Computer Networks

Name : M. Sai Saranya

Regno: 22BAI1471

Course Title : Computer Networks

Course code : BCSE308P

Slot : L45-46

Faculty : Dr Neelanarayanan V

S.No	Experiment Name	Date	Page No.	Marks
1.	**Basic Network Configuration Commands**	**10-01-2024**		
2.	**Client-Server Application Echo**	**17-01-2024**		
3.	**IP Address Validation and Simple application of ATM using TCP**	**24-01-2024**		
4.	**CRC code generator using socket programming**	**07-02-2024**		
5. a)	**Echo programming using UDP**	**21-02-2024**		
5. b)	**IP address validation using UDP**	**21-02-2024**		

S.No	Experiment Name	Date	Page No.	Marks
5. c)	**ATM simulation using UDP**	**21-02-2024**		

Experiment No. 5

Experiment Name: Client-Server Application (Echo client-server)

Date: 21-2-2024

Problem Statement

Design a simple client-server application named Echo client-server using c program in UDP protocol and execute in Linux.

Aim

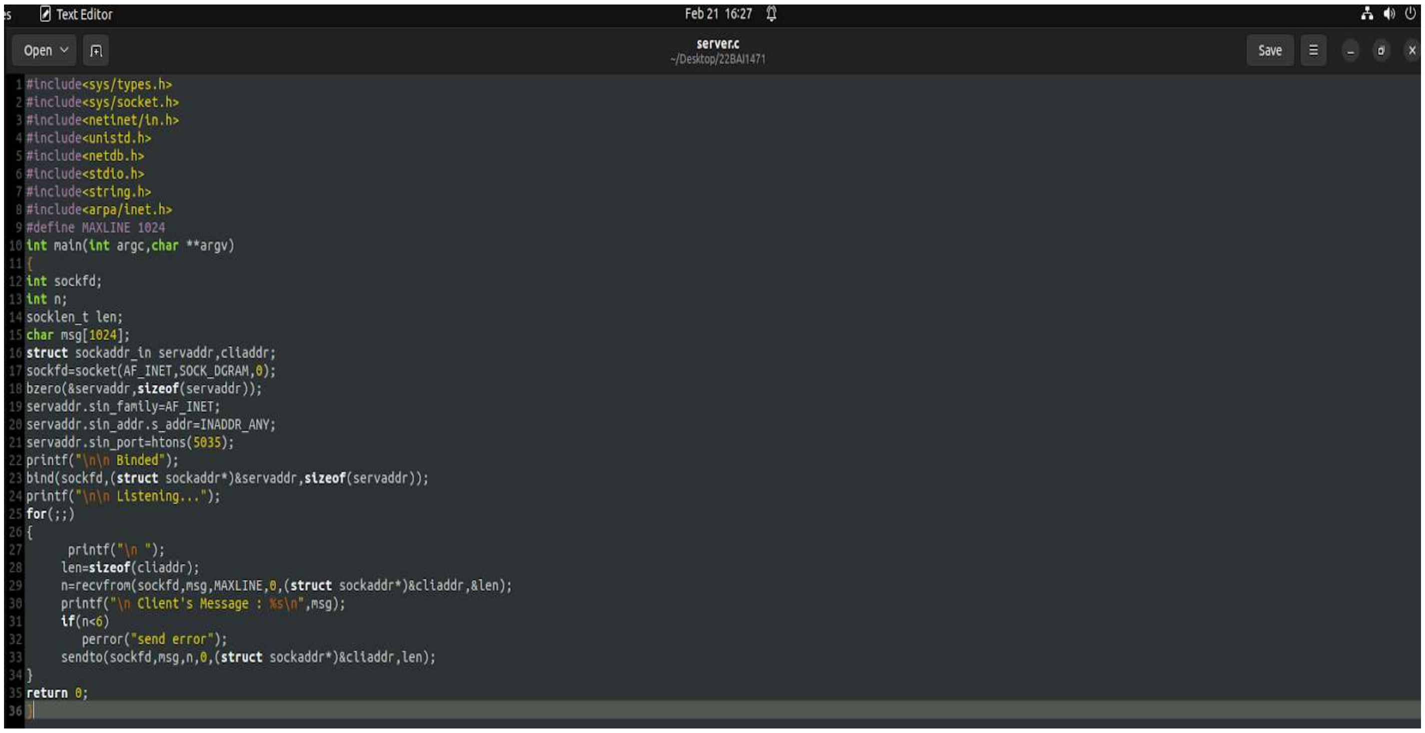
To write a c program for echo client-server application (UDP protocol) and execute in Linux environment.

Algorithm or Procedure

1. Start
2. Writing client and server files separately using socket programming and by using User Datagram Protocol(UDP)
3. Create a UDP socket
4. Assign a port to the socket
5. Communicate with server and client simultaneously

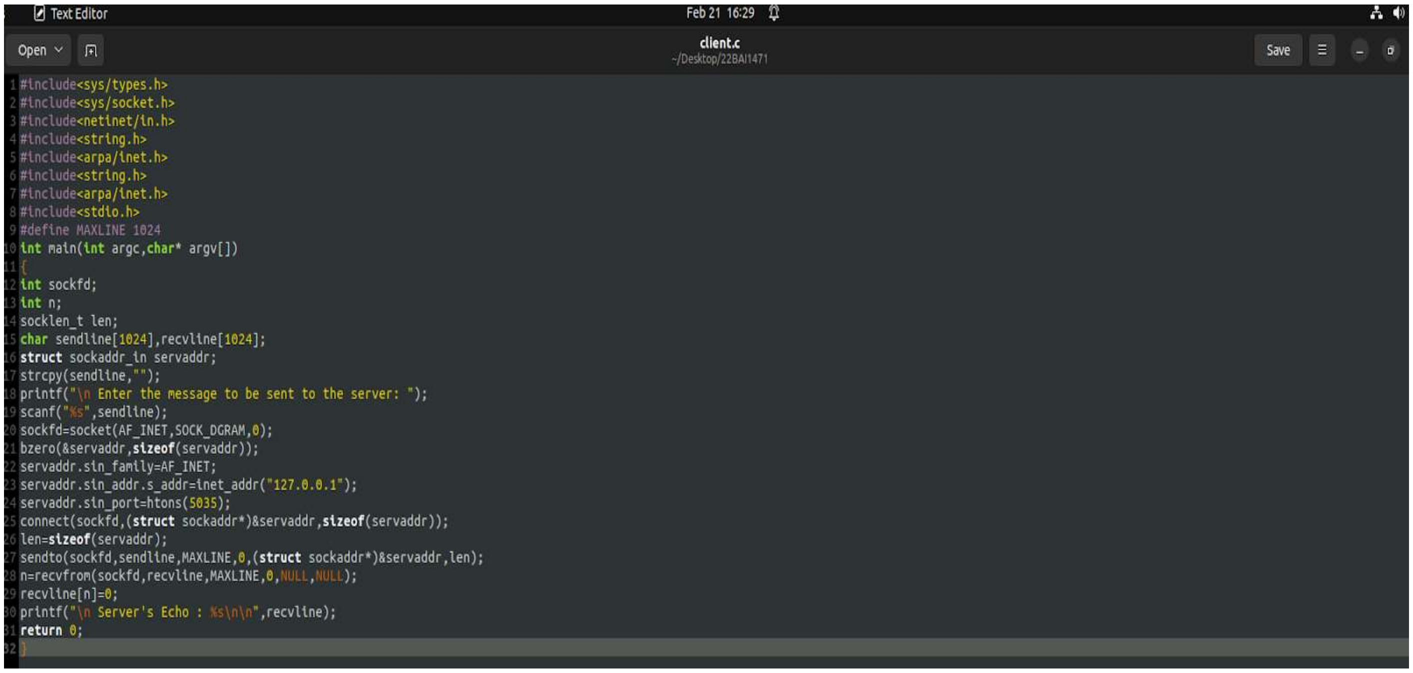
6. Close the socket
7. Execute client and server files separately or simultaneously in two parallel linux terminal windows.
8. Giving the message that is to be return back by server through client.
9. Obtaining desired output
10. Stop

