



**VIT**<sup>®</sup>  
**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

# **Computer Networks**

## **Laboratory**

### **Experiment - 04**

**Course code: BCSE308P**

**Faculty: N G Bhuvaneshwari**

**Name: Voladri Manohar Reddy**

**Registration No: 21BRS1177**

**Task: 01**

**Aim:** Write a client and server program using UDP socket, where the server echoes the message which is sent by the client in reverse form (that is, if client says hello, then server replies with olleh), and also print the client's IP address on server's console.

**Server:****Algorithm:**

- Create a UDP socket.
- Bind the socket to a specific address and port.
- Print "Server started. Waiting for client messages...".
- Repeat forever:
  - Receive data from the client.
  - Reverse the received message.
  - Print the client's IP address and the received message.
  - Send the reversed message back to the client.
- Close the socket.

**Code:**

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <arpa/inet.h>

#include <unistd.h>

#define SERVER_PORT 12345
```

```
#define BUFFER_SIZE 1024
```

```
void reverseMessage(char* message) {
```

```
    int length = strlen(message);
```

```
    int i, j;
```

```
    char temp;
```

```
    for (i = 0, j = length - 1; i < j; ++i, --j) {
```

```
        temp = message[i];
```

```
        message[i] = message[j];
```

```
        message[j] = temp;
```

```
    }
```

```
}
```

```
int main() {
```

```
    int sockfd;
```

```
    struct sockaddr_in server_addr, client_addr;
```

```
    socklen_t client_addr_len;
```

```
    char buffer[BUFFER_SIZE];
```

```
    // Create UDP socket
```

```
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
```

```
    if (sockfd < 0) {
```

```
        perror("Error creating socket");
```

```
        exit(1);
```

```
    }
```

```
memset(&server_addr, 0, sizeof(server_addr));

// Server configuration
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
server_addr.sin_port = htons(SERVER_PORT);

// Bind socket to address and port
if (bind(sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
    perror("Error binding socket");
    exit(1);
}

printf("Server started. Waiting for client messages...\n");

while (1) {
    client_addr_len = sizeof(client_addr);

    // Receive data from client
    ssize_t received_bytes = recvfrom(sockfd, buffer, BUFFER_SIZE, 0, (struct
sockaddr*)&client_addr, &client_addr_len);
    if (received_bytes < 0) {
        perror("Error receiving data from client");
        exit(1);
    }

    buffer[received_bytes] = '\0';
```

```
// Reverse the received message
reverseMessage(buffer);

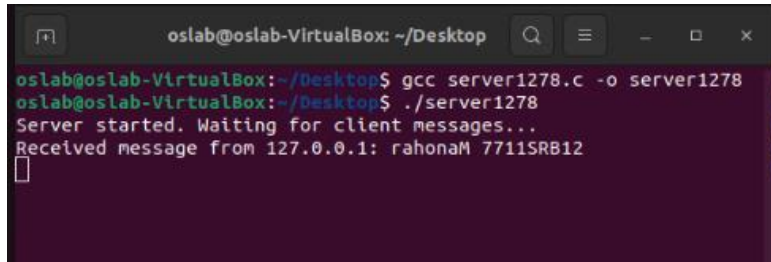
// Print client's IP address
char client_ip[INET_ADDRSTRLEN];
inet_ntop(AF_INET, &(client_addr.sin_addr), client_ip,
INET_ADDRSTRLEN);
printf("Received message from %s: %s\n", client_ip, buffer);

// Send reversed message back to client
if (sendto(sockfd, buffer, strlen(buffer), 0, (struct sockaddr*)&client_addr,
client_addr_len) < 0) {
    perror("Error sending data to client");
    exit(1);
}

// Close the socket
close(sockfd);

return 0;
}
```

**Output:**

A terminal window titled 'oslab@oslab-VirtualBox: ~/Desktop' showing the compilation and execution of a C program. The user runs 'gcc server1278.c -o server1278' and then './server1278'. The output shows 'Server started. Waiting for client messages...' followed by 'Received message from 127.0.0.1: rahonaM 7711SRB12' and a cursor on a new line.

