

Computer Networks

Name : M. Sai Saranya

Regno: 22BAI1471

Course Title : Computer Networks

Course code : BCSE308P

Slot : L45-46

Faculty : Dr Neelananarayanan 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: IP address validation and simulation of ATM using UDP socket-client server

Date: 21-2-2024

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

```
*server.c
client.c

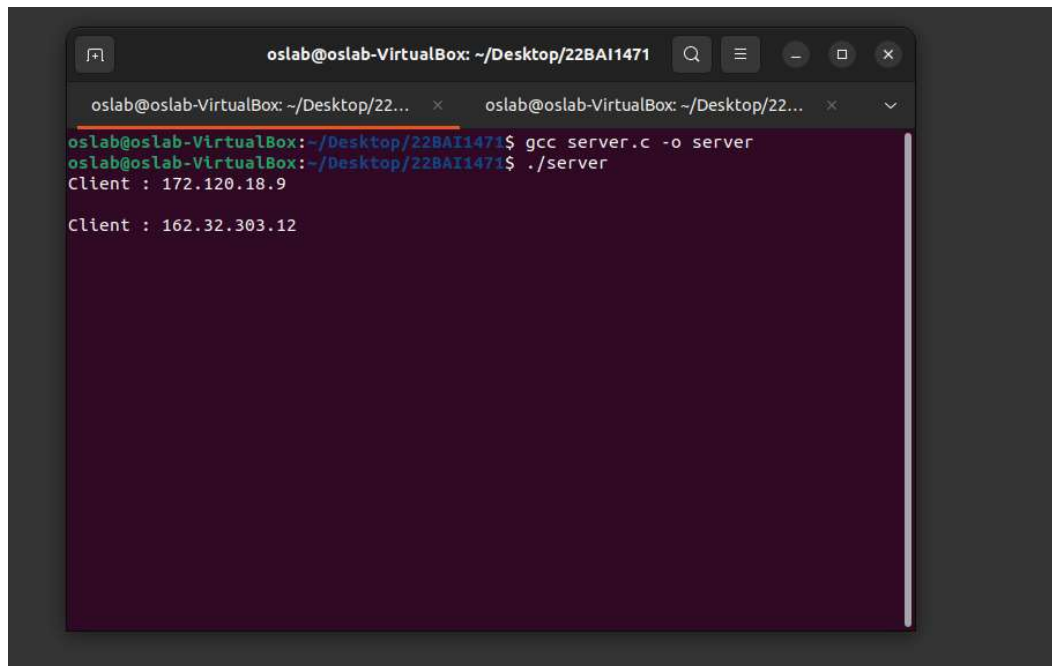
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5 #include <arpa/inet.h>
6
7 #define PORT 8080
8 #define MAXLINE 1024
9
10 int main() {
11     int sockfd;
12     char buffer[MAXLINE];
13     struct sockaddr_in servaddr, cliaddr;
14
15     if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
16         perror("socket creation failed");
17         exit(EXIT_FAILURE);
18     }
19
20     memset(&servaddr, 0, sizeof(servaddr));
21     memset(&cliaddr, 0, sizeof(cliaddr));
22
23     servaddr.sin_family = AF_INET;
24     servaddr.sin_addr.s_addr = INADDR_ANY;
25     servaddr.sin_port = htons(PORT);
26
27     if (bind(sockfd, (const struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
28         perror("bind failed");
29         exit(EXIT_FAILURE);
30     }
31
32     int len, n;
33     len = sizeof(cliaddr);
34
35     while (1) {
36         n = recvfrom(sockfd, (char *)buffer, MAXLINE, MSG_WAITALL, (struct sockaddr *)&cliaddr, &len);
37         buffer[n] = '\0';
38         printf("Client : %s\n", buffer);
39         if (buffer[0] >= '0' && buffer[0] <= '9') {
40             sendto(sockfd, "Valid IP address", strlen("Valid IP address"), MSG_CONFIRM, (const struct sockaddr *)&cliaddr, len);
41         } else {
42             sendto(sockfd, "Invalid IP address", strlen("Invalid IP address"), MSG_CONFIRM, (const struct sockaddr *)&cliaddr, len);
43         }
44     }
45     return 0;
46 }
```