Server side program:



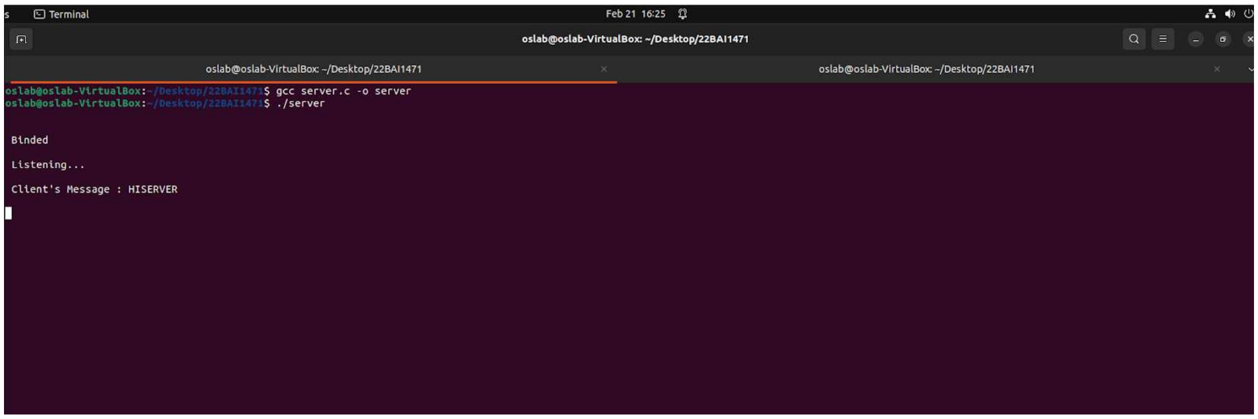
```c
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<netdb.h>
#include<stdio.h>
#include<string.h>
#include<arpa/inet.h>
#define MAXLINE 1024
int main(int argc,char **argv)
{
int sockfd;
int n;
socklen_t len;
char msg[1024];
struct sockaddr_in servaddr,cliaddr;
sockfd=socket(AF_INET,SOCK_DGRAM,0);
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=INADDR_ANY;
servaddr.sin_port=htons(5035);
printf("\n\n Binded");
bind(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
printf("\n\n Listening...");
for(;;)
{
      printf("\n ");
     len=sizeof(cliaddr);
     n=recvfrom(sockfd,msg,MAXLINE,0,(struct sockaddr*)&cliaddr,&len);
     printf("\n Client's Message : %s\n",msg);
     if(n<6)
        perror("send error");
     sendto(sockfd,msg,n,0,(struct sockaddr*)&cliaddr,len);
}
return 0;
}
```

Client side programming:



```c
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<arpa/inet.h>
#include<string.h>
#include<arpa/inet.h>
#include<stdio.h>
#define MAXLINE 1024
int main(int argc,char* argv[])
{
int sockfd;
int n;
socklen_t len;
char sendline[1024],recvline[1024];
struct sockaddr_in servaddr;
strcpy(sendline,"");
printf("\n Enter the message to be sent to the server: ");
scanf("%s",sendline);
sockfd=socket(AF_INET,SOCK_DGRAM,0);
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
servaddr.sin_port=htons(5035);
connect(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
len=sizeof(servaddr);
sendto(sockfd,sendline,MAXLINE,0,(struct sockaddr*)&servaddr,len);
n=recvfrom(sockfd,recvline,MAXLINE,0,NULL,NULL);
recvline[n]=0;
printf("\n Server's Echo : %s\n\n",recvline);
return 0;
}
```

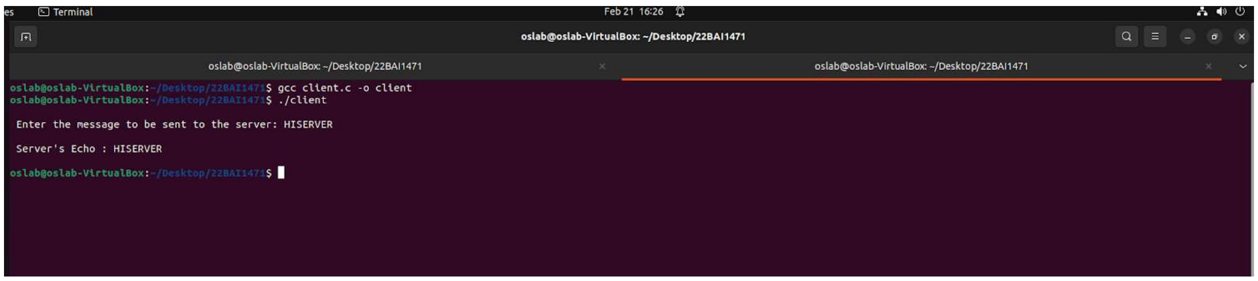
Output at server side



Terminal
Feb 21 16:25
oslab@oslab-VirtualBox: ~/Desktop/22BAI1471
oslab@oslab-VirtualBox: ~/Desktop/22BAI1471
oslab@oslab-VirtualBox: ~/Desktop/22BAI1471

```
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$ gcc server.c -o server
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$ ./server


 Binded

 Listening...

 Client's Message : HISERVER

```

Output at client side



Terminal
Feb 21 16:26
oslab@oslab-VirtualBox: ~/Desktop/22BAI1471
oslab@oslab-VirtualBox: ~/Desktop/22BAI1471
oslab@oslab-VirtualBox: ~/Desktop/22BAI1471

```
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$ gcc client.c -o client
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$ ./client

 Enter the message to be sent to the server: HISERVER

 Server's Echo : HISERVER

oslab@oslab-VirtualBox:~/Desktop/22BAI1471$ 
```

Conclusion

Linux terminal : Output

When a message is entered in the command prompt, the same message is reflected or sent back to the client by the server thus making an echo.
This is achieved by using UDP protocol.(no-feedback method).