```
oslab@oslab-VirtualBox: ~/Desktop
oslab@oslab-VirtualBox:~/Desktop$ gcc server1278.c -o server1278
oslab@oslab-VirtualBox:~/Desktop$ ./server1278
Server started. Waiting for client messages...
Received message from 127.0.0.1: rahonaM 7711SRB12
█
```

## Client:

### Algorithm:

- Create a UDP socket.
- Repeat forever:
  - Prompt the user to enter a message (or "quit" to exit).
  - If the message is "quit", break the loop.
  - Send the message to the server.
  - Receive the reversed message from the server.
  - Print the reversed message.
- Close the socket.

### Code:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <arpa/inet.h>

#include <unistd.h>

#define SERVER_IP "127.0.0.1"

#define SERVER_PORT 12345

#define BUFFER_SIZE 1024

int main() {
```

```
int sockfd;

struct sockaddr_in server_addr;

socklen_t server_addr_len;

char buffer[BUFFER_SIZE];


// Create UDP socket

sockfd = socket(AF_INET, SOCK_DGRAM, 0);

if (sockfd < 0) {
    perror("Error creating socket");
    exit(1);
}


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


// Server configuration

server_addr.sin_family = AF_INET;

server_addr.sin_addr.s_addr = inet_addr(SERVER_IP);

server_addr.sin_port = htons(SERVER_PORT);


while (1) {

    printf("Enter a message (or 'quit' to exit): ");

    fgets(buffer, BUFFER_SIZE, stdin);


    // Remove trailing newline character

    buffer[strcspn(buffer, "\n")] = '\0';


    if (strcmp(buffer, "quit") == 0) {
```

```
        break;
    }

    // Send message to server

    ssize_t sent_bytes = sendto(sockfd, buffer, strlen(buffer), 0, (struct
sockaddr*)&server_addr, sizeof(server_addr));

    if (sent_bytes < 0) {
        perror("Error sending data to server");
        exit(1);
    }

    server_addr_len = sizeof(server_addr);

    // Receive reversed message from server

    ssize_t received_bytes = recvfrom(sockfd, buffer, BUFFER_SIZE - 1, 0,
(struct sockaddr*)&server_addr, &server_addr_len);

    if (received_bytes < 0) {
        perror("Error receiving data from server");
        exit(1);
    }

    buffer[received_bytes] = '\0';

    printf("Reversed message: %s\n", buffer);
}

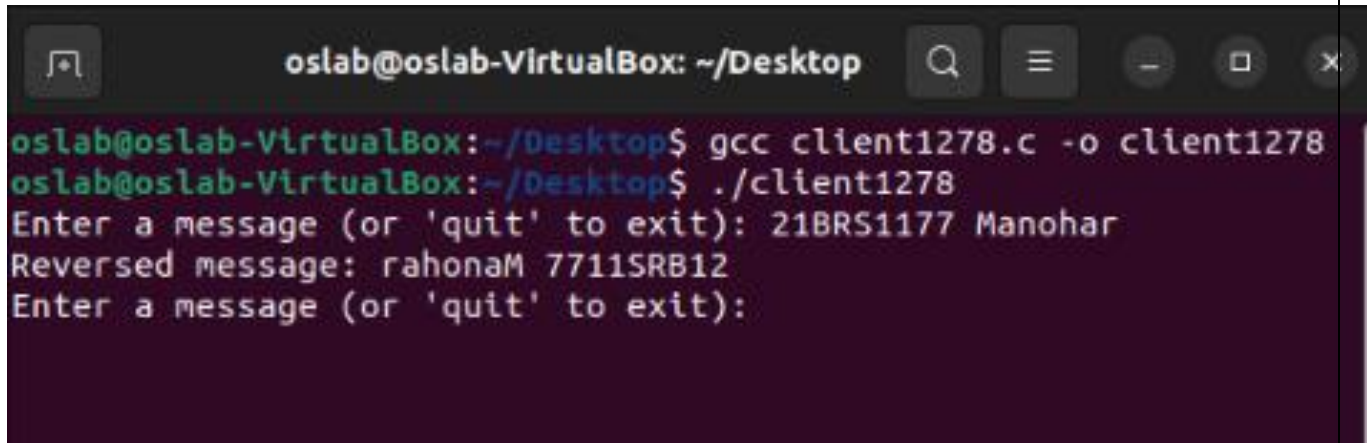
// Close the socket

close(sockfd);
```



```
    return 0;  
}
```

### Output:



```
oslab@oslab-VirtualBox: ~/Desktop  
oslab@oslab-VirtualBox:~/Desktop$ gcc client1278.c -o client1278  
oslab@oslab-VirtualBox:~/Desktop$ ./client1278  
Enter a message (or 'quit' to exit): 21BRS1177 Manohar  
Reversed message: rahonaM 77115RB12  
Enter a message (or 'quit' to exit):
```

## Task: 02

### Aim:

Develop a “Remote Calculator” application that works as follows: The client program inputs two integers and an arithmetic operation ('\*', '/', '%', '+', '-') from the user and sends these three values to the server side. The server does the given operation on the two integers and sends back the result of the operation to the client. Help the server to implement this scenario using connectionless sockets.

