Computer Networks

-supervised by:

Prof. Saikat Basu, and

Prof. Subhanjan Sarkar

Rohit Das

B. Tech(Computer Sc. and Engg)

Roll: 30000114022

Regn. No.:143000110023

6th Semester,2016

Maulana Abul Kalam Azad University of Technology,

West Bengal.

LaTeX 2017

Contents

1. Date: 14/4/2017

1..1 Write a C program to implement chat server and client using TCP.

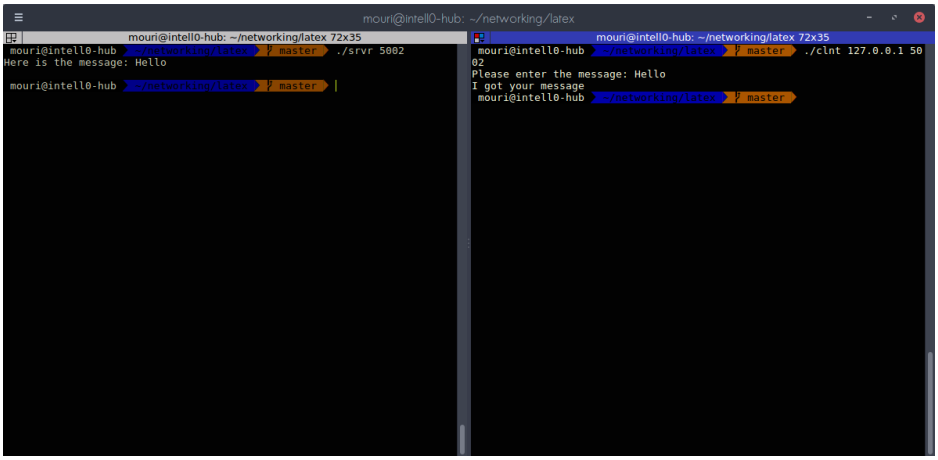
Program:

```
/* echoServer.c */

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <strings.h>
#include <unistd.h>
void error(char *msg)
{
    perror(msg);        exit(1);
}
int main(int argc, char *argv[])
{
    int sockfd, newsockfd, portno, clilen;     char buffer[256];
    struct sockaddr_in serv_addr, cli_addr;    int n;
    if (argc < 2) {
        fprintf(stderr,"ERROR, no port provided\n");  exit(1);
    }
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)    error("ERROR opening socket");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = atoi(argv[1]);
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
             sizeof(serv_addr)) < 0)    error("ERROR on binding");
    listen(sockfd,5);    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
    if (newsockfd < 0)    error("ERROR on accept");
    bzero(buffer,256);    n = read(newsockfd,buffer,255);
    if (n < 0) error("ERROR reading from socket");
    printf("Here is the message: %s\n",buffer);
    n = write(newsockfd,"I got your message",18);
    if (n < 0) error("ERROR writing to socket");
    return 0;
}
```

```
/* echoClient.c */

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
```

```c
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
void error(char *msg)
{
    perror(msg);    exit(0);
}
int main(int argc, char *argv[])
{
    int sockfd, portno, n;     struct sockaddr_in serv_addr;
    struct hostent *server;    char buffer[256];
    if (argc < 3) {
        fprintf(stderr,"usage %s hostname port\n", argv[0]);  exit(0);
    }
    portno = atoi(argv[2]);  sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)    error("ERROR opening socket");
    server = gethostbyname(argv[1]);
    if (server == NULL) {
        fprintf(stderr,"ERROR, no such host\n");  exit(0);
    }
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr, (char *)&serv_addr.sin_addr.s_addr,
          server->h_length);
    serv_addr.sin_port = htons(portno);
    if (connect(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr))
          < 0)    error("ERROR connecting");
    printf("Please enter the message: ");
    bzero(buffer,256);    fgets(buffer,255,stdin);
    n = write(sockfd,buffer,strlen(buffer));
    if (n < 0)    error("ERROR writing to socket");
    bzero(buffer,256);    n = read(sockfd,buffer,255);
    if (n < 0)    error("ERROR reading from socket");
    printf("%s\n",buffer);
    return 0;
}
```

Output:



2. Date: 21/4/2017

2..1 Write a program to implement chat server and client using UDP.