Problem Statement

1) **Write a program to validate IP address**
2) **Implement a simulation of ATM functions using a UDP socket client server program**

Aim

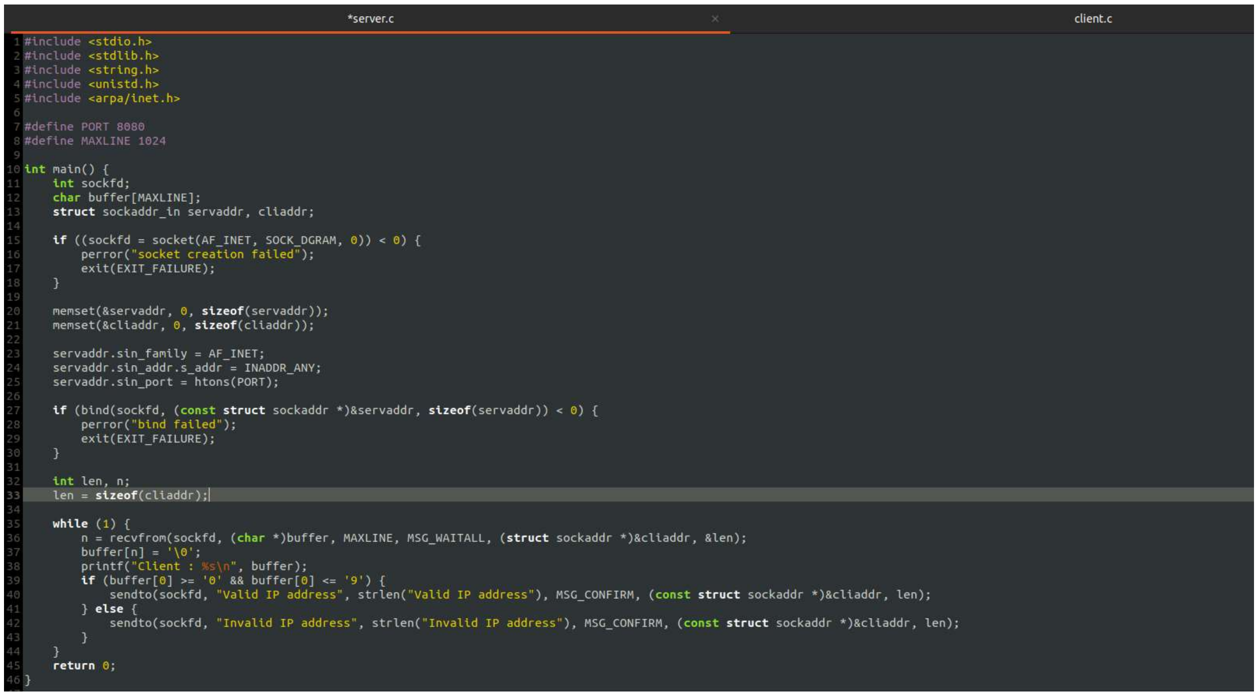
To write a c program for IP address validation and implementation of ATM basic functions using UDP socket client server program

Algorithm or Procedure

IPv4 Validation :

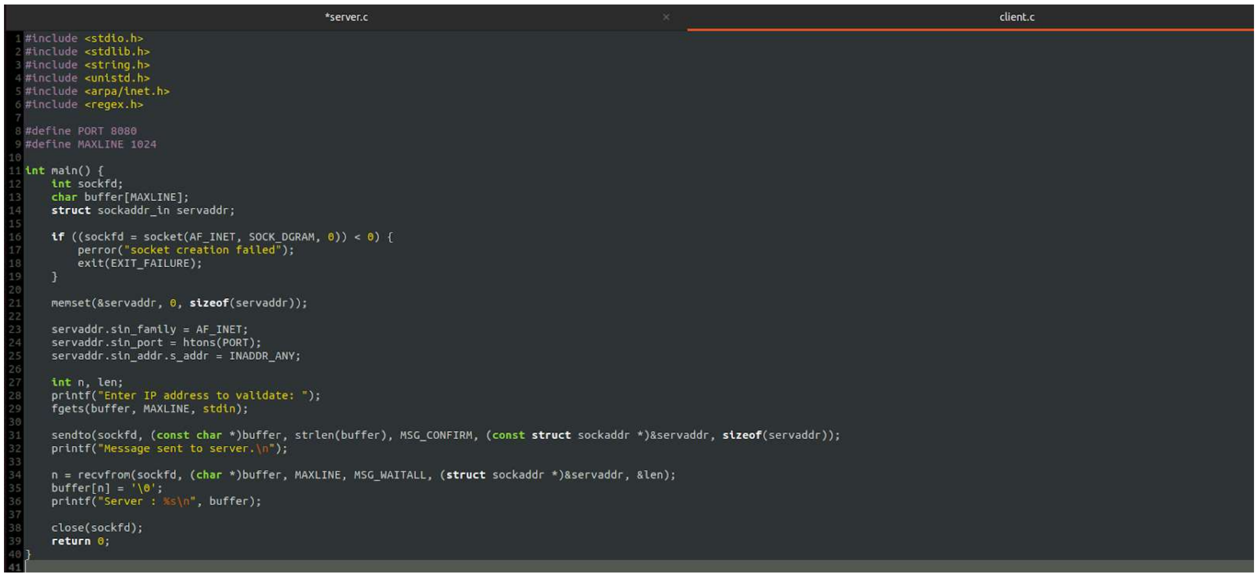
1. Split string by ., ensure exactly 4 parts.
2. Each part: convert to int, check 0-255 range.
3. No part can have leading zeros (except "0" itself).
4. No alpha characters allowed in any part.
5. If all checks pass, valid; else, invalid.

Server side program



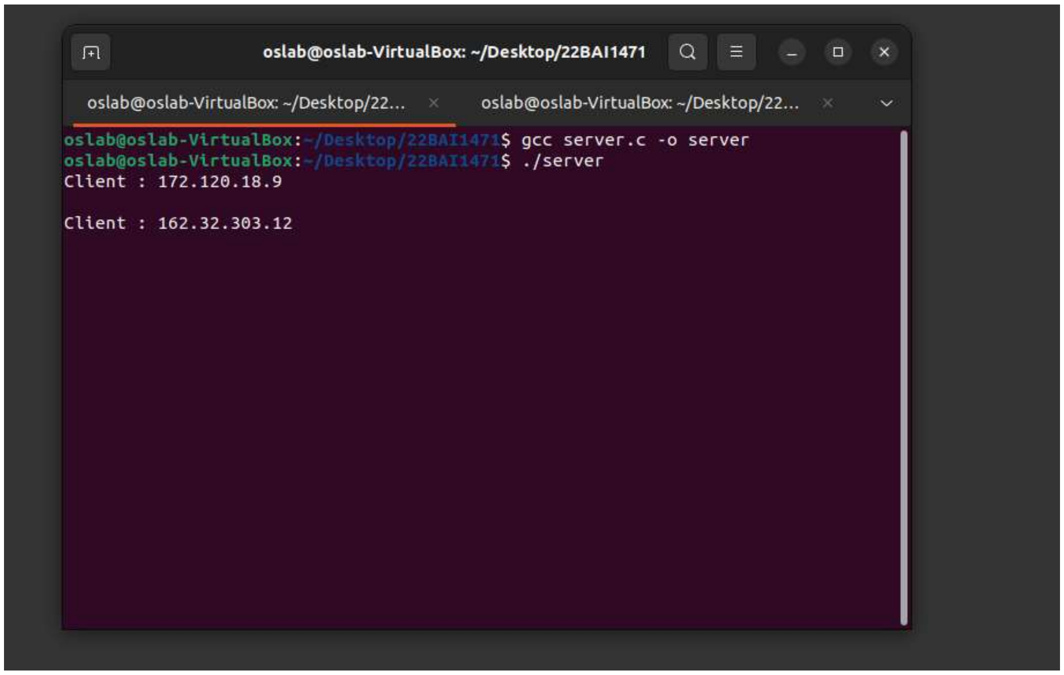
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define MAXLINE 1024

