Muhammad Muntazer Mehdi

**TCP/UDP**

**Client-Server**

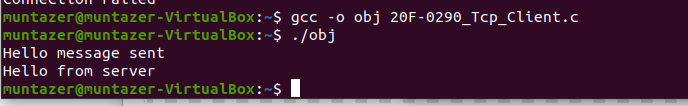
**Task 1:**

**TCP:**

1. **SERVER.C:**

#include <netinet/in.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <sys/socket.h>  
#include <unistd.h>  
#define PORT 8080 //define port number  
int main(int argc, char const\* argv[])  
{  
int server\_fd, new\_socket, valread;  
struct sockaddr\_in address; //used to define socket address  
int opt = 1;  
int addrlen = sizeof(address); //stores the size of socket address  
char buffer[1024] = { 0 };//declares a buffer for storing message  
char\* hello = "Hello from server";  
  
// Creating socket file descriptor  
if ((server\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {  
perror("socket failed");//display error in case of failure  
exit(EXIT\_FAILURE);//exit  
}  
  
// Forcefully attaching socket to the port 8080  
  
if (setsockopt(server\_fd, SOL\_SOCKET,  
  
SO\_REUSEADDR | SO\_REUSEPORT, &opt,  
sizeof(opt))) {  
  
perror("setsockopt");//controls the behaviour of socket  
exit(EXIT\_FAILURE);  
}  
address.sin\_family = AF\_INET;//refereing to address from internet that requires a pair of host,  
port.  
address.sin\_addr.s\_addr = INADDR\_ANY; //binds to a special IP address of INADDR\_ANY  
address.sin\_port = htons(PORT); //translates a short integer from host byte order to network  
byte order  
  
// Forcefully attaching socket to the port 8080  
if (bind(server\_fd, (struct sockaddr\*)&address,  
  
sizeof(address))  
< 0) {  
perror("bind failed");  
exit(EXIT\_FAILURE);  
}  
if (listen(server\_fd, 3) < 0) {  
perror("listen"); //listens to client connection request.  
exit(EXIT\_FAILURE);  
}  
if ((new\_socket  
= accept(server\_fd, (struct sockaddr\*)&address,  
(socklen\_t\*)&addrlen))  
  
< 0) {  
perror("accept");  
exit(EXIT\_FAILURE);  
  
}  
valread = read(new\_socket, buffer, 1024); ////read message from client  
printf("%s\n", buffer);  
send(new\_socket, hello, strlen(hello), 0); //system call used to sent message to client  
printf("Hello message sent\n");  
  
close(new\_socket); // closing the connected socket  
  
shutdown(server\_fd, SHUT\_RDWR); // closing the listening socket  
return 0;  
}

1. **CLIENT.C:**
2. #include <arpa/inet.h>  
   #include <stdio.h>  
   #include <string.h>  
   #include <sys/socket.h>  
   #include <unistd.h>  
   #define PORT 8080 //define port number  
     
   int main(int argc, char const\* argv[])  
   {  
   int sock = 0, valread, client\_fd;  
   struct sockaddr\_in serv\_addr;//used to define server address  
   char\* hello = "Hello from client";  
   char buffer[1024] = { 0 };//declares a buffer for storing message  
     
   if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {  
   printf("\n Socket creation error \n");  
   return -1;  
   }  
     
   serv\_addr.sin\_family = AF\_INET;//refereing to address from internet that requires a pair of host,  
   port.  
   serv\_addr.sin\_port = htons(PORT);//translates a short integer from host byte order to network  
   byte order  
     
   // Convert IPv4 and IPv6 addresses from text to binary  
   // form  
   if (inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr)<= 0) {  
   printf(  
   "\nInvalid address/ Address not supported \n");  
   return -1;  
   }  
     
   //check for connection failure  
   if ((client\_fd = connect(sock, (struct sockaddr\*)&serv\_addr,sizeof(serv\_addr)))< 0) {  
   printf("\nConnection Failed \n");  
   return -1;  
   }  
   send(sock, hello, strlen(hello), 0);//sending message to client using socket  
   printf("Hello message sent\n");  
   valread = read(sock, buffer, 1024);//read message from server  
   printf("%s\n", buffer);  
     