Client side program:

```
*server.c
client.c

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5 #include <arpa/inet.h>
6 #include <regex.h>
7
8 #define PORT 8080
9 #define MAXLINE 1024
10
11 int main() {
12     int sockfd;
13     char buffer[MAXLINE];
14     struct sockaddr_in servaddr;
15
16     if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
17         perror("socket creation failed");
18         exit(EXIT_FAILURE);
19     }
20
21     memset(&servaddr, 0, sizeof(servaddr));
22
23     servaddr.sin_family = AF_INET;
24     servaddr.sin_port = htons(PORT);
25     servaddr.sin_addr.s_addr = INADDR_ANY;
26
27     int n, len;
28     printf("Enter IP address to validate: ");
29     fgets(buffer, MAXLINE, stdin);
30
31     sendto(sockfd, (const char *)buffer, strlen(buffer), MSG_CONFIRM, (const struct sockaddr *)&servaddr, sizeof(servaddr));
32     printf("Message sent to server.\n");
33
34     n = recvfrom(sockfd, (char *)buffer, MAXLINE, MSG_WAITALL, (struct sockaddr *)&servaddr, &len);
35     buffer[n] = '\0';
36     printf("Server : %s\n", buffer);
37
38     close(sockfd);
39     return 0;
40 }
41
```

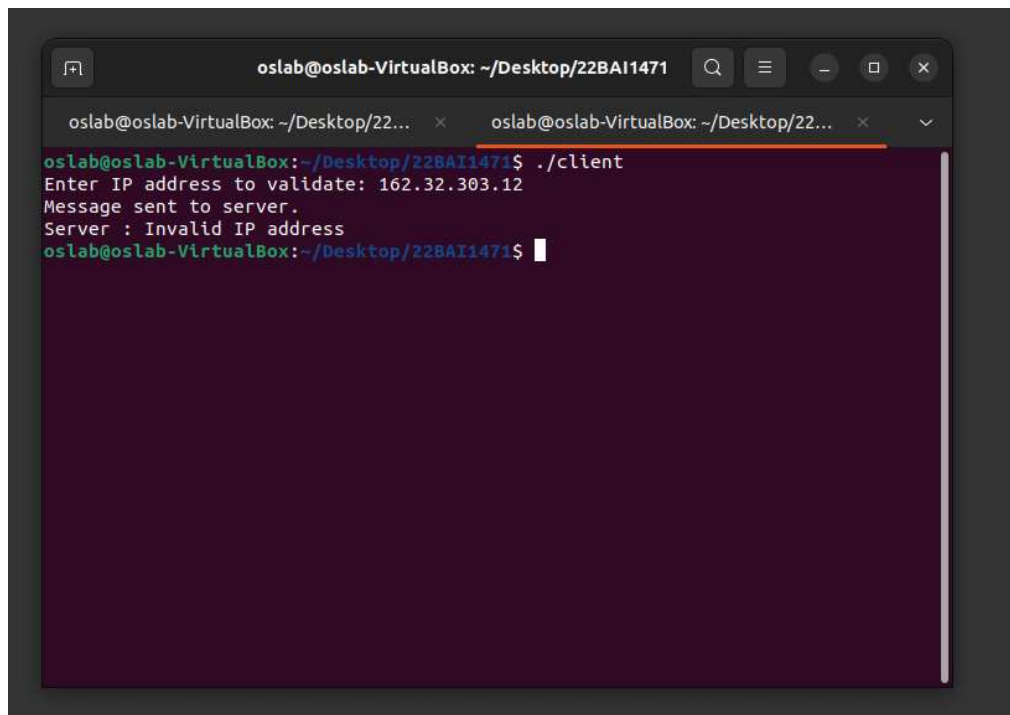
Server side



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

A terminal window titled 'oslab@oslab-VirtualBox: ~/Desktop/22BAI1471' showing the compilation of 'server.c' into 'server' and its execution. The output shows two client IP addresses: '172.120.18.9' and '162.32.303.12'.

