CSE 2005 -OPERATING SYSTEM

PROJECT REPORT

TWO WAY COMMUNICATION BETWEEN CLIENTS THROUGH SERVER

SUBMITTED BY:

1. NAREN ADITHYA (15BCE0124)
2. GUNEET KAUR BHATIA(15BCE0004)

SUBMITTED TO:

PROF. V VIJAYASHERELY ,SCOPE

VIT®

UNIVERSITY

(Estd. u/s 3 of UGC Act 1956)

VELLORE ■ CHENNAI

www.vit.ac.in

INTRODUCTION

A network socket is one endpoint in a communication flow between two programs running over a network. Sockets are created and used with a set of programming requests or "function calls" sometimes called the sockets application programming interface (API). Sockets can also be used for communication between processes within the same computer or communication for process interacting with socket on other system over network.

This is the typical sequence of sockets requests from a server application in the connectionless context of the Internet in which a server handles many client requests and does not maintain a connection longer than the serving of the immediate request:

```
socket()
|
bind()
|
recvfrom()
|
(wait for a sendto request from some client)
|
(process the sendto request)
|
sendto (in reply to the request from the client)
```

A corresponding client sequence of sockets requests would be:

```
 socket()
|
bind()
|
sendto()
|
recvfrom()
```

In our project, we have two clients (A,B) and a server. First, socket is created between client A -server and client B-server. Two way communications between clients is through server.

- A writes to server
- Server writes to B while B reads from server
- B writes to server while server reads from B.
- Server writes to A while A reads from server.

For the two clients to talk to each other accept() is used twice before fork() processes, so both processes know each other.

PROBLEM:

TWO-WAY COMMUNICATION BETWEEN TWO CLIENTS THROUGH SERVER.

The two clients can simultaneously receive and send messages to each other.

SOLUTION:

After creating socket between client A –server and client B –server , fork() system call is used which creates a child process of the main program. Since, accept() is used twice in the main function before fork(), both the processes know each other. Using close() system call, the main function stops listening to new connections but the child process can listen. The main function handles the connected clients.