int main() {
    int sockfd;
    char buffer[MAXLINE];
    struct sockaddr_in servaddr, cliaddr;

    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&servaddr, 0, sizeof(servaddr));
    memset(&cliaddr, 0, sizeof(cliaddr));

    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(PORT);

    if (bind(sockfd, (const struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

    int len, n;
    len = sizeof(cliaddr);

    while (1) {
        n = recvfrom(sockfd, (char *)buffer, MAXLINE, MSG_WAITALL, (struct sockaddr *)&cliaddr, &len);
        buffer[n] = '\0';
        printf("Client : %s\n", buffer);
        if (buffer[0] >= '0' && buffer[0] <= '9') {
            sendto(sockfd, "Valid IP address", strlen("Valid IP address"), MSG_CONFIRM, (const struct sockaddr *)&cliaddr, len);
        } else {
            sendto(sockfd, "Invalid IP address", strlen("Invalid IP address"), MSG_CONFIRM, (const struct sockaddr *)&cliaddr, len);
        }
    }
    return 0;
}
```

Client side program:



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <regex.h>

#define PORT 8080
#define MAXLINE 1024

int main() {
    int sockfd;
    char buffer[MAXLINE];
    struct sockaddr_in servaddr;

    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }

    memset(&servaddr, 0, sizeof(servaddr));

    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORT);
    servaddr.sin_addr.s_addr = INADDR_ANY;

    int n, len;
    printf("Enter IP address to validate: ");
    fgets(buffer, MAXLINE, stdin);

    sendto(sockfd, (const char *)buffer, strlen(buffer), MSG_CONFIRM, (const struct sockaddr *)&servaddr, sizeof(servaddr));
    printf("Message sent to server.\n");

    n = recvfrom(sockfd, (char *)buffer, MAXLINE, MSG_WAITALL, (struct sockaddr *)&servaddr, &len);
    buffer[n] = '\0';
    printf("Server : %s\n", buffer);

    close(sockfd);
    return 0;
}
```

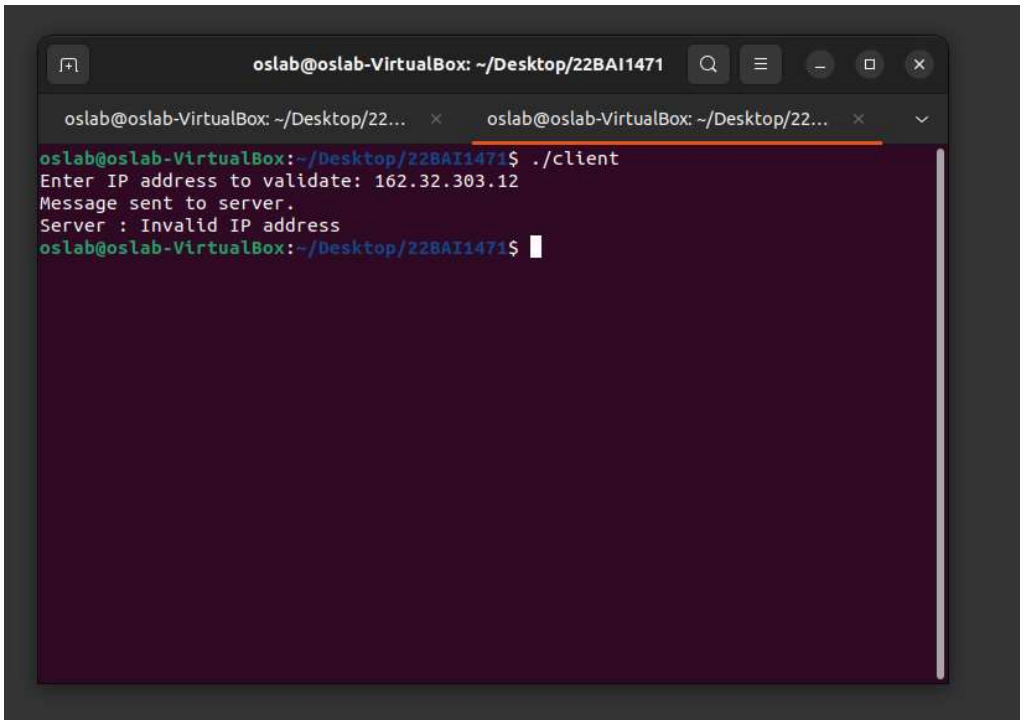
Server side



oslab@oslab-VirtualBox: ~/Desktop/22BAI1471
oslab@oslab-VirtualBox: ~/Desktop/22...
oslab@oslab-VirtualBox: ~/Desktop/22...