Client side



```
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$
```

A terminal window titled 'oslab@oslab-VirtualBox: ~/Desktop/22BAI1471' showing the execution of the 'client' program. The user enters '162.32.303.12', and the program outputs 'Message sent to server.' followed by 'Server : Invalid IP address'.

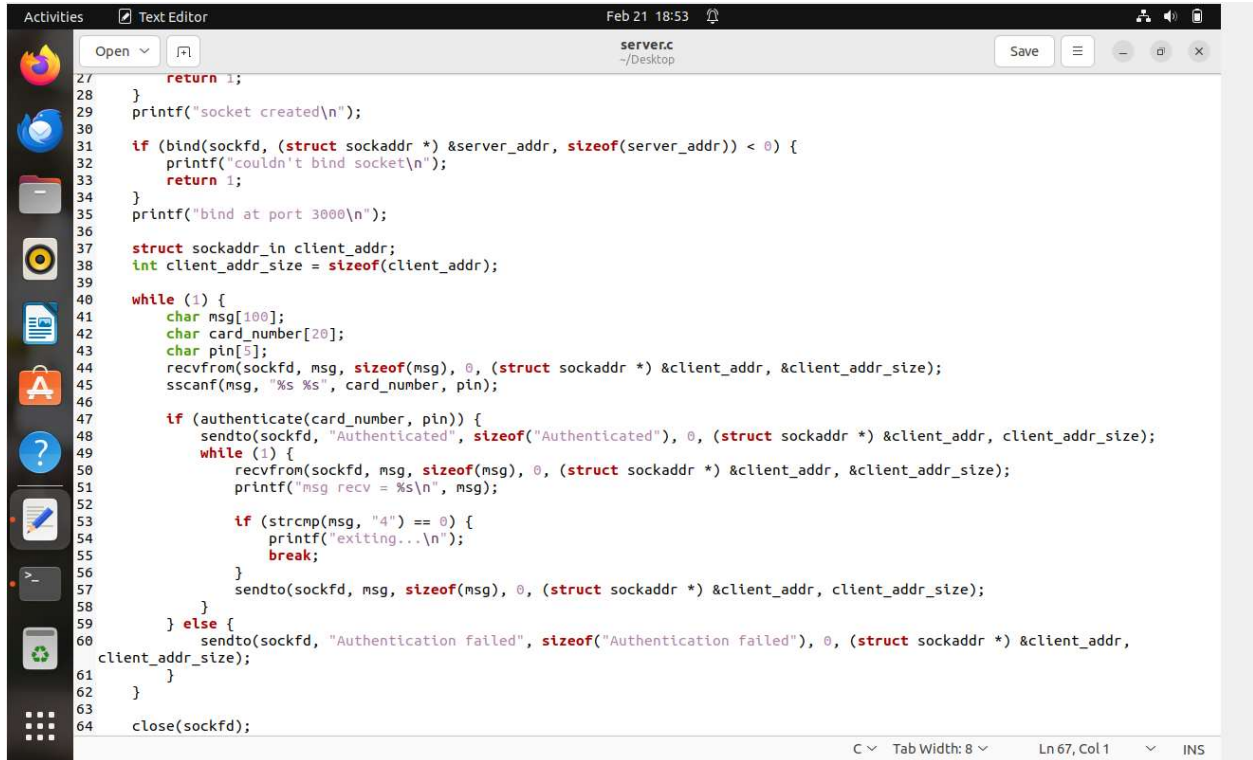
```
oslab@oslab-VirtualBox: ~/Desktop/22BAI1471