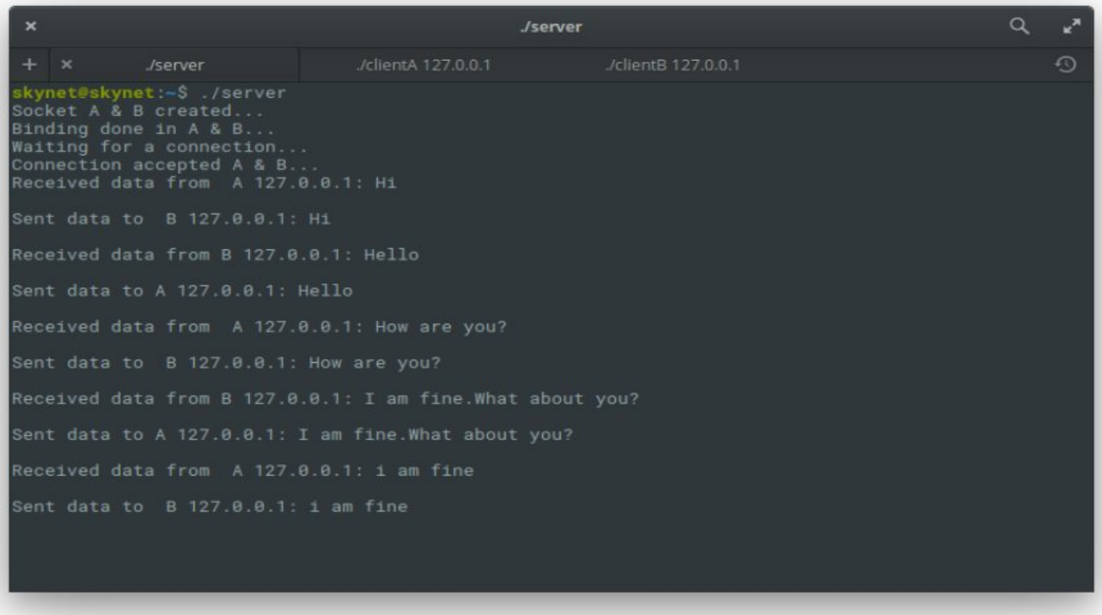
IMPLEMENTATION

1. SERVER

```
#include"stdio.h"
#include"stdlib.h"
#include"sys/types.h"
#include"sys/socket.h"
#include"string.h"
#include"unistd.h"
#include"arpa/inet.h"
#include"netinet/in.h"

#define PORTA 4444
#define PORTB 5555
#define BUF_SIZE 2000
#define CLADDR_LEN 100

void main() {
 struct sockaddr_in addrA,addrB, cl_addrA,cl_addrB;
 int sockfdA, lenA, retA, newsockfdA;
 int sockfdB, lenB, retB, newsockfdB;
 char bufferA[BUF_SIZE];
 char bufferB[BUF_SIZE];
 pid_t childpidA,childpidB;
 char clientAddrA[CLADDR_LEN];
 char clientAddrB[CLADDR_LEN];

 sockfdA = socket(AF_INET, SOCK_STREAM, 0);
 sockfdB = socket(AF_INET, SOCK_STREAM, 0);
 if (sockfdA < 0) {
  printf("Error creating socketA!\n");
  exit(1);
 }
if (sockfdB < 0) {
  printf("Error creating socketB!\n");
  exit(1);
 }
 printf("Socket A & B created...\n");

 memset(&addrA, 0, sizeof(addrA));
 memset(&addrB, 0, sizeof(addrB));
 addrA.sin_family = AF_INET;
 addrA.sin_addr.s_addr = INADDR_ANY;
 addrA.sin_port = PORTA;
 addrB.sin_family = AF_INET;
 addrB.sin_addr.s_addr = INADDR_ANY;
 addrB.sin_port = PORTB;

 retA = bind(sockfdA, (struct sockaddr *) &addrA, sizeof(addrA));
 retB = bind(sockfdB, (struct sockaddr *) &addrB, sizeof(addrB));
 if (retA < 0) {
  printf("Error binding in A!\n");
  exit(1);
 }
```

```
if (retB < 0) {
  printf("Error binding in B!\n");
  exit(1);
 }
 printf("Binding done in A & B...\n");

 printf("Waiting for a connection...\n");
 listen(sockfdA, 5);
 listen(sockfdB, 5);
  lenA = sizeof(cl_addrA);
  lenB = sizeof(cl_addrB);
  newsockfdA = accept(sockfdA, (struct sockaddr *) &cl_addrA, &lenA);
  newsockfdB = accept(sockfdB, (struct sockaddr *) &cl_addrB, &lenB);
  if (newsockfdA < 0) {
   printf("Error accepting connection!\n");
   exit(1);
  }
 if (newsockfdB < 0) {
   printf("Error accepting connection!\n");
   exit(1);
  }
  printf("Connection accepted A & B...\n");

  inet_ntop(AF_INET, &(cl_addrA.sin_addr), clientAddrA, CLADDR_LEN);
  inet_ntop(AF_INET, &(cl_addrB.sin_addr), clientAddrB, CLADDR_LEN);
  childpidA = fork();
//) == 0) { for (;;){
    if (childpidA > 0)
{
memset(bufferA, 0, BUF_SIZE);
memset(bufferB, 0, BUF_SIZE);
retA = recvfrom(newsockfdA, bufferA, BUF_SIZE, 0, (struct sockaddr *) &cl_addrA,
&lenA);
if(retA < 0) {

     printf("Error receiving data!\n");

     exit(1);

    }

    printf("Received data from  A %s: %s\n", clientAddrA, bufferA);
retB = sendto(newsockfdB, bufferA, BUF_SIZE, 0, (struct sockaddr *) &cl_addrB, lenB);
 if (retB < 0) {

     printf("Error sending data!\n");

     exit(1);

    }

    printf("Sent data to  B %s: %s\n", clientAddrB, bufferA);
}

else
```

```
{
    memset(bufferA, 0, BUF_SIZE);
    memset(bufferB, 0, BUF_SIZE);

    retB = recvfrom(newsockfdB, bufferB, BUF_SIZE, 0, (struct sockaddr *) &cl_addrB,
&lenB);

    if(retB < 0) {
     printf("Error receiving data!\n");
     exit(1);
    }
    printf("Received data from B %s: %s\n", clientAddrB, bufferB);
   retA = sendto(newsockfdA, bufferB, BUF_SIZE, 0, (struct sockaddr *) &cl_addrA, lenA);
       if (retA < 0) {
     printf("Error sending data!\n");
     exit(1);
    }
    printf("Sent data to A %s: %s\n", clientAddrA, bufferB);

   }
}
//  }
  close(newsockfdA);
  close(newsockfdB);
 }
```