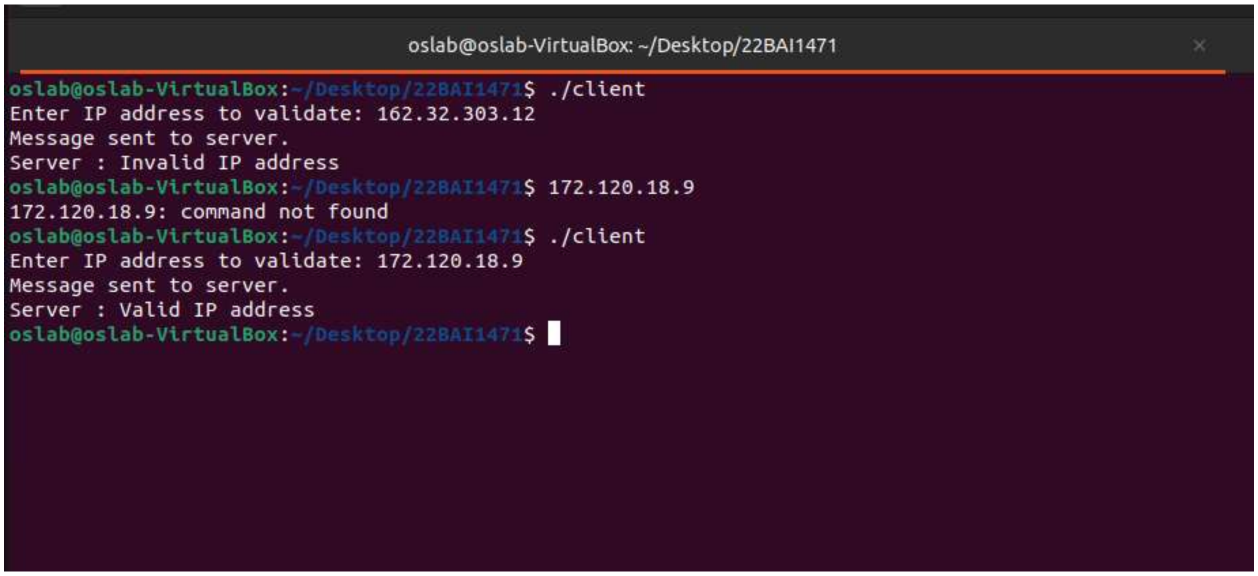
```
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$ gcc server.c -o server
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$ ./server
Client : 172.120.18.9

Client : 162.32.303.12
```

Client side



oslab@oslab-VirtualBox: ~/Desktop/22BAI1471
oslab@oslab-VirtualBox: ~/Desktop/22...
oslab@oslab-VirtualBox: ~/Desktop/22...

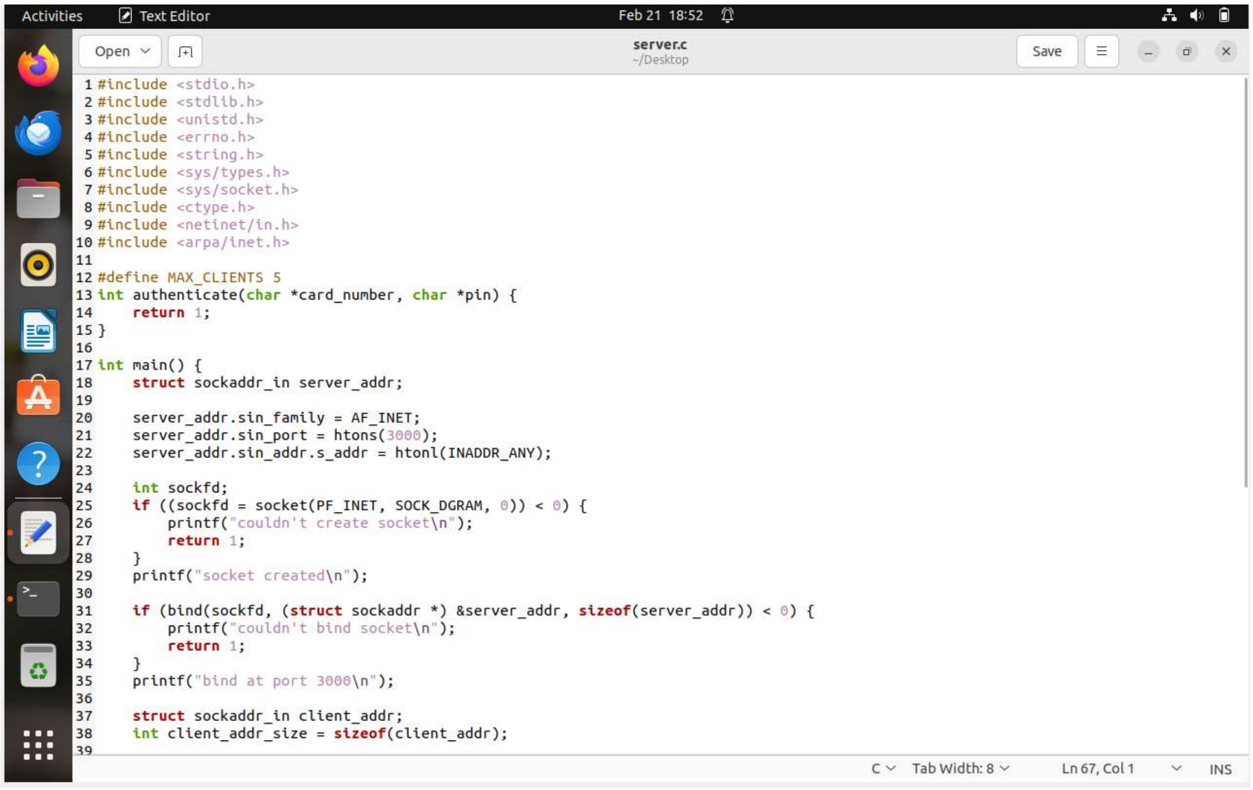
```
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$ ./client
Enter IP address to validate: 162.32.303.12
Message sent to server.
Server : Invalid IP address
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$ 
```



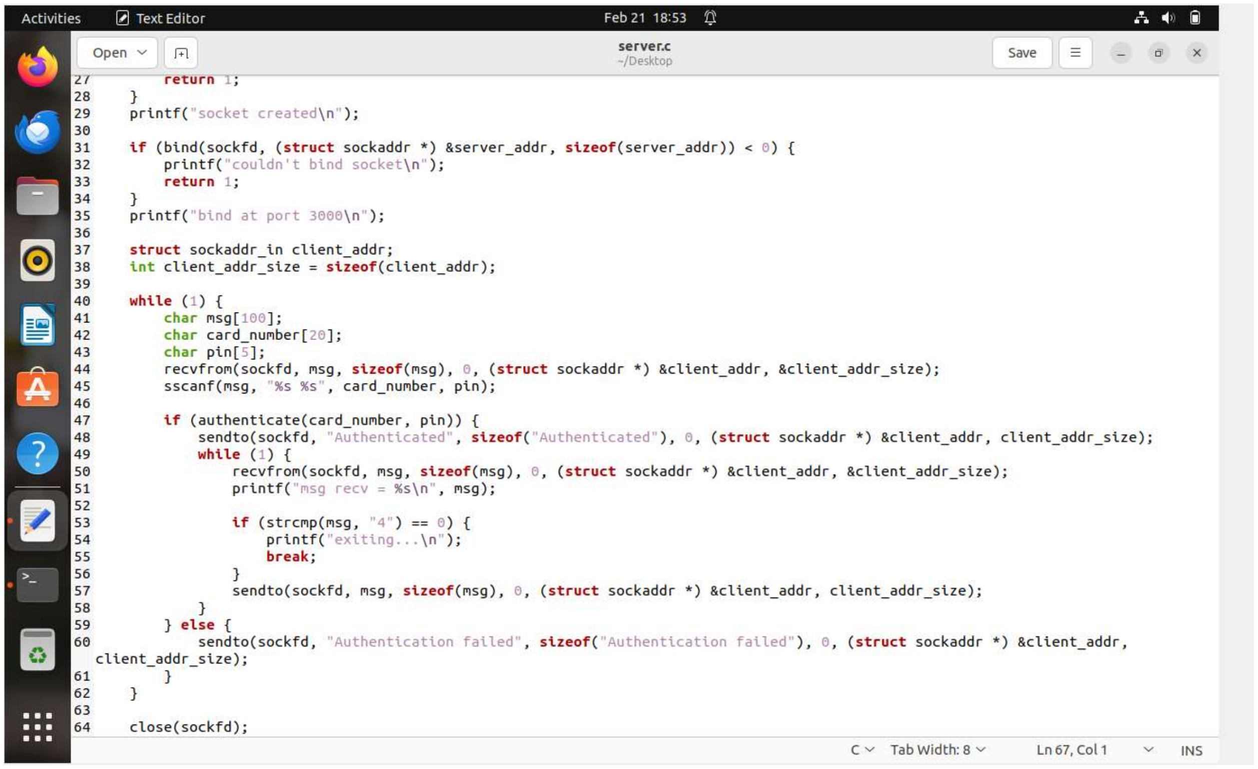
```
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$ ./client
Enter IP address to validate: 162.32.303.12
Message sent to server.
Server : Invalid IP address
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$ 172.120.18.9
172.120.18.9: command not found
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$ ./client
Enter IP address to validate: 172.120.18.9
Message sent to server.
Server : Valid IP address
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$
```