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$ ./client
Enter IP address to validate: 162.32.303.12
Message sent to server.
Server : Invalid IP address
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$ 172.120.18.9
172.120.18.9: command not found
oslab@oslab-VirtualBox:~/Desktop/22BAI1471$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

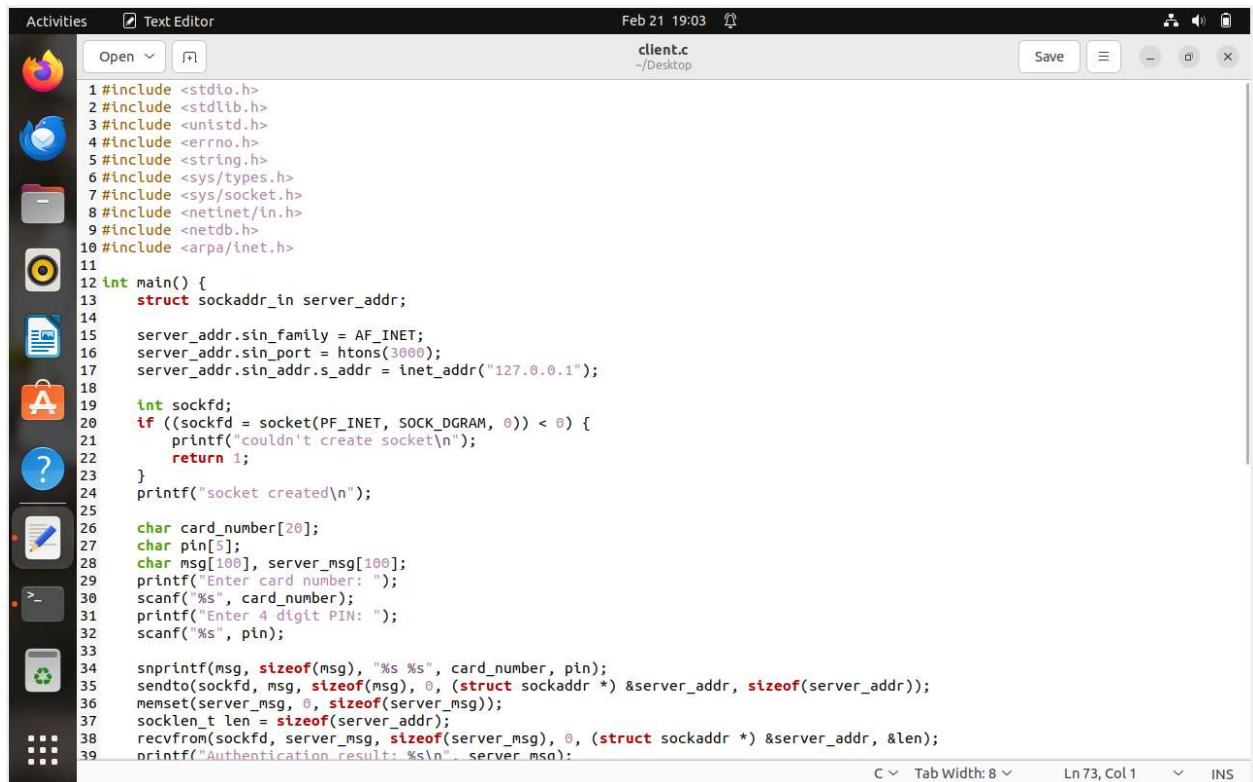
```
Activities Text Editor Feb 21 18:52
server.c
~/Desktop
Save

