Experiment-10

Aim: Socket Programming.

1. **Iterative Server**

⇨ **Server:**

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

int main(){
        int cd,sd,n,clilen;
        struct sockaddr_in servaddr, cliaddr;
        char data[100];
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr=inet_addr("192.168.250.241");
        servaddr.sin_port = htons(9500);
        sd = socket(AF_INET,SOCK_STREAM,0);
        bind(sd,(struct sockaddr*)&servaddr,sizeof(servaddr));
        listen(sd,5);
        for(;;){
                clilen=sizeof(cliaddr);
                cd= accept(sd,(struct sockaddr*)&cliaddr,&clilen);
                printf("Connection from %s, port
%d\n",inet_ntoa(cliaddr.sin_addr),ntohs(cliaddr.sin_port));
                bzero(&data,sizeof(data));
```

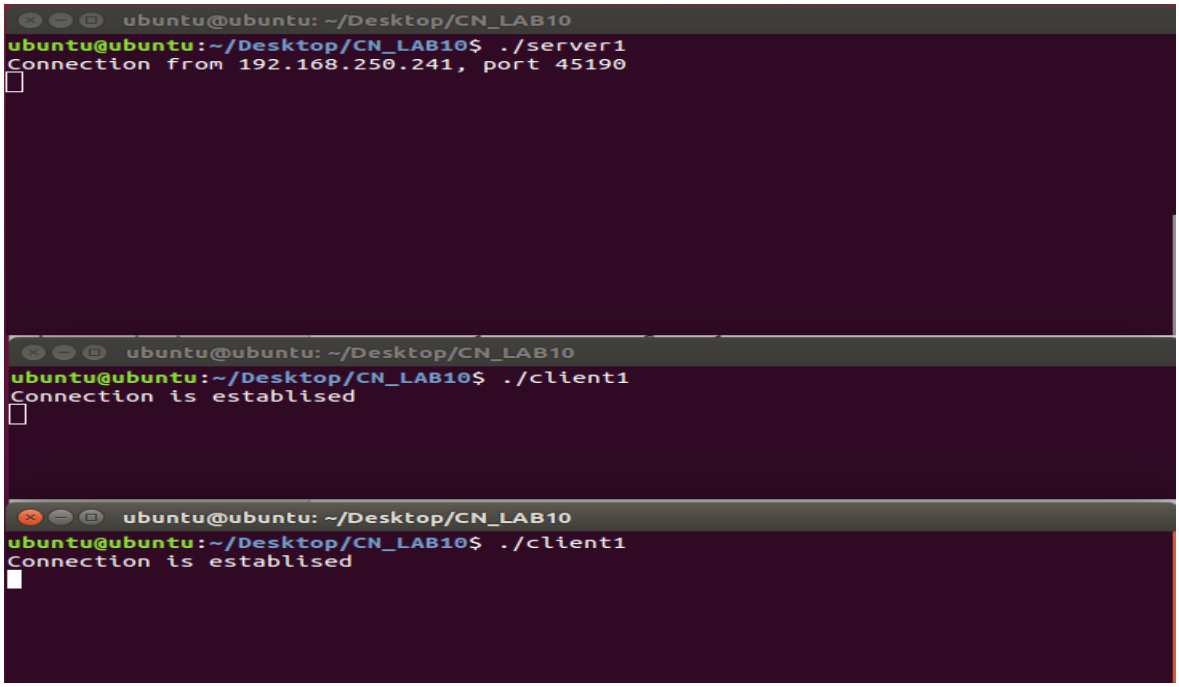
```
                n=read(cd,data,sizeof(data));
                data[n]='\0';
                write(cd,data,strlen(data));
                close(cd);
        }
        return 0;
}
```

⇨ **Client:**

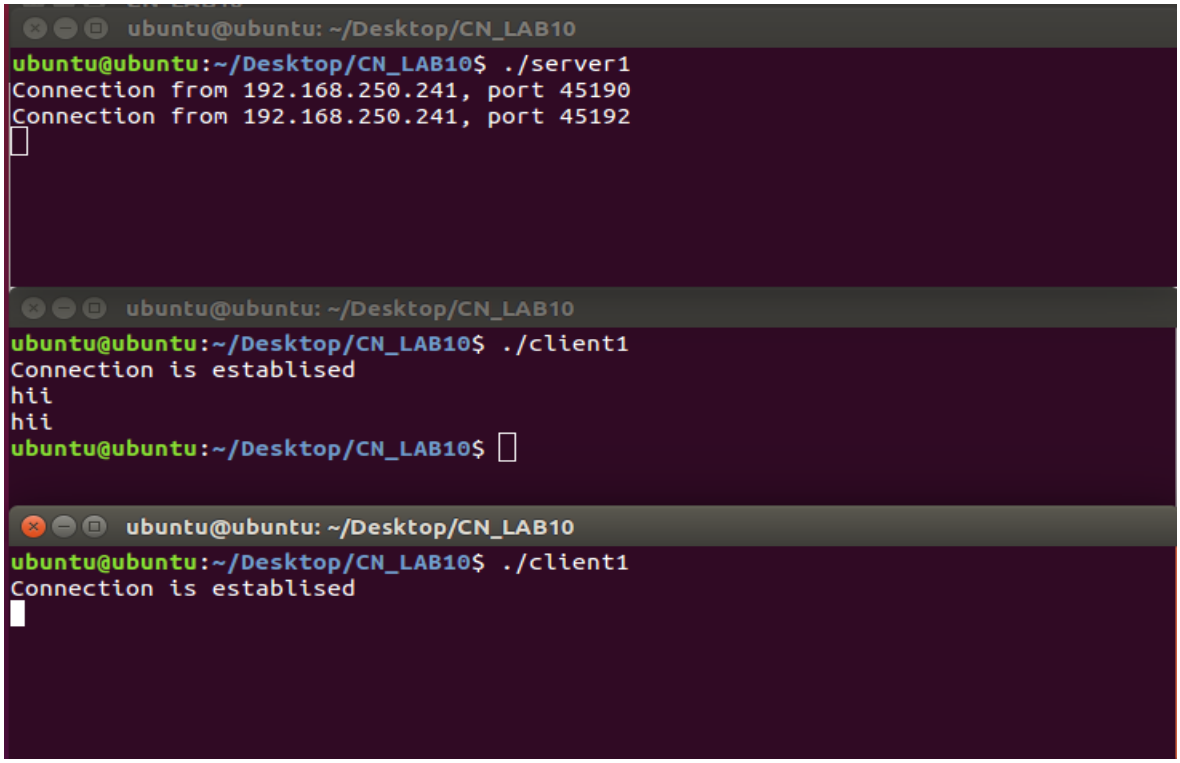
```
#include<stdlib.h>
#include<sys/socket.h>
#include<unistd.h>
#include<netinet/in.h>
#include<string.h>
#include<stdio.h>
#include<arpa/inet.h>

