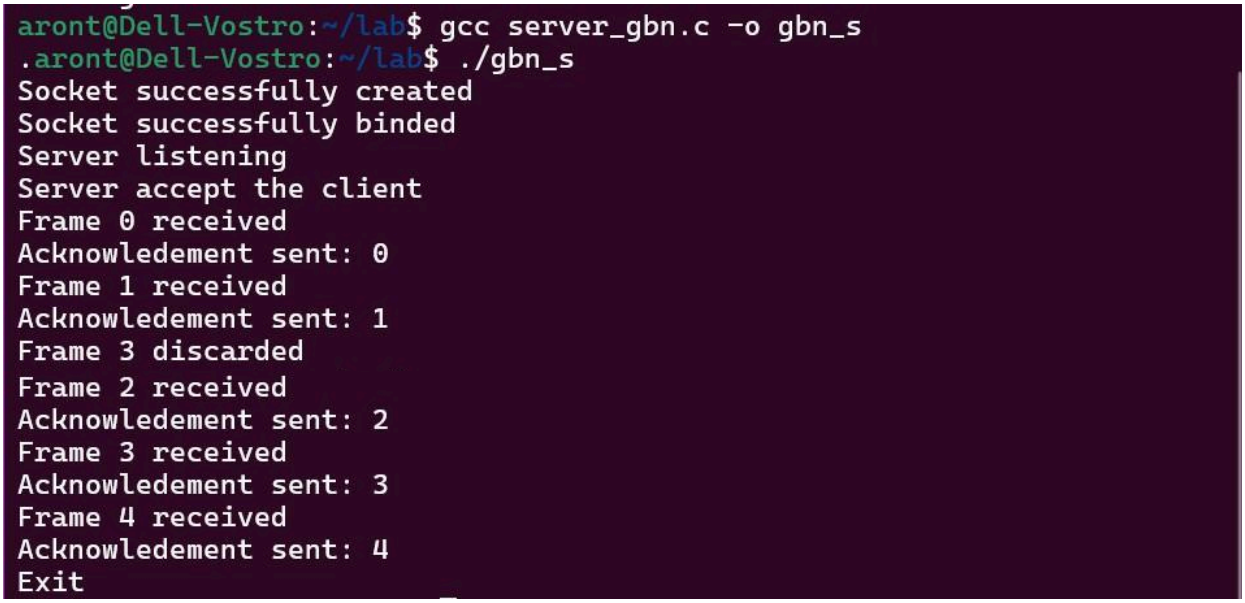
GO_BACK_N_CLIENT

```
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <time.h>
#include <sys/time.h>
#include <unistd.h>
#include <arpa/inet.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
struct timeval timeout;
void func(int sockfd, int nf, int ws)
{
   char buff[MAX];
   int ack, i = 0, n, k, w1 = 0, w2 = ws - 1, j, flag = 0;
   if (setsockopt(sockfd, SOL_SOCKET, SO_RCVTIMEO, (const char *)&timeout, sizeof(timeout)) < 0)
      perror("setsockopt(SO_RCVTIMEO) failed");

   for (i = 0; i < nf && i <= w2; i++)
   {
      bzero(buff, sizeof(buff));
      snprintf(buff, sizeof(buff), "%d", i);
      k = send(sockfd, buff, sizeof(buff), 0);
      printf("Frame %d sent\n", i);
   }
   while (1)
   {
      if (w2 - w1 != ws - 1 && flag == 0 && i != nf)
      {
         bzero(buff, sizeof(buff));
         snprintf(buff, sizeof(buff), "%d", i);
         k = send(sockfd, buff, sizeof(buff), 0);
         printf("Frame %d sent\n", i);
         w2++;
         i++;
      }
```

```
        flag = 0;
        bzero(buff, sizeof(buff));
        n = recv(sockfd, buff, MAX, 0);
        ack = atoi(buff);
        if (n > 0)
        {
            if (ack + 1 == nf)
            {
                printf("Acknowlegement received: %d\nExit\n", ack);
                bzero(buff, sizeof(buff));
                strcpy(buff, "Exit");
                k = send(sockfd, buff, sizeof(buff), 0);
                break;
            }
            if (ack == w1)
            {
                w1++;
                printf("Acknowlegement received: %d\n", ack);
            }
        }
        else
        {
            printf("Acknowledgement not received for %d\nResending frames\n", w1);
            for (j = w1; j < nf && j < w1 + ws; j++)
            {
                bzero(buff, sizeof(buff));
                snprintf(buff, sizeof(buff), "%d", j);
                k = send(sockfd, buff, sizeof(buff), 0);
                printf("Frame %d sent\n", j);
            }
            flag = 1;
        }
    }
}
void main()
{
    int sockfd, connfd, f, w;
    struct sockaddr_in servaddr, cli;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1)
    {
        printf("Socket creation failed\n");
        exit(0);
    }
```

```
    else
        printf("Socket successfully created\n");
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);
    timeout.tv_sec = 3;
    timeout.tv_usec = 0;
    if (connect(sockfd, (SA *)&servaddr, sizeof(servaddr)) != 0)
    {
        printf("Connection with the server failed\n");
        exit(0);
    }
    else
        printf("Connected to the server\n");
    printf("Enter the number of frames: ");
    scanf("%d", &f);
    printf("Enter the window size: ");
    scanf("%d", &w);
    func(sockfd, f, w);
    close(sockfd);
}
```



```
aront@Dell-Vostro:~/lab$ gcc client_gbn.c -o gbn_c
aront@Dell-Vostro:~/lab$ ./gbn_c
Socket successfully created
Connected to the server
Enter the number of frames: 5
Enter the window size: 2
Frame 0 sent
Frame 1 sent
Acknowlegement received: 0
Frame 2 sent
Acknowlegement received: 1
Frame 3 sent
Acknowledgement not received for 2
Resending frames
Frame 2 sent
Frame 3 sent
Acknowlegement received: 2
Frame 4 sent
Acknowlegement received: 3
Acknowlegement received: 4
Exit
```