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <errno.h>
5 #include <string.h>
6 #include <sys/types.h>
7 #include <sys/socket.h>
8 #include <ctype.h>
9 #include <netinet/in.h>
10 #include <arpa/inet.h>
11
12 #define MAX_CLIENTS 5
13 int authenticate(char *card_number, char *pin) {
14     return 1;
15 }
16
17 int main() {
18     struct sockaddr_in server_addr;
19
20     server_addr.sin_family = AF_INET;
21     server_addr.sin_port = htons(3000);
22     server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
23
24     int sockfd;
25     if ((sockfd = socket(PF_INET, SOCK_DGRAM, 0)) < 0) {
26         printf("couldn't create socket\n");
27         return 1;
28     }
29     printf("socket created\n");
30
31     if (bind(sockfd, (struct sockaddr *) &server_addr, sizeof(server_addr)) < 0) {
32         printf("couldn't bind socket\n");
33         return 1;
34     }
35     printf("bind at port 3000\n");
36
37     struct sockaddr_in client_addr;
38     int client_addr_size = sizeof(client_addr);
39 }
```



```
27     return 1;
28 }
29 printf("socket created\n");
30
31 if (bind(sockfd, (struct sockaddr *) &server_addr, sizeof(server_addr)) < 0) {
32     printf("couldn't bind socket\n");
33     return 1;
34 }
35 printf("bind at port 3000\n");
36
37 struct sockaddr_in client_addr;
38 int client_addr_size = sizeof(client_addr);
39
40 while (1) {
41     char msg[100];
42     char card_number[20];
43     char pin[5];
44     recvfrom(sockfd, msg, sizeof(msg), 0, (struct sockaddr *) &client_addr, &client_addr_size);
45     sscanf(msg, "%s %s", card_number, pin);
46
47     if (authenticate(card_number, pin)) {
48         sendto(sockfd, "Authenticated", sizeof("Authenticated"), 0, (struct sockaddr *) &client_addr, client_addr_size);
49         while (1) {
50             recvfrom(sockfd, msg, sizeof(msg), 0, (struct sockaddr *) &client_addr, &client_addr_size);
51             printf("msg recv = %s\n", msg);
52
53             if (strcmp(msg, "4") == 0) {
54                 printf("exiting...\n");
55                 break;
56             }
57             sendto(sockfd, msg, sizeof(msg), 0, (struct sockaddr *) &client_addr, client_addr_size);
58         }
59     } else {
60         sendto(sockfd, "Authentication failed", sizeof("Authentication failed"), 0, (struct sockaddr *) &client_addr,
61             client_addr_size);
62     }
63 }
64 close(sockfd);
```

Client program



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <errno.h>
5 #include <string.h>
6 #include <sys/types.h>
7 #include <sys/socket.h>
8 #include <netinet/in.h>
9 #include <netdb.h>
10 #include <arpa/inet.h>
11
12 int main() {
13     struct sockaddr_in server_addr;
14
15     server_addr.sin_family = AF_INET;
16     server_addr.sin_port = htons(3000);
17     server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
18
19     int sockfd;
20     if ((sockfd = socket(PF_INET, SOCK_DGRAM, 0)) < 0) {
21         printf("couldn't create socket\n");
22         return 1;
23     }
24     printf("socket created\n");
25
26     char card_number[20];
27     char pin[5];
28     char msg[100], server_msg[100];
29     printf("Enter card number: ");
30     scanf("%s", card_number);
31     printf("Enter 4 digit PIN: ");
32     scanf("%s", pin);
33
34     sprintf(msg, "%s %s", card_number, pin);
35     sendto(sockfd, msg, sizeof(msg), 0, (struct sockaddr *) &server_addr, sizeof(server_addr));
36     memset(server_msg, 0, sizeof(server_msg));
37     socklen_t len = sizeof(server_addr);
38     recvfrom(sockfd, server_msg, sizeof(server_msg), 0, (struct sockaddr *) &server_addr, &len);
39     printf("Authentication result: %s\n", server_msg);
```