int main(){
        int cd,sd,n,i;
        struct sockaddr_in servaddr, myaddr;
        char data[100];
        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr=inet_addr("192.168.250.241");
        servaddr.sin_port = htons(9500);
        sd = socket(AF_INET,SOCK_STREAM,0);
        if(connect(sd,(struct
sockaddr*)&servaddr,sizeof(servaddr))==0){
                printf("Connection is establised\n");
```

```
            n=read(0,data,sizeof(data));
            write(sd,data,n);
            bzero(&data,sizeof(data));
            n=0;
            n=read(sd,data,sizeof(data));
            write(1,data,n);
            close(sd);
        }
        return 0;
}
```

⇨ **Output:**

a. 

- As we can see here only at a time one client is communicate with server. Others connection is established but it's in server's queue.

b. 

- Here when 1st client close the connection another client who are in queue get chance to communicate.

In server side code we can see one parameter in listen() it indicates the queue size. If connection request will come after the queue is full then those connections are rejected.

2. Concurrent Server

⇨ **Server**

```
#include<stdlib.h>
#include<sys/socket.h>
#include<sys/wait.h>
#include<unistd.h>
#include<signal.h>
#include<netinet/in.h>
#include<string.h>
#include<stdio.h>
#include<arpa/inet.h>

int main(){
        int cd,sd,n,pid;
        socklen_t clilen;
        struct sockaddr_in servaddr, cliaddr;
        char data[100];
        char cliaddrS[20];

        servaddr.sin_family = AF_INET;
        inet_aton("192.168.250.241",&servaddr.sin_addr);
        servaddr.sin_port = htons(9700);
        sd = socket(AF_INET,SOCK_STREAM,0);
        bind(sd,(struct sockaddr*)&servaddr,sizeof(servaddr));
        listen(sd,10);