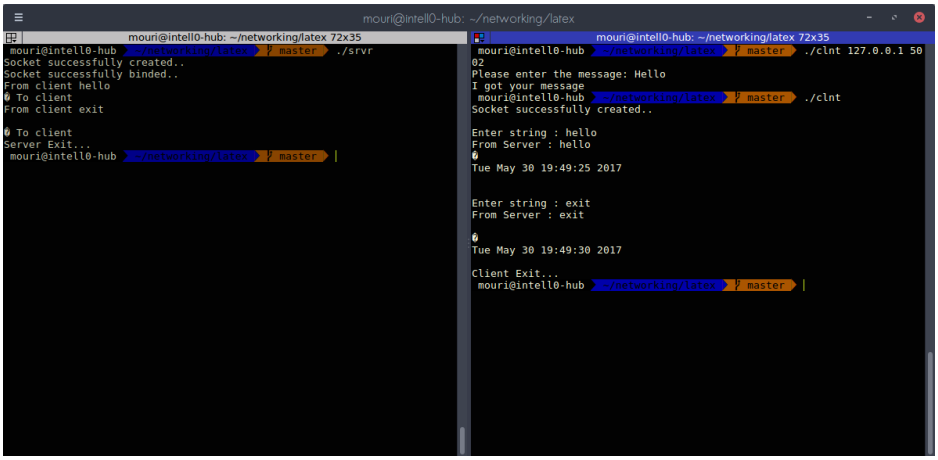
Program:

```
//udpserver.c

#include <stdio.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h> /* For close()*/
#define MAX 80
#define PORT 43454
#define SA struct sockaddr
/*
 * listen function: To listen from client
 */
void listen_client(int sockfd)
{
    char buff[MAX];    int n,clen;
    struct sockaddr_in cli;    clen = sizeof(cli);
    for(;;) {
        bzero(buff,MAX);
        recvfrom(sockfd,buff,sizeof(buff),0,(SA *)&cli,&clen);
        printf("From client %s To client\n",buff);  n = 0;
        //while ((buff[n++]=getchar()) != '\n');
        sendto(sockfd,buff,sizeof(buff),0,(SA *)&cli,clen);
        if(strncmp("exit",buff,4) == 0)
        {
            printf("Server Exit...\n");    break;
        }
    }
}
int main()
{
    int sockfd;    struct sockaddr_in servaddr;
    sockfd=socket(AF_INET,SOCK_DGRAM,0);
    if(sockfd == -1) {
        printf("socket creation failed...\n");    exit(1);
    }
    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(1);
    }
```

```
    else   printf("Socket successfully binded..\n");
    listen_client(sockfd);   close(sockfd);
}
```

```
//udpclient.c

#include <sys/socket.h>
#include <netdb.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <time.h>
#define MAX 80
#define PORT 43454
#define SA struct sockaddr
int main()
{
    char buff[MAX];   int sockfd,len,n;
    struct sockaddr_in servaddr;
    sockfd = socket(AF_INET,SOCK_DGRAM,0);
    if(sockfd == -1) {
        printf("socket creation failed...\n");   exit(1);
    }
    else printf("Socket successfully created..\n");
    bzero(&servaddr,sizeof(len));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);
    len = sizeof(servaddr);
    for(;;) {
        printf("\nEnter string : ");
        n = 0;
        while ((buff[n++]=getchar()) != '\n');
        sendto(sockfd,buff,sizeof(buff),0,(SA *)&servaddr,len);
        bzero(buff,sizeof(buff));
        recvfrom(sockfd,buff,sizeof(buff),0,(SA *)&servaddr,&len);
        printf("From Server : %s\n",buff);
        time_t current_time = time(NULL);
        printf("%s\n",ctime(&current_time));
        if(strncmp("exit",buff,4) == 0) {
            printf("Client Exit...\n");  break;
        }
    }
    close(sockfd);
}
```

Output:



3. Date: 28/4/2017

3..1 Write a program to implement Math server and client using TCP.

Program:

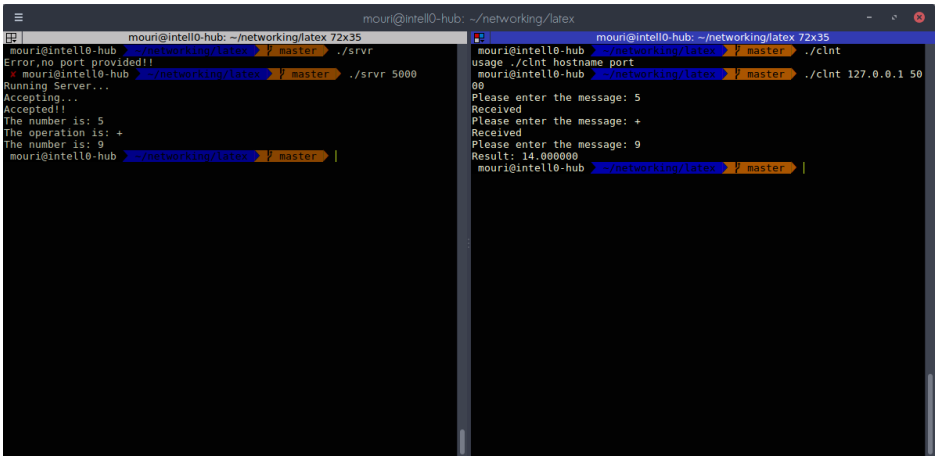
```
//mathserver.c

#include<stdio.h>
#include<stdlib.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <string.h>
#include <unistd.h>
#include <netinet/in.h>
void error(char *msg)
{
    perror(msg);
    exit(1);
}
int stoi(char *str)
{
    int n=0,l=strlen(str),i;
    for(i=0;i<l;i++){
        if(str[i]=='\n')break;    n=(n*10)+(str[i]-'0');
    }
    return n;
}
float calc(int a,int b,char op)
{
    if(op=='+')return a+b;        else if(op=='-')return a-b;
    else if(op=='*')return a*b;   else return ((float)(a*1.0)/b);
}

int main(int argc,char *argv[])
{
    int sockfd,newsockfd,portno,clilen;    char buffer[256];
```

```
    struct sockaddr_in serv_addr,cli_addr;   int n;
    if(argc<2){
        fprintf(stderr,"Error,no port provided!!\n");   exit(1);
    }
    sockfd=socket(AF_INET,SOCK_STREAM,0);
    if(sockfd<0)error("Error opening socket!!");
    bzero((char *)&serv_addr,sizeof(serv_addr));
    portno=atoi(argv[1]);
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=INADDR_ANY;
    serv_addr.sin_port=htons(portno);
    if(bind(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr))<0)
        error("Error on bind!!");
    printf("Running Server...\n");
    listen(sockfd,5);     clilen=sizeof(cli_addr);
    printf("Accepting...\n");
    newsockfd=accept(sockfd,(struct sockaddr *)&cli_addr,&clilen);
    printf("Accepted!!\n");
    if(newsockfd<0)error("Error accepting!!");
    int num,frst,secnd;   float result;   char op;   int i;
    for(i=0;i<3;i++){
        bzero(buffer,256);
        n=read(newsockfd,buffer,256);
        if(n<0)error("Error on read from client!!");
        if(i==0 || i==2){
            num=stoi(buffer);
            printf("The number is: %d \n",num);
            if(i==0)frst=num;     else if(i==2)secnd=num;
        }
        else{
            op=buffer[0];     printf("The operation is: %c \n",op);
        }
        if(i!=2) n=write(newsockfd,"Received",8);
        else{
            result=calc(frst,secnd,op);
            char msg[]="Result: ";
            char final_msg[100];
            sprintf(final_msg,"%s%f",msg,result);
            n=write(newsockfd,final_msg,sizeof(final_msg));
        }
        if(n<0)error("Error writing to socket!!");
    }
    return 0;
}
```

```
//mathclient.c

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
```

```c
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
void error(char *msg)
{
    perror(msg);    exit(0);
}

int main(int argc, char *argv[])
{
    int sockfd, portno, n;
    struct sockaddr_in serv_addr;
    struct hostent *server;
    char buffer[256];
    if (argc < 3) {
        fprintf(stderr,"usage %s hostname port\n", argv[0]);   exit(0);
    }
    portno = atoi(argv[2]);
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)   error("ERROR opening socket");
    server = gethostbyname(argv[1]);
    if (server == NULL) {
        fprintf(stderr,"ERROR, no such host\n");   exit(0);
    }
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr, (char *)&serv_addr.sin_addr.s_addr,
            server->h_length);
    serv_addr.sin_port = htons(portno);
    if (connect(sockfd,(struct sockaddr *)&serv_addr,
                sizeof(serv_addr)) < 0)   error("ERROR connecting");
    for (int i = 0;i < 3; i++) {
        printf("Please enter the message: ");
        bzero(buffer,256);   fgets(buffer,255,stdin);
        n = write(sockfd,buffer,strlen(buffer));
        if (n < 0)   error("ERROR writing to socket");
        bzero(buffer,256);   n = read(sockfd,buffer,255);
        if (n < 0)   error("ERROR reading from socket");
        printf("%s\n",buffer);
    }
    return 0;
}
```

Output:



.

3..2 Write a program to implement day-time chat server and client using TCP.