Activities Text Editor Feb 21 19:19

client.c
~/Desktop

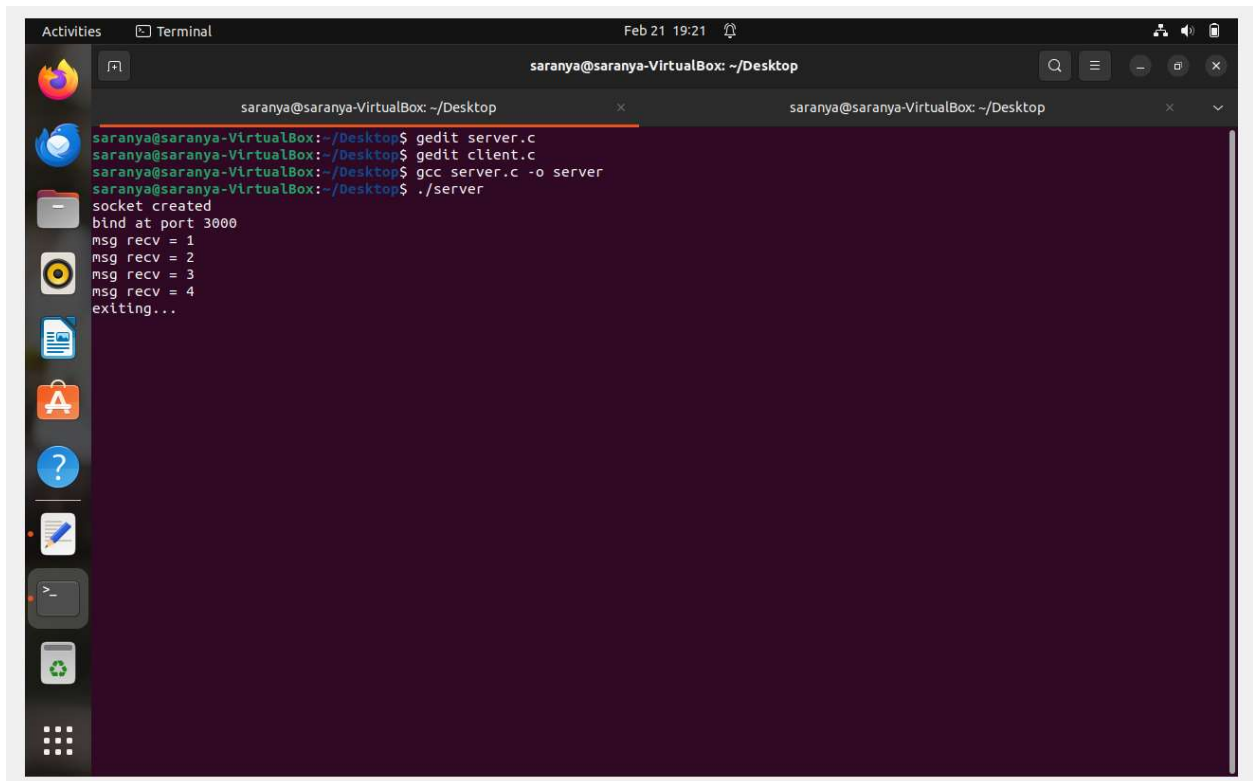
Save

```
32 scanf("%s", pin);
33
34 snprintf(msg, sizeof(msg), "%s %s", card_number, pin);
35 sendto(sockfd, msg, sizeof(msg), 0, (struct sockaddr *) &server_addr, sizeof(server_addr));
36 memset(server_msg, 0, sizeof(server_msg));
37 socklen_t len = sizeof(server_addr);
38 recvfrom(sockfd, server_msg, sizeof(server_msg), 0, (struct sockaddr *) &server_addr, &len);
39 printf("Authentication result: %s\n", server_msg);
40
41 if (strcmp(server_msg, "Authenticated") == 0) {
42     while (1) {
43         printf("Options:\n1. Deposit\n2. Withdrawal\n3. Check Balance\n4. Exit\n");
44         printf("Enter option: ");
45         scanf("%s", msg);
46         sendto(sockfd, msg, sizeof(msg), 0, (struct sockaddr *) &server_addr, sizeof(server_addr));
47
48         if (strcmp(msg, "4") == 0) {
49             printf("Exiting...\n");
50             break;
51         }
52
53         memset(server_msg, 0, sizeof(server_msg));
54         recvfrom(sockfd, server_msg, sizeof(server_msg), 0, (struct sockaddr *) &server_addr, &len);
55
56         if (strcmp(server_msg, "3") == 0){
57             printf("The balance is 1000000\n");
58         }
59
60         else if (strcmp(server_msg, "2") == 0){
61             printf("The withdrawn amount is 3000\n");
62         }
63
64         else if (strcmp(server_msg, "1") == 0){
65             printf("The deposited amount is 30000\n");
66         }
67     }
68 }
69
70 close(sockfd);
```