        for(;;){
                clilen=sizeof(cliaddr);
```

```
		cd= accept(sd,(struct sockaddr*)&cliaddr,&clilen);

		if((pid=fork())==0){
			bzero(&cliaddrS,sizeof(cliaddrS));
				printf("In Child Process for client %s
		%d\n",inet_ntop(AF_INET,&cliaddr.sin_addr,cliaddrS,si
		zeof(cliaddrS)),ntohs(cliaddr.sin_port));
			close(sd);
			for(;;){
				bzero(&data,sizeof(data));
				n=read(cd,data,sizeof(data));
				data[n]='\0';
				if(n==0){
					bzero(&cliaddrS,sizeof(cliaddrS));
						printf("Client %s %d is
			closed\n",inet_ntop(AF_INET,&cliaddr.sin_addr,cl
			iaddrS,sizeof(cliaddrS)),ntohs(cliaddr.sin_port));
					exit(0);
				}
				write(cd,data,n);
			}
			exit(0);
		}
		close(cd);
	}
	close(sd);
	return 0;
}
```

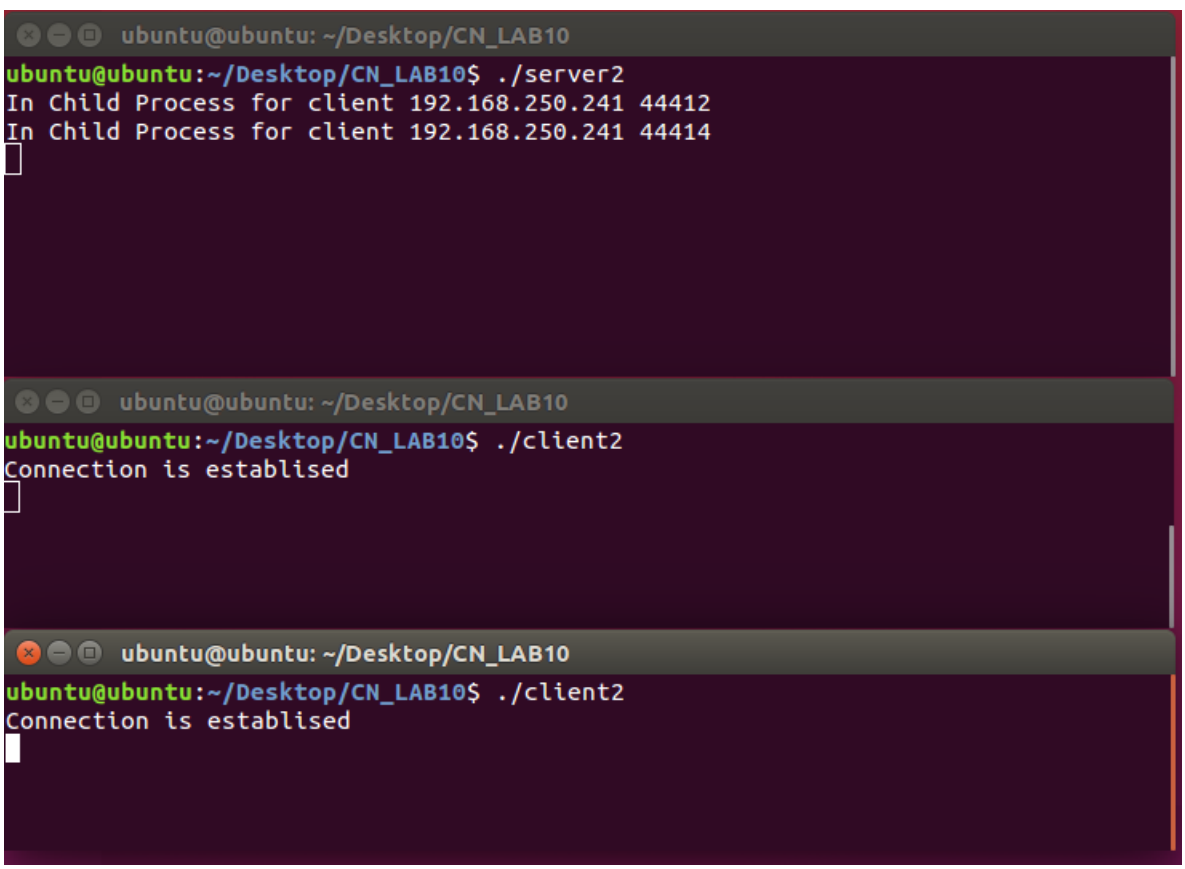
⇨ **Client**

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

int main(){
        int cd,sd,n,i,cv,flag;
        struct sockaddr_in servaddr, myaddr;
        char data[100];

        servaddr.sin_family = AF_INET;
        servaddr.sin_addr.s_addr=inet_addr("192.168.250.241");
        servaddr.sin_port = htons(9700);

        sd = socket(AF_INET,SOCK_STREAM,0);
        cv=connect(sd,(struct
sockaddr*)&servaddr,sizeof(servaddr));
        if(cv<0){
                printf("Error in connection\n");
                exit(0);
        }
        printf("Connection is establised\n");
        while(1){
                bzero(&data,sizeof(data));
```

```
            n=read(0,data,sizeof(data));
            flag=strncmp(data,"exit",4);
            if(!flag)
                break;
            write(sd,data,n);

            bzero(&data,sizeof(data));
            n=read(sd,data,sizeof(data));
            write(1,data,n);
        }
        close(sd);
        return 0;
}
```