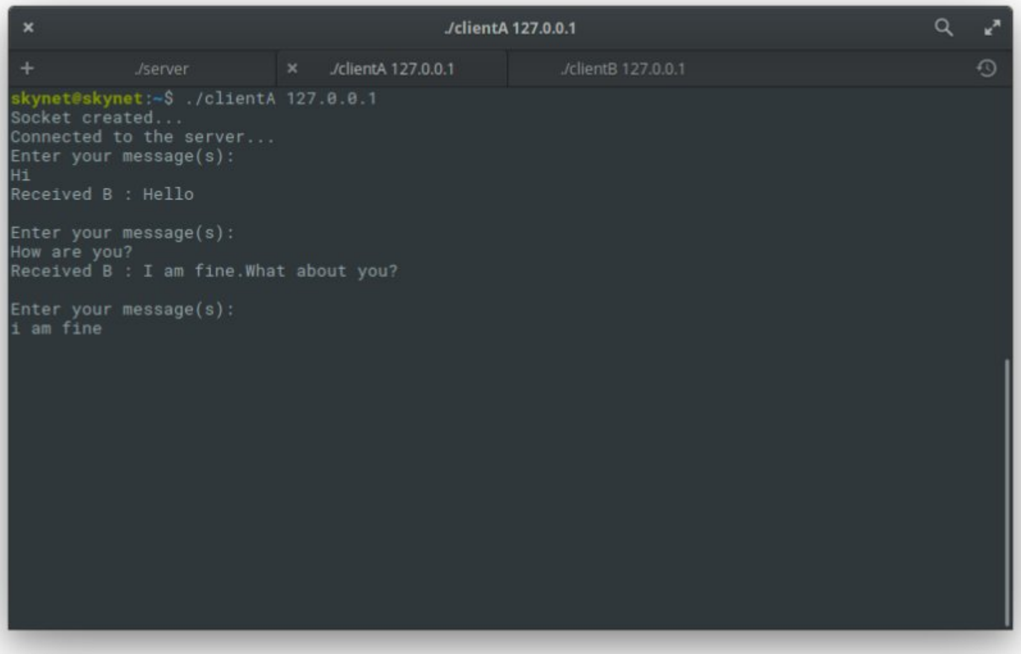


```
skynet@skynet:~$ ./server
Socket A & B created...
Binding done in A & B...
Waiting for a connection...
Connection accepted A & B...
Received data from  A 127.0.0.1: Hi

Sent data to  B 127.0.0.1: Hi

Received data from B 127.0.0.1: Hello

Sent data to A 127.0.0.1: Hello

Received data from  A 127.0.0.1: How are you?

Sent data to  B 127.0.0.1: How are you?

Received data from B 127.0.0.1: I am fine.What about you?

Sent data to A 127.0.0.1: I am fine.What about you?

Received data from  A 127.0.0.1: i am fine

Sent data to  B 127.0.0.1: i am fine
```

2. CLIENT A

```
#include"stdio.h"

#include"stdlib.h"

#include"sys/types.h"

#include"sys/socket.h"

#include"string.h"

#include"netinet/in.h"

#include"netdb.h"

#include"arpa/inet.h"

#include"unistd.h"


#define PORT 4444

#define BUF_SIZE 2000

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

 struct sockaddr_in addr, cl_addr;

 int sockfd, ret;

 pid_t re_child;

 char buffer[BUF_SIZE];

 struct hostent * server;

 char * serverAddr;


 if (argc < 2) {

  printf("usage: client < ip address >\n");

  exit(1);

 }


 serverAddr = argv[1];
 sockfd = socket(AF_INET, SOCK_STREAM, 0);
 if (sockfd < 0) {
  printf("Error creating socket!\n");
  exit(1);
 }
 printf("Socket created...\n");
 memset(&addr, 0, sizeof(addr));
 addr.sin_family = AF_INET;
 addr.sin_addr.s_addr = inet_addr(serverAddr);
 addr.sin_port = PORT;

 ret = connect(sockfd, (struct sockaddr *) &addr, sizeof(addr));
 if (ret < 0) {
  printf("Error connecting to the server!\n");
  exit(1);
 }
 printf("Connected to the server...\n");
re_child=fork();
if(re_child>0)
{
for(; ;){
while (fgets(buffer, BUF_SIZE, stdin) != NULL) {
  ret = sendto(sockfd, buffer, BUF_SIZE, 0, (struct sockaddr *) &addr, sizeof(addr));
  if (ret < 0) {
   printf("Error sending data!\n\t-%s", buffer);
  }}
}else
```

```
    {
    for(; ;)
    {
    printf("Enter your message(s): \n");
    ret = recvfrom(sockfd, buffer, BUF_SIZE, 0, NULL, NULL);
     if (ret < 0) {
      printf("Error receiving data!\n");
     }
     else {
      printf("Received B : ");
      fputs(buffer, stdout);
      printf("\n");
    }}}
return 0;}
```