Program:

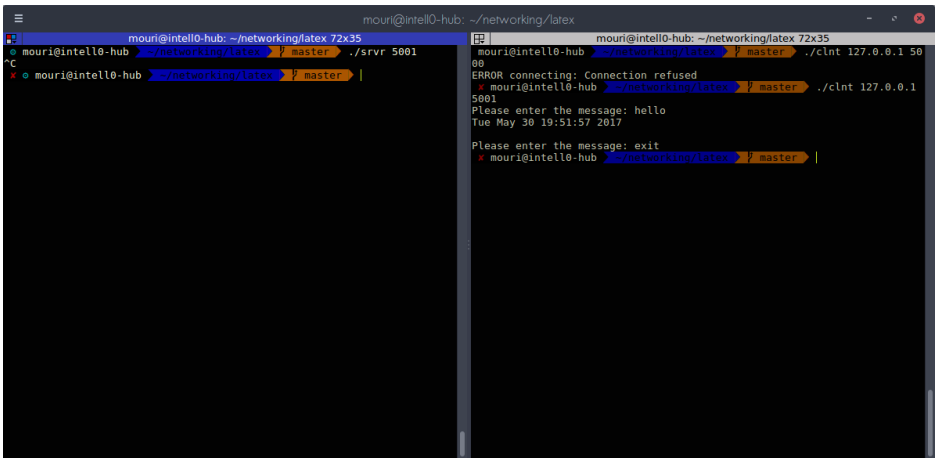
```
//day-time-server.c

#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <unistd.h>

int main(int argc, char **argv)
{
    int listenfd, connfd;
    int port = atoi(argv[1]);

    struct sockaddr_in servaddr;
    char buff[1000];
    time_t ticks;
    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, 8);
    for (;;) {
        connfd = accept(listenfd, (struct sockaddr *) NULL, NULL);
        ticks = time(NULL);
        snprintf(buff, sizeof(buff), "%.24s\r\n", ctime(&ticks));
        write(connfd, buff, strlen(buff));
```

```
        close(connfd);
    }
}
```

```
//day-time-client.c

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

void error(char *msg)
{
    perror(msg);
    exit(1);
}
int main(int argc, char *argv[])
{
    int sockfd, portno, n;
    struct sockaddr_in serv_addr;
    struct hostent *server;
    char buffer[256];
    if (argc < 3) {
        fprintf(stderr,"usage %s hostname port\n", argv[0]);
        exit(0);
    }
    portno = atoi(argv[2]);
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) error("ERROR opening socket");
    server = gethostbyname(argv[1]);
    if (server == NULL) {
        fprintf(stderr,"ERROR, no such host\n");
        exit(0);
    }
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr, (char *)&serv_addr.sin_addr.s_addr,
          server->h_length);
    serv_addr.sin_port = htons(portno);
    if (connect(sockfd,(struct sockaddr *)&serv_addr,
                sizeof(serv_addr)) < 0)
        error("ERROR connecting");
    while (1) {
        printf("Please enter the message: ");
        bzero(buffer,256);
        fgets(buffer,255,stdin);
        n = write(sockfd,buffer,strlen(buffer));
        if (n < 0)
```

```
            error("ERROR writing to socket");
        bzero(buffer,256);
        n = read(sockfd,buffer,255);
        if (n < 0)
            error("ERROR reading from socket");
        printf("%s\n",buffer);
    }
    return 0;
}
```

Output:



.

4. Date: 5/5/2017

4..1 Write a program to implement concurrent server and client using TCP.

Program:

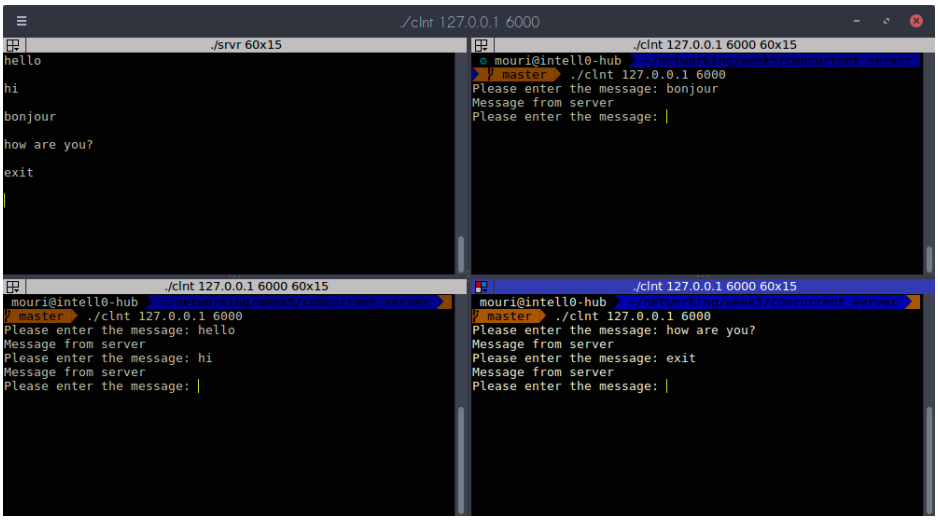
```
//concurrent-server.c

#include <stdio.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <string.h>
int main()
{
    int sockfd, newsockfd;  int clilen;
    struct sockaddr_in cli_addr, serv_addr;    int i;    char buff[100];
    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        printf("Cannot create socket\n");   exit(1);
    }
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(6000); // port: 6000
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
                sizeof(serv_addr)) < 0) {
```

```
        printf("Unable to bind local address\n");   exit(1);
    }
    listen(sockfd, 5); // upto 5 concurrent clients
    while (1) {
        clilen = sizeof(cli_addr);
        newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr,
                &clilen);
        if (newsockfd < 0)   printf("Accept error\n");    exit(1);
        if (fork() == 0) {
            close(sockfd);
            while (1) {
                strcpy(buff, "Message from server");
                send(newsockfd, buff, strlen(buff) + 1, 0);
                for (i = 0;i < 100;i++)   buff[i] = '\0';
                recv(newsockfd, buff, 100, 0);     printf("%s\n",buff);
            }
            close(newsockfd);
            exit(0);
        }
        close(newsockfd);
    }
}
```

```
//concurrent-client.c

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
void error(char *msg)
{
    perror(msg);    exit(0);
}

int main(int argc, char *argv[])
{
    int sockfd, portno, n;    struct sockaddr_in serv_addr;
    struct hostent *server;    char buffer[256];
    if (argc < 3) {
        fprintf(stderr,"usage %s hostname port\n", argv[0]);   exit(0);
    }
    portno = atoi(argv[2]);
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)    error("ERROR opening socket");
    server = gethostbyname(argv[1]);
    if (server == NULL) {
        fprintf(stderr,"ERROR, no such host\n");   exit(0);
    }
```

```
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr, (char *)&serv_addr.sin_addr.s_addr,
          server->h_length);
    serv_addr.sin_port = htons(portno);
    if (connect(sockfd,(struct sockaddr *)&serv_addr,
                sizeof(serv_addr)) < 0)     error("ERROR connecting");
    while (1) {
        printf("Please enter the message: ");
        bzero(buffer,256);    fgets(buffer,255,stdin);
        n = write(sockfd,buffer,strlen(buffer));
        if (n < 0)      error("ERROR writing to socket");
        bzero(buffer,256);    n = read(sockfd,buffer,255);
        if (n < 0)      error("ERROR reading from socket");
        printf("%s\n",buffer);
    }
    return 0;
}
```

Output:



.

5. Date: 12/5/2017

5..1 Write a program to implement file server and client using TCP.

Program:

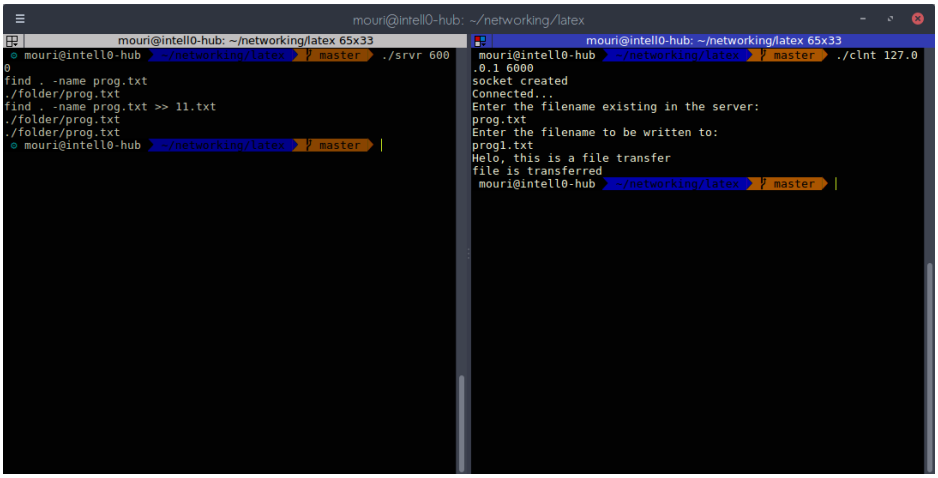
```
//file-server.c

#include<stdio.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<string.h>
#include<stdlib.h>
int main(int argc,char *argv[])
{
    FILE *fp,*fp2;
```

```
    int sockfd,newsockfd,portno,clilen,n,i;     size_t  max = 100;
    char fname[100],name[100],fname1[100],arg[100],arg1[100];
    struct sockaddr_in serv_addr,cli_addr;
    if (argc < 2)
    {
        fprintf(stderr,"ERROR, no port provided\n");    exit(1);
    }
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)    error("ERROR opening socket");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = atoi(argv[1]);
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
             sizeof(serv_addr)) < 0)    error("ERROR on binding");
    listen(sockfd,5);  clilen = sizeof(cli_addr);
    newsockfd=accept(sockfd,(struct sockaddr *) &cli_addr, &clilen);
    if(newsockfd<0)    printf("error on accept\n");
    memset(fname1,'\0',100);     memset(arg,'\0',100);
    memset(arg1,'\0',100);
    n=recv(newsockfd,fname,100,0);     fname[n]='\0';
    strcpy(fname1,"find . -name ");
    strcat(fname1,fname);     printf("%s\n",fname1);
    system(fname1);     strcat(fname1," >> 11.txt");
    printf("%s\n",fname1);     system(fname1);
    system("cat 11.txt");
    fp2=fopen("11.txt","r");     fgets(arg,100,fp2);
    arg[strlen(arg)-1]='\0';     printf("%s\n",arg);
    if(n<0)      printf("error on read");
    else
    {
        fp=fopen(arg,"r"); //read mode
        if(fp==NULL)
        {
            send(newsockfd,"error",5,0);     close(newsockfd);
        }
        else
        {
            while(fgets(name,100,fp))
            {
                if(write(newsockfd,name,100)<0)     printf("can't send\n");
            }
            if(!fgets(name,sizeof(name),fp))  send(newsockfd,"Done",4,0);
            return 0;
        }
    }
}
```

```
//file-client.c

#include<stdio.h>
```

```c
#include<stdlib.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<stdlib.h>
#include<string.h>
int main(int argc,char *argv[])
{
    FILE *fp;
    int sockfd,newsockfd,portno,r;
    char fname[100],fname1[100],text[100];
    struct sockaddr_in serv_addr;    portno = atoi(argv[2]);
    sockfd=socket(AF_INET,SOCK_STREAM,0);
    if(sockfd<0)
    {
        printf("Error on socket creation\n");   exit(0);
    }
    else   printf("socket created\n");
    serv_addr.sin_family=AF_INET;
    serv_addr.sin_addr.s_addr=inet_addr(argv[1]);
    serv_addr.sin_port=htons(portno);
    if(connect(sockfd,(struct sockaddr*)&serv_addr,sizeof(serv_addr))<0)
    {
        printf("Error in Connection...\n");    exit(0);
    }
    else   printf("Connected...\n");
    printf("Enter the filename existing in the server:\n");
    scanf("%s",fname);
    printf("Enter the filename to be written to:\n");
    scanf("%s",fname1);      fp=fopen(fname1,"w");
    send(sockfd,fname,100,0);
    while(1)
    {
        r=recv(sockfd,text,100,0);   text[r]='\0';
        fprintf(fp,"%s",text);
        if(strcmp(text,"error")==0)    printf("file not available\n");
        if(strcmp(text,"Done")==0)
        {
            printf("file is transferred\n");
            fclose(fp);    close(sockfd);
            break;
        }
        else    fputs(text,stdout);
    }
    return 0;
}
```

Output:



.

6. 19/5/2017

6..1 Write a program to implement multicast server and client.

Program:

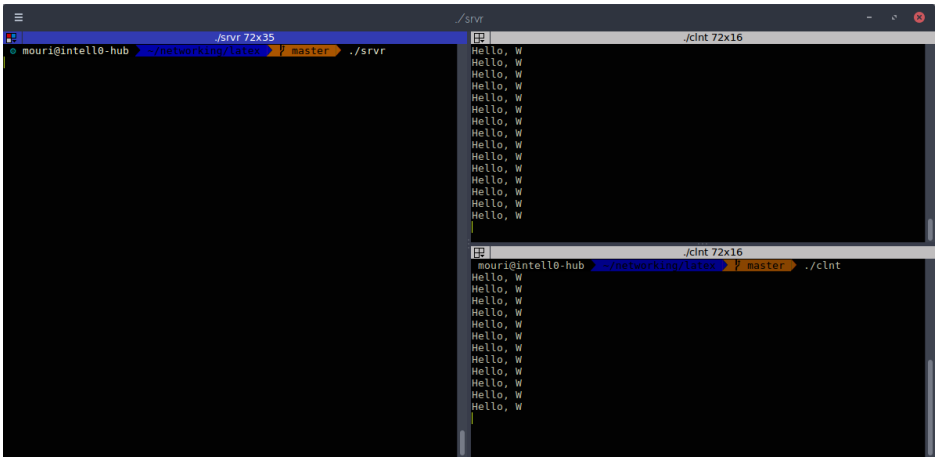
```
//multicast-server.c

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <time.h>
#include <string.h>
#include <stdio.h>
#define HELLO_PORT 12345
#define HELLO_GROUP "225.0.0.37"
int main(int argc, char *argv[])
{
    struct sockaddr_in addr;     int fd, cnt;
    struct ip_mreq mreq;     char *message="Hello, World!";
    /* create what looks like an ordinary UDP socket */
    if ((fd=socket(AF_INET,SOCK_DGRAM,0)) < 0) {
        perror("socket");    exit(1);
    }
    /* set up destination address */
    memset(&addr,0,sizeof(addr));
    addr.sin_family=AF_INET;
    addr.sin_addr.s_addr=inet_addr(HELLO_GROUP);
    addr.sin_port=htons(HELLO_PORT);
    /* now just sendto() our destination! */
    while (1) {
        if (sendto(fd,message,sizeof(message),0,(struct sockaddr *)
                   &addr,sizeof(addr)) < 0) {
            perror("sendto");     exit(1);
        }
        sleep(1);
```

```
    }
}
```

```
//multicast-client.c

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <time.h>
#include <string.h>
#include <stdio.h>
#define HELLO_PORT 12345
#define HELLO_GROUP "225.0.0.37"
#define MSGBUFSIZE 256
int main(int argc, char *argv[])
{
    struct sockaddr_in addr;    int fd, nbytes,addrlen;
    struct ip_mreq mreq;    char msgbuf[MSGBUFSIZE];
    u_int yes=1;            /*** MODIFICATION TO ORIGINAL */
    /* create what looks like an ordinary UDP socket */
    if ((fd=socket(AF_INET,SOCK_DGRAM,0)) < 0) {
        perror("socket");   exit(1);
    }
    /**** MODIFICATION TO ORIGINAL */
    /* allow multiple sockets to use the same PORT number */
    if (setsockopt(fd,SOL_SOCKET,SO_REUSEADDR,&yes,sizeof(yes)) < 0) {
        perror("Reusing ADDR failed");   exit(1);
    }
    /*** END OF MODIFICATION TO ORIGINAL */

    /* set up destination address */
    memset(&addr,0,sizeof(addr));
    addr.sin_family=AF_INET;
    addr.sin_addr.s_addr=htonl(INADDR_ANY); //N.B.: differs from sender
    addr.sin_port=htons(HELLO_PORT);
    /* bind to receive address */
    if (bind(fd,(struct sockaddr *) &addr,sizeof(addr)) < 0) {
        perror("bind");   exit(1);
    }
    //use setsockopt() to request that the kernel join a multicast group
    mreq.imr_multiaddr.s_addr=inet_addr(HELLO_GROUP);
    mreq.imr_interface.s_addr=htonl(INADDR_ANY);
    if (setsockopt(fd,IPPROTO_IP,IP_ADD_MEMBERSHIP,&mreq,sizeof(mreq))
            < 0) {
        perror("setsockopt");   exit(1);
    }
    /* now just enter a read-print loop */
    while (1) {
        addrlen=sizeof(addr);
        if ((nbytes=recvfrom(fd,msgbuf,MSGBUFSIZE,0,
                        (struct sockaddr *) &addr,&addrlen)) < 0) {
```

```
            perror("recvfrom");    exit(1);
        }
        puts(msgbuf);
    }
}
```

Output:



.

6..2 Write a program to implement broadcast server and client.

Program:

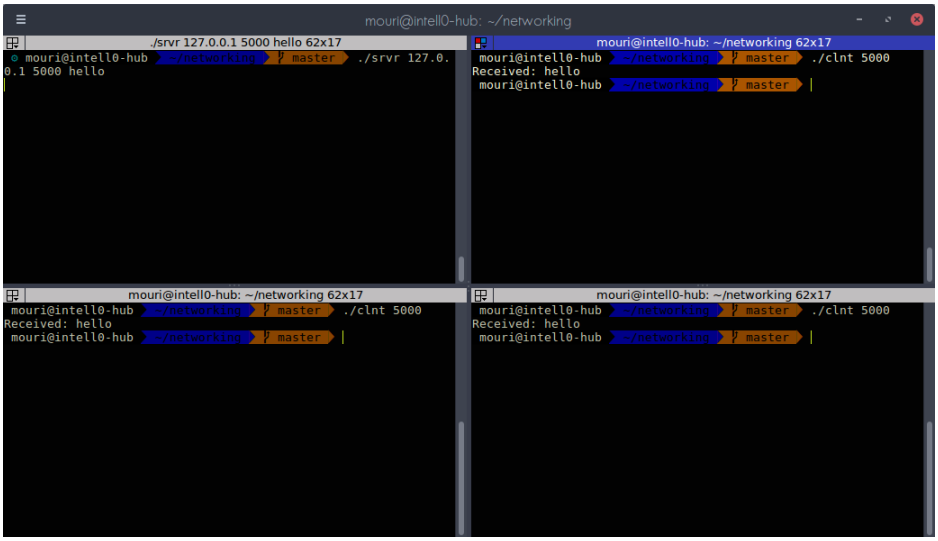
```
//broadcast-server.c

#include <stdio.h>      /* for printf() and fprintf() */
#include <sys/socket.h> /* for socket() and bind() */
#include <arpa/inet.h>  /* for sockaddr_in */
#include <stdlib.h>     /* for atoi() and exit() */
#include <string.h>     /* for memset() */
#include <unistd.h>     /* for close() */

void DieWithError(char *errorMessage);//External error handling function

int main(int argc, char *argv[])
{
    int sock;                        /* Socket */
    struct sockaddr_in broadcastAddr; /* Broadcast address */
    char *broadcastIP;               /* IP broadcast address */
    unsigned short broadcastPort;    /* Server port */
    char *sendString;                /* String to broadcast */
    int broadcastPermission;/*Socket opt to set permission to broadcast*/
    unsigned int sendStringLen;      /* Length of string to broadcast */

    if (argc < 4)/* Test for correct number of parameters */
    {
        fprintf(stderr,"Usage:  %s <IP Address> <Port> <Send String>\n",
                argv[0]);
        exit(1);
    }
```

```
    broadcastIP = argv[1];          /* First arg:  broadcast IP address */
    broadcastPort = atoi(argv[2]);    /* Second arg:  broadcast port */
    sendString = argv[3];           /* Third arg:  string to broadcast */
    /* Create socket for sending/receiving datagrams */
    if ((sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0)
        perror("socket() failed");

    /* Set socket to allow broadcast */
    broadcastPermission = 1;
    if (setsockopt(sock, SOL_SOCKET, SO_BROADCAST,
                (void *) &broadcastPermission,
          sizeof(broadcastPermission)) < 0)
        perror("setsockopt() failed");

    /* Construct local address structure */
    memset(&broadcastAddr, 0, sizeof(broadcastAddr));/*Zero out
                                                    structure*/
    broadcastAddr.sin_family = AF_INET;/*Internet address family*/
    broadcastAddr.sin_addr.s_addr = inet_addr(broadcastIP);/*Broadcast
                                                    IP address*/
    broadcastAddr.sin_port = htons(broadcastPort);/*Broadcast port*/

    sendStringLen = strlen(sendString);/*Find length of sendString*/
    for (;;) /* Run forever */
    {
        /*Broadcast sendString in datagram to clients every 3 seconds*/
        if (sendto(sock, sendString, sendStringLen, 0, (struct sockaddr *)
              &broadcastAddr, sizeof(broadcastAddr)) != sendStringLen)
            perror("sendto() sent a different number of bytes than expected");

        sleep(3);   /* Avoids flooding the network */
    }
    /* NOT REACHED */
}
```

```
//broadcast-client.c

#include <stdio.h>      /* for printf() and fprintf() */
#include <sys/socket.h> /* for socket(), connect(), sendto(), and
                           recvfrom() */
#include <arpa/inet.h>  /* for sockaddr_in and inet_addr() */
#include <stdlib.h>     /* for atoi() and exit() */
#include <string.h>     /* for memset() */
#include <unistd.h>     /* for close() */
#define MAXRECVSTRING 255  /* Longest string to receive */

void DieWithError(char *errorMessage);//External error handling function

int main(int argc, char *argv[])
{
    int sock;                         /* Socket */
    struct sockaddr_in broadcastAddr; /* Broadcast Address */
```

```
    unsigned short broadcastPort;      /* Port */
    char recvString[MAXRECVSTRING+1]; /* Buffer for received string */
    int recvStringLen;                /* Length of received string */
    if (argc != 2)    /* Test for correct number of arguments */
    {
        fprintf(stderr,"Usage: %s <Broadcast Port>\n", argv[0]);     exit(1);
    }
    broadcastPort = atoi(argv[1]);    /* First arg: broadcast port */
    /* Create a best-effort datagram socket using UDP */
    if ((sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP)) < 0)
        perror("socket() failed");
    /* Construct bind structure */
    memset(&broadcastAddr, 0, sizeof(broadcastAddr));/*Zero out structure
    */
    broadcastAddr.sin_family = AF_INET;/*Internet address family*/
    broadcastAddr.sin_addr.s_addr = htonl(INADDR_ANY);/*Any incoming
                                                        interface*/
    broadcastAddr.sin_port = htons(broadcastPort);/*Broadcast port*/
    /* Bind to the broadcast port */
    if (bind(sock, (struct sockaddr *) &broadcastAddr,
                sizeof(broadcastAddr)) < 0)
        perror("bind() failed");
    /* Receive a single datagram from the server */
    if ((recvStringLen = recvfrom(sock, recvString, MAXRECVSTRING, 0,
                    NULL, 0)) < 0)
        perror("recvfrom() failed");
    recvString[recvStringLen] = '\0';
    printf("Received: %s\n", recvString);/*Print the received string*/
    close(sock);     exit(0);
}
```

Output:



.