2) ATM simulation using UDP socket client server program

Server program

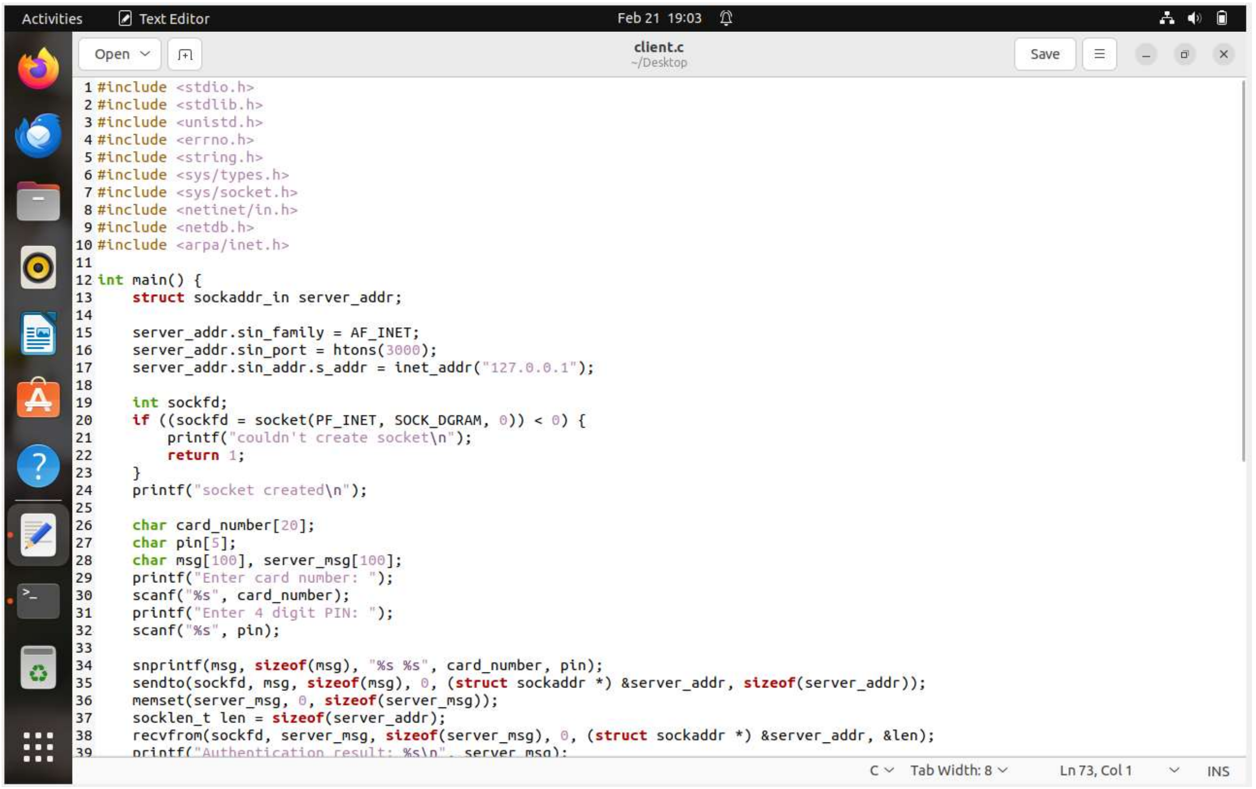


```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <ctype.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define MAX_CLIENTS 5
int authenticate(char *card_number, char *pin) {
    return 1;
}

int main() {
    struct sockaddr_in server_addr;

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(3000);
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);

    int sockfd;
    if ((sockfd = socket(PF_INET, SOCK_DGRAM, 0)) < 0) {
        printf("couldn't create socket\n");
        return 1;
    }
    printf("socket created\n");

    if (bind(sockfd, (struct sockaddr *) &server_addr, sizeof(server_addr)) < 0) {
        printf("couldn't bind socket\n");
        return 1;
    }
    printf("bind at port 3000\n");

    struct sockaddr_in client_addr;
    int client_addr_size = sizeof(client_addr);
```