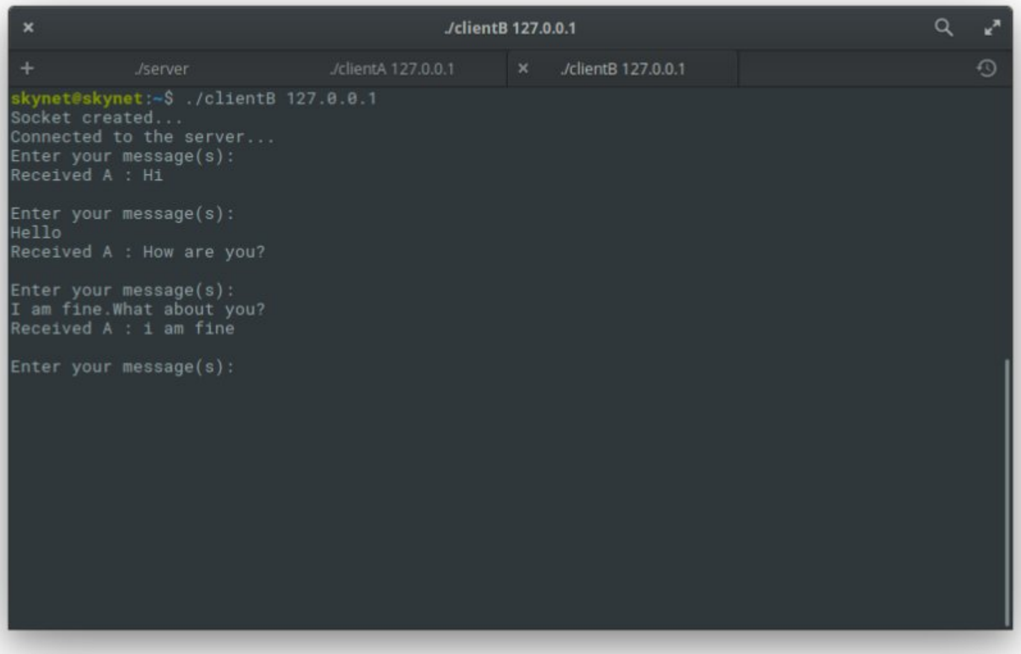


```
skynet@skynet:~$ ./clientA 127.0.0.1
Socket created...
Connected to the server...
Enter your message(s):
Hi
Received B : Hello

Enter your message(s):
How are you?
Received B : I am fine.What about you?

Enter your message(s):
i am fine
```

3. CLIENT B

```
#include"stdio.h"

#include"stdlib.h"

#include"sys/types.h"

#include"sys/socket.h"

#include"string.h"

#include"netinet/in.h"

#include"netdb.h"

#include"arpa/inet.h"

#include"unistd.h"


#define PORT 5555

#define BUF_SIZE 2000

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

 struct sockaddr_in addr, cl_addr;

 int sockfd, ret;

 pid_t re_child;

 char buffer[BUF_SIZE];

 struct hostent * server;

 char * serverAddr;

 if (argc < 2) {

  printf("usage: client < ip address >\n");

  exit(1);

 }

 serverAddr = argv[1];

 sockfd = socket(AF_INET, SOCK_STREAM, 0);

 if (sockfd < 0) {
  printf("Error creating socket!\n");
  exit(1);
 }
 printf("Socket created...\n");
 memset(&addr, 0, sizeof(addr));
 addr.sin_family = AF_INET;
 addr.sin_addr.s_addr = inet_addr(serverAddr);
 addr.sin_port = PORT;
 ret = connect(sockfd, (struct sockaddr *) &addr, sizeof(addr));
 if (ret < 0) {
  printf("Error connecting to the server!\n");
  exit(1);
 }
 printf("Connected to the server...\n");
re_child=fork();
if(re_child>0)
{
for(; ;)
{
 while (fgets(buffer, BUF_SIZE, stdin) != NULL) {
ret = sendto(sockfd, buffer, BUF_SIZE, 0, (struct sockaddr *) &addr, sizeof(addr));
  if (ret < 0) {
   printf("Error sending data!\n\t-%s", buffer);
}}}
}else
{
for(; ;)
```

```
{
printf("Enter your message(s): \n");
ret = recvfrom(sockfd, buffer, BUF_SIZE, 0, NULL, NULL);
  if (ret < 0) {
   printf("Error receiving data!\n");
  }
else {
   printf("Received A : ");
   fputs(buffer, stdout);
   printf("\n");
}}
return 0;}
```



```
skynet@skynet:~$ ./clientB 127.0.0.1
Socket created...
Connected to the server...
Enter your message(s):
Received A : Hi

Enter your message(s):
Hello
Received A : How are you?

Enter your message(s):
I am fine.What about you?
Received A : i am fine

Enter your message(s):
```

CONCLUSION:

Many clients can communicate with each other with server as platform of communication. Sending and receiving can occur simultaneously between the clients. The project is basically an application of group chat where clients are communicating with each other and history of messages is stored in server.

REFERENCES:

1.www.stackoverflow.com

2.Wikipedia