   // closing the connected socket  
     
   close(client\_fd);  
   return 0;  
   }

****

**A picture containing text, screenshot, font

Description automatically generated**

**Task 2:**

**UDP:**

1. **SERVER.C:**

#include <stdio.h>  
#include <stdlib.h>  
#include <unistd.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/socket.h>  
#include <arpa/inet.h> //internet address conversion  
#include <netinet/in.h> //used for internet address family  
#define PORT 8080 //port number defined for communication  
#define MAXLINE 1024 //max length defined for message  
int main()  
{  
int sockfd;  
char buffer[MAXLINE]; //buffer for storing messages  
char \*hello = "Hello from server"; //client will receive this message  
struct sockaddr\_in servaddr, cliaddr; //address structures for client and  
server  
// setup UDP socket for communication  
if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0)  
{  
perror("Socket Setup Failed.."); //failure error  
  
exit(EXIT\_FAILURE);  
}  
//clear address structures  
memset(&servaddr, 0, sizeof(servaddr));  
memset(&cliaddr, 0, sizeof(cliaddr));  
//address structure setup for server  
servaddr.sin\_family = AF\_INET; //this will use IPv4  
servaddr.sin\_addr.s\_addr = INADDR\_ANY; //this will use server's IP address  
servaddr.sin\_port = htons(PORT); //this will set port number  
//server address is binded to the socket using this line  
if (bind(sockfd, (const struct sockaddr \*)&servaddr, sizeof(servaddr)) < 0)  
{  
perror("bind failed");  
exit(EXIT\_FAILURE);  
}  
int len, n;  
len = sizeof(cliaddr); //client address structure is set and stored in len  
n = recvfrom(sockfd, (char \*)buffer, MAXLINE, MSG\_WAITALL,  
  
(struct sockaddr \*)&cliaddr, &len); //message from a  
  
client is received  
buffer[n] = '\0'; //this will terminate the message  
printf("Client: %s\n", buffer);  
sendto(sockfd, (const char \*)hello, strlen(hello), MSG\_CONFIRM,  
(const struct sockaddr \*)&cliaddr, len); //message is sent back to  
  
the client  
printf("Hello message sent.\n");  
  
close(sockfd); //socket is closed  
return 0;  
}

1. **CLIENT.C:**

#include <stdio.h>  
#include <stdlib.h>  
#include <unistd.h>  
#include <string.h>  
#include <sys/types.h>  
#include <sys/socket.h>  
#include <arpa/inet.h> //internet address conversion  
#include <netinet/in.h> //used for internet address family  
#define PORT 8080 //port number defined for communication  
#define MAXLINE 1024 //max length defined for message  
int main()  
{  
int sockfd;  
char buffer[MAXLINE]; //buffer for storing messages  
char \*hello = "Hello from server"; //client will receive this message  
struct sockaddr\_in servaddr, cliaddr; //address structures for client and  
server  
//setup UDP socket for communication  
if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0) {  
perror("socket creation failed"); //failure error  
exit(EXIT\_FAILURE);  
}  
// start a server address structure  
memset(&servaddr, 0, sizeof(servaddr));  
memset(&cliaddr, 0, sizeof(cliaddr));  
servaddr.sin\_family = AF\_INET; // for IPv4  
servaddr.sin\_addr.s\_addr = INADDR\_ANY; //This will bind to any IP  
address on local machine  
servaddr.sin\_port = htons(PORT); //This will bind to specified  
port number  
//server address is binded to the socket using this line  
if (bind(sockfd, (const struct sockaddr \*)&servaddr, sizeof(servaddr)) <  
0) {  
perror("bind failed");  
exit(EXIT\_FAILURE);  
}  
int len, n;  
len = sizeof(cliaddr); //client address structure is set and stored  
in len  
n = recvfrom(sockfd, (char \*)buffer, MAXLINE, MSG\_WAITALL, (struct  
sockaddr \*)&cliaddr, &len); //client has received the message  
  
buffer[n] = '\0'; //this will terminate the message  
printf("Client : %s\n", buffer);  
// this will Send message to client  
sendto(sockfd, (const char \*)hello, strlen(hello), MSG\_CONFIRM, (const  
struct sockaddr \*)&cliaddr, len);  
printf("Hello message sent.\n");  
  
close(sockfd); //socket is closed  
return 0;  
}