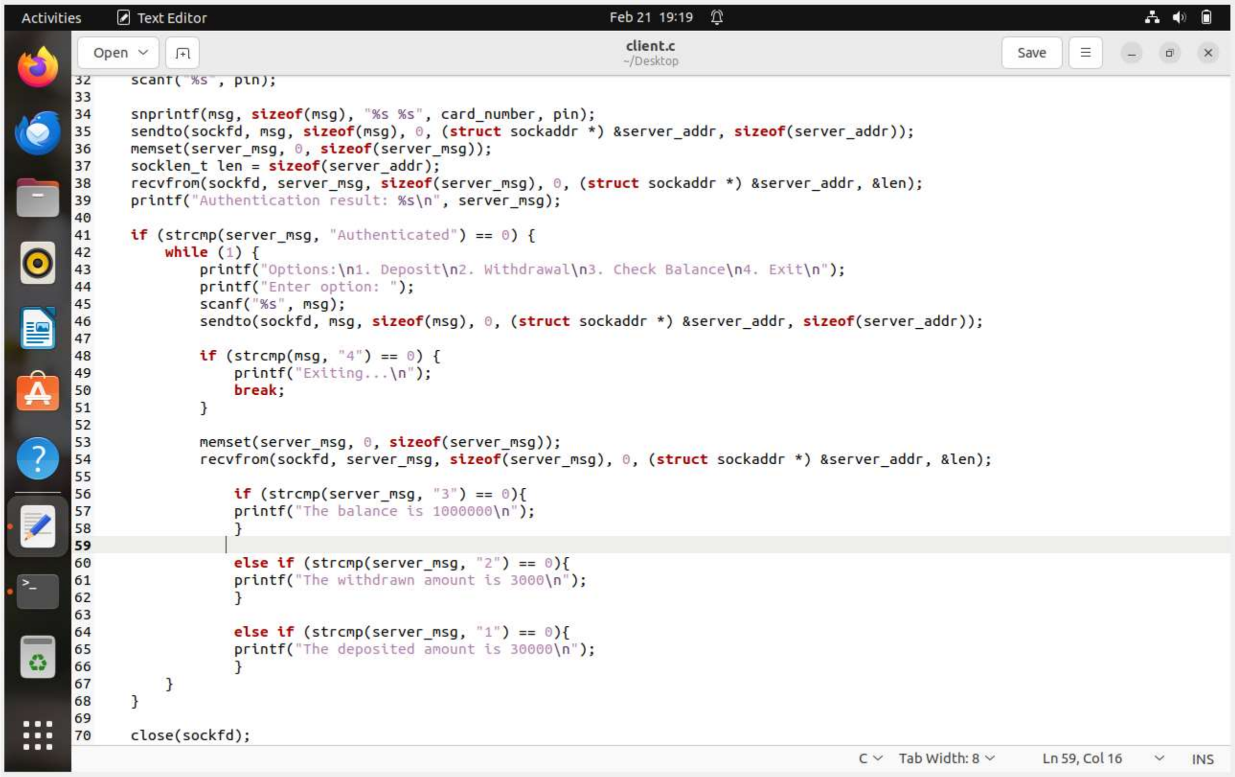


```c
        return 1;
    }
    printf("socket created\n");

    if (bind(sockfd, (struct sockaddr *) &server_addr, sizeof(server_addr)) < 0) {
        printf("couldn't bind socket\n");
        return 1;
    }
    printf("bind at port 3000\n");

    struct sockaddr_in client_addr;
    int client_addr_size = sizeof(client_addr);

    while (1) {
        char msg[100];
        char card_number[20];
        char pin[5];
        recvfrom(sockfd, msg, sizeof(msg), 0, (struct sockaddr *) &client_addr, &client_addr_size);
        sscanf(msg, "%s %s", card_number, pin);

        if (authenticate(card_number, pin)) {
            sendto(sockfd, "Authenticated", sizeof("Authenticated"), 0, (struct sockaddr *) &client_addr, client_addr_size);
            while (1) {
                recvfrom(sockfd, msg, sizeof(msg), 0, (struct sockaddr *) &client_addr, &client_addr_size);
                printf("msg recv = %s\n", msg);

                if (strcmp(msg, "4") == 0) {
                    printf("exiting...\n");
                    break;
                }
                sendto(sockfd, msg, sizeof(msg), 0, (struct sockaddr *) &client_addr, client_addr_size);
            }
        } else {
            sendto(sockfd, "Authentication failed", sizeof("Authentication failed"), 0, (struct sockaddr *) &client_addr, client_addr_size);
        }
    }

    close(sockfd);
```

Client program

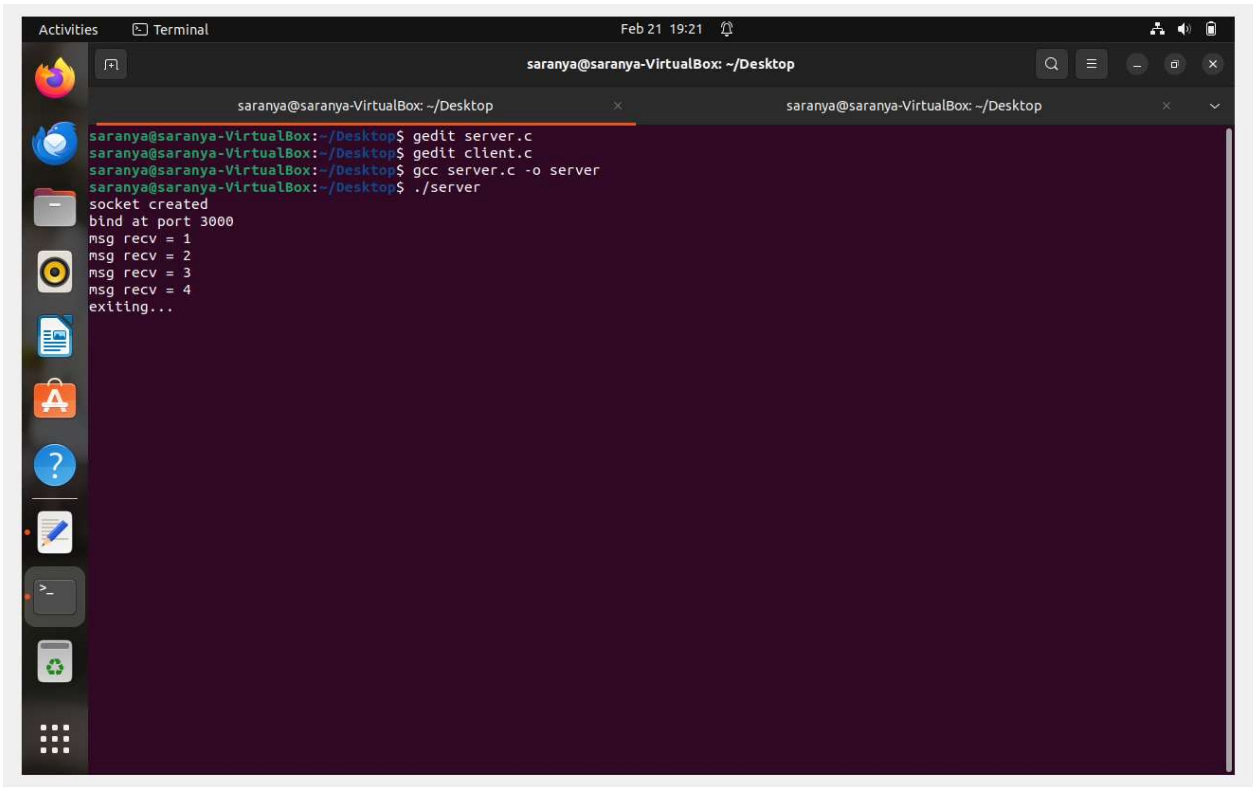


```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>

int main() {
    struct sockaddr_in server_addr;

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(3000);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    int sockfd;
    if ((sockfd = socket(PF_INET, SOCK_DGRAM, 0)) < 0) {
        printf("couldn't create socket\n");
        return 1;
    }
    printf("socket created\n");

    char card_number[20];
    char pin[5];
    char msg[100], server_msg[100];
    printf("Enter card number: ");
    scanf("%s", card_number);
    printf("Enter 4 digit PIN: ");
    scanf("%s", pin);

    snprintf(msg, sizeof(msg), "%s %s", card_number, pin);
    sendto(sockfd, msg, sizeof(msg), 0, (struct sockaddr *) &server_addr, sizeof(server_addr));
    memset(server_msg, 0, sizeof(server_msg));
    socklen_t len = sizeof(server_addr);
    recvfrom(sockfd, server_msg, sizeof(server_msg), 0, (struct sockaddr *) &server_addr, &len);
    printf("Authentication result: %s\n", server_msg);
```



```
    scanf("%s", pin);

    snprintf(msg, sizeof(msg), "%s %s", card_number, pin);
    sendto(sockfd, msg, sizeof(msg), 0, (struct sockaddr *) &server_addr, sizeof(server_addr));
    memset(server_msg, 0, sizeof(server_msg));
    socklen_t len = sizeof(server_addr);
    recvfrom(sockfd, server_msg, sizeof(server_msg), 0, (struct sockaddr *) &server_addr, &len);
    printf("Authentication result: %s\n", server_msg);

    if (strcmp(server_msg, "Authenticated") == 0) {
        while (1) {
            printf("Options:\n1. Deposit\n2. Withdrawal\n3. Check Balance\n4. Exit\n");
            printf("Enter option: ");
            scanf("%s", msg);
            sendto(sockfd, msg, sizeof(msg), 0, (struct sockaddr *) &server_addr, sizeof(server_addr));

            if (strcmp(msg, "4") == 0) {
                printf("Exiting...\n");
                break;
            }

            memset(server_msg, 0, sizeof(server_msg));
            recvfrom(sockfd, server_msg, sizeof(server_msg), 0, (struct sockaddr *) &server_addr, &len);

                if (strcmp(server_msg, "3") == 0){
                printf("The balance is 1000000\n");
                }

                else if (strcmp(server_msg, "2") == 0){
                printf("The withdrawn amount is 3000\n");
                }

                else if (strcmp(server_msg, "1") == 0){
                printf("The deposited amount is 30000\n");
                }
        }
    }

    close(sockfd);
```