⇨ **Output**

a.



- Here we can see both the connections are established and can communicate with server parallel.

b.

```
ubuntu@ubuntu: ~/Desktop/CN_LAB10
ubuntu@ubuntu:~/Desktop/CN_LAB10$ ./server2
In Child Process for client 192.168.250.241 44412
In Child Process for client 192.168.250.241 44414
```

```
ubuntu@ubuntu: ~/Desktop/CN_LAB10
ubuntu@ubuntu:~/Desktop/CN_LAB10$ ./client2
Connection is establised
hii
hii
```

```
ubuntu@ubuntu: ~/Desktop/CN_LAB10
ubuntu@ubuntu:~/Desktop/CN_LAB10$ ./client2
Connection is establised
```

c.

```
ubuntu@ubuntu: ~/Desktop/CN_LAB10
ubuntu@ubuntu:~/Desktop/CN_LAB10$ ./server2
In Child Process for client 192.168.250.241 44412
In Child Process for client 192.168.250.241 44414
```

```
ubuntu@ubuntu: ~/Desktop/CN_LAB10
ubuntu@ubuntu:~/Desktop/CN_LAB10$ ./client2
Connection is establised
hii
hii
```

```
ubuntu@ubuntu: ~/Desktop/CN_LAB10
Connection is establised
hello
hello
```

d.

```
ubuntu@ubuntu: ~/Desktop/CN_LAB10
ubuntu@ubuntu:~/Desktop/CN_LAB10$ ./server2
In Child Process for client 192.168.250.241 44412
In Child Process for client 192.168.250.241 44414
Client 192.168.250.241 44412 is closed
```

```
ubuntu@ubuntu: ~/Desktop/CN_LAB10
ubuntu@ubuntu:~/Desktop/CN_LAB10$ ./client2
Connection is establised
hii
hii
^C
ubuntu@ubuntu:~/Desktop/CN_LAB10$
```

```
ubuntu@ubuntu: ~/Desktop/CN_LAB10
Connection is establised
hello
hello
```

- Here we can see one client closed its connection but another one can still communicate with server.