C Tab Width: 8 Ln 59, Col 16 INS

Output

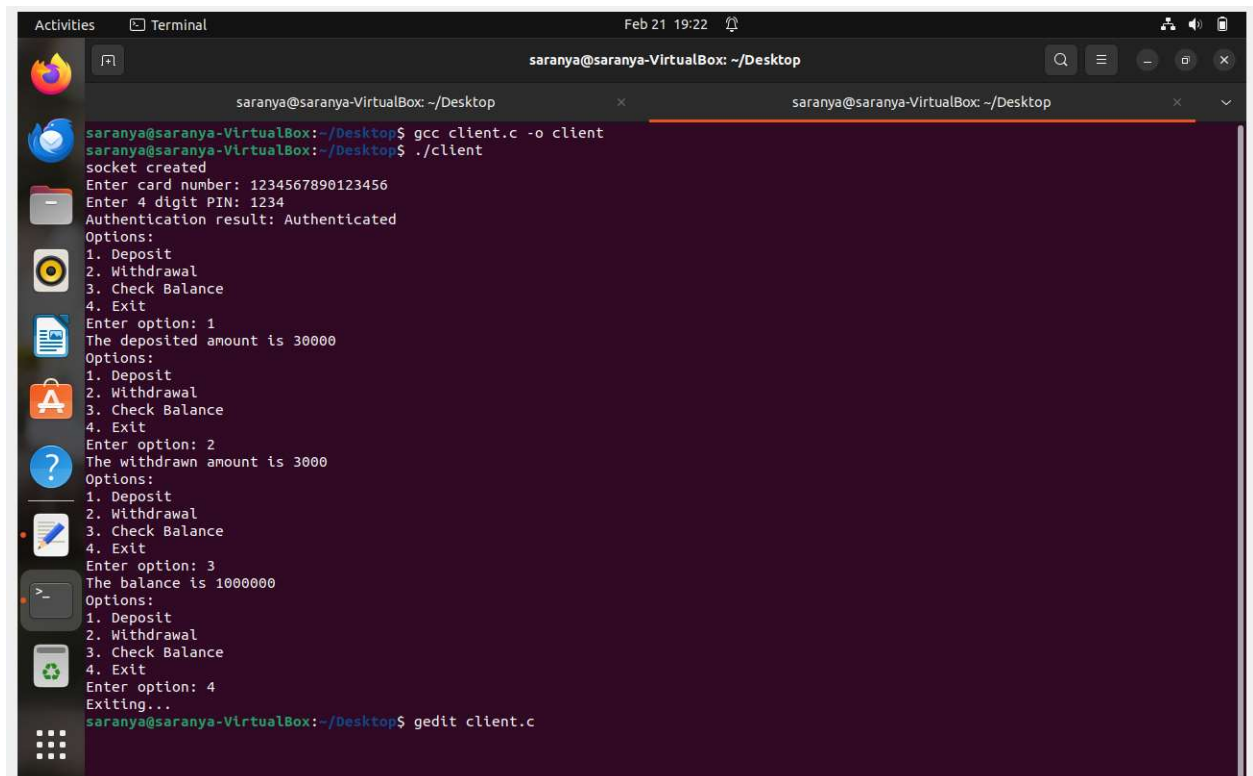
Server side terminal



The screenshot shows a terminal window titled "saranya@saranya-VirtualBox: ~/Desktop". The terminal displays the following output:

```
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 rcv = 1
msg rcv = 2
msg rcv = 3
msg rcv = 4
exiting...
```

Client side



The screenshot shows a terminal window titled "saranya@saranya-VirtualBox: ~/Desktop". The terminal displays the following output:

```
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