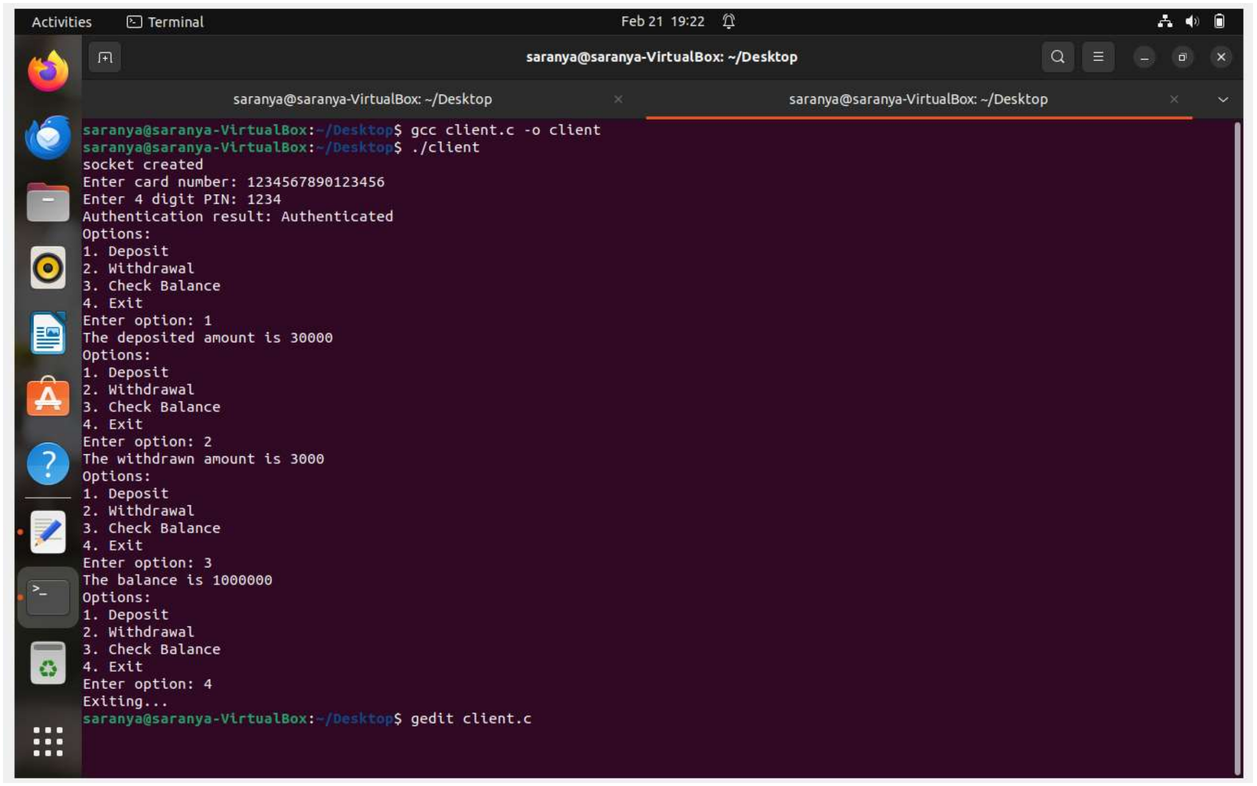
Output
Server side terminal



Activities
Terminal
Feb 21 19:21
saranya@saranya-VirtualBox: ~/Desktop
saranya@saranya-VirtualBox: ~/Desktop
saranya@saranya-VirtualBox: ~/Desktop

```
saranya@saranya-VirtualBox:~/Desktop$ gedit server.c
saranya@saranya-VirtualBox:~/Desktop$ gedit client.c
saranya@saranya-VirtualBox:~/Desktop$ gcc server.c -o server
saranya@saranya-VirtualBox:~/Desktop$ ./server
socket created
bind at port 3000
msg recv = 1
msg recv = 2
msg recv = 3
msg recv = 4
exiting...
```

Client side



Activities
Terminal
Feb 21 19:22
saranya@saranya-VirtualBox: ~/Desktop
saranya@saranya-VirtualBox: ~/Desktop
saranya@saranya-VirtualBox: ~/Desktop

```
saranya@saranya-VirtualBox:~/Desktop$ gcc client.c -o client
saranya@saranya-VirtualBox:~/Desktop$ ./client
socket created
Enter card number: 1234567890123456
Enter 4 digit PIN: 1234
Authentication result: Authenticated
Options:
1. Deposit
2. Withdrawal
3. Check Balance
4. Exit
Enter option: 1
The deposited amount is 30000
Options:
1. Deposit
2. Withdrawal
3. Check Balance
4. Exit
Enter option: 2
The withdrawn amount is 3000
Options:
1. Deposit
2. Withdrawal
3. Check Balance
4. Exit
Enter option: 3
The balance is 1000000
Options:
1. Deposit
2. Withdrawal
3. Check Balance
4. Exit
Enter option: 4
Exiting...
saranya@saranya-VirtualBox:~/Desktop$ gedit client.c
```

Conclusion

The program gives the user options to choose various banking services.
This program uses User Datagram Protocol which is feedback-less connection and also provides faster data transfer than TCP