### Server:

#### Algorithm:

- Create a UDP socket.
- Configure the server address and port.
- Bind the socket to the server address and port.
- Print "Server started. Waiting for client messages..."
- Start an infinite loop to receive and process client requests.

- Receive data from the client.
- Parse the received data to obtain the two integers and arithmetic operation.
- Perform the requested arithmetic operation on the two integers.
- Get the client's IP address and port.
- Print the client connection status (IP address and port) and the answer.
- Convert the answer to a string.
- Send the answer back to the client.
- End the loop.
- Close the socket.

**Code:**

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

```
#include <unistd.h>
```

```
#define BUFLLEN 512
#define PORT 8888
```

```
void error(char *msg) {
    perror(msg);
    exit(1);
}
```

```
int main(void) {
```

```
struct sockaddr_in si_me, si_other;
int s, i, slen=sizeof(si_other);
char buf[BUFLEN];

// Create a UDP socket
if ((s=socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1) {
    error("socket");
}

// Bind the socket to PORT
memset((char *) &si_me, 0, sizeof(si_me));
si_me.sin_family = AF_INET;
si_me.sin_port = htons(PORT);
si_me.sin_addr.s_addr = htonl(INADDR_ANY);
if (bind(s, (struct sockaddr*)&si_me, sizeof(si_me)) == -1) {
    error("bind");
}

// Wait for messages from clients and perform calculations
while(1) {
    printf("Waiting for data...\n");
    fflush(stdout);

    // Receive message from client
    if (recvfrom(s, buf, BUFLen, 0, (struct sockaddr *) &si_other, &slen) == -1) {
        error("recvfrom");
    }

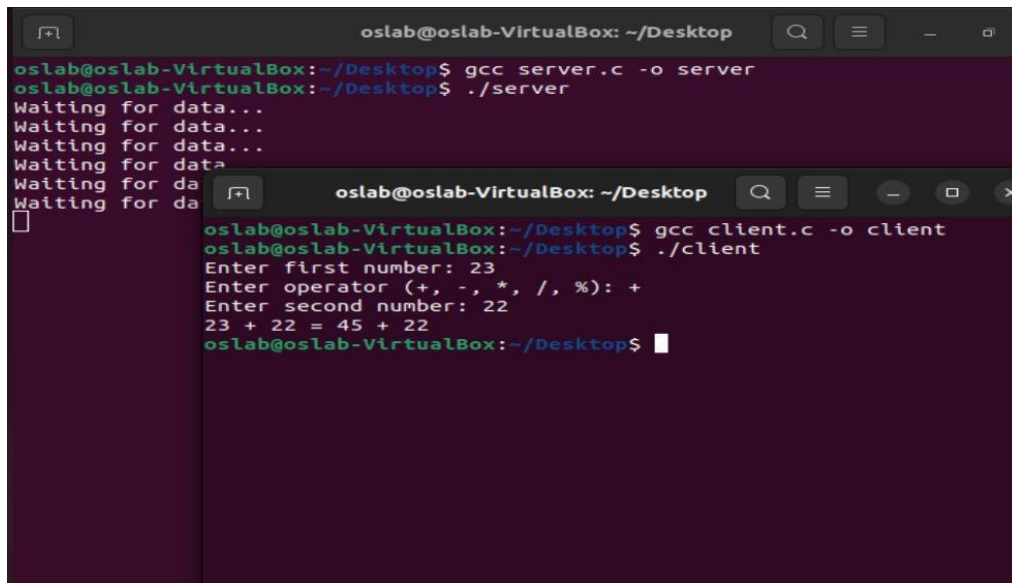
    // Parse message to obtain two integers and an operator
    int num1, num2;
    char op;
    if (sscanf(buf, "%d %c %d", &num1, &op, &num2) != 3) {
        printf("Invalid message format: %s\n", buf);
        continue;
    }
}
```

```
// Calculate the result of the operation
int result;
switch(op) {
    case '+':
        result = num1 + num2;
        break;
    case '-':
        result = num1 - num2;
        break;
    case '*':
        result = num1 * num2;
        break;
    case '/':
        if (num2 == 0) {
            printf("Division by zero\n");
            continue;
        }
        result = num1 / num2;
        break;
    case '%':
        result = num1 % num2;
        break;
    default:
        printf("Invalid operator: %c\n", op);
        continue;
}

// Send the result back to the client
sprintf(buf, "%d", result);
if (sendto(s, buf, strlen(buf), 0, (struct sockaddr*) &si_other, slen) == -1) {
    error("sendto");
}

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

## Output:



```
oslab@oslab-VirtualBox: ~/Desktop
oslab@oslab-VirtualBox:~/Desktop$ gcc server.c -o server
oslab@oslab-VirtualBox:~/Desktop$ ./server
Waiting for data...
Waiting for data...
Waiting for data...
Waiting for data...
Waiting for data...
Waiting for data...
oslab@oslab-VirtualBox:~/Desktop$ gcc client.c -o client
oslab@oslab-VirtualBox:~/Desktop$ ./client
Enter first number: 23
Enter operator (+, -, *, /, %): +
Enter second number: 22
23 + 22 = 45 + 22
oslab@oslab-VirtualBox:~/Desktop$
```

## Client:

### Algorithm:

- Create a UDP socket.
- Configure the server address and port.
- Start an infinite loop to input and send requests to the server.
- Prompt the user to enter the first number.
- Read the first number from the user.
- Prompt the user to enter the second number.
- Read the second number from the user.
- Prompt the user to enter the arithmetic operation.
- Read the arithmetic operation from the user.
- Create a message string with the two numbers and the operation.
- Send the message to the server.
- Receive the result from the server.

- Convert the result string to an integer.
- Print the result.
- End the loop.

### **Code:**

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

#include <unistd.h>

#define BUFLen 512
#define PORT 8888

void error(char *msg) {
    perror(msg);
    exit(1);
}

int main(void) {
    struct sockaddr_in si_me, si_other;
    int s, i, slen=sizeof(si_other);
    char buf[BUFLen];

    // Create a UDP socket
    if ((s=socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1) {
        error("socket");
    }
```

```
}
```

```
// Bind the socket to PORT
memset((char *) &si_me, 0, sizeof(si_me));
si_me.sin_family = AF_INET;
si_me.sin_port = htons(PORT);
si_me.sin_addr.s_addr = htonl(INADDR_ANY);
if (bind(s, (struct sockaddr*)&si_me, sizeof(si_me)) == -1) {
    error("bind");
}
```

```
// Wait for messages from clients and perform calculations
while(1) {
    printf("Waiting for data...\n");
    fflush(stdout);
```

```

    // Receive message from client
    if (recvfrom(s, buf, BUFLen, 0, (struct sockaddr *) &si_other, &slen) == -1) {
        error("recvfrom");
    }
```

```

    // Parse message to obtain two integers and an operator
    int num1, num2;
    char op;
    if (sscanf(buf, "%d %c %d", &num1, &op, &num2) != 3) {
        printf("Invalid message format: %s\n", buf);
        continue;
    }
```

```

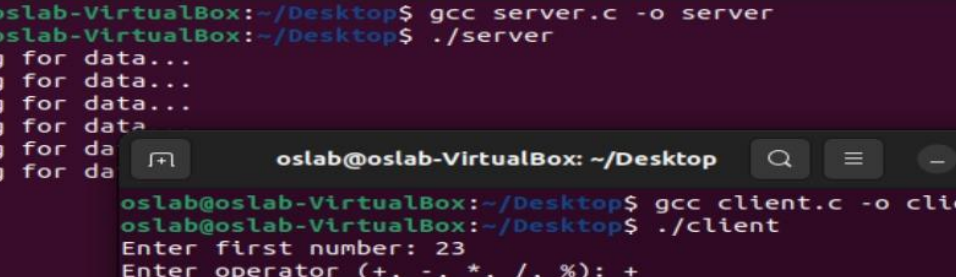
    // Calculate the result of the operation
    int result;
    switch(op) {
        case '+':
            result = num1 + num2;
            break;
```

```
    case '-':
        result = num1 - num2;
        break;
    case '*':
        result = num1 * num2;
        break;
    case '/':
        if (num2 == 0) {
            printf("Division by zero\n");
            continue;
        }
        result = num1 / num2;
        break;
    case '%':
        result = num1 % num2;
        break;
    default:
        printf("Invalid operator: %c\n", op);
        continue;
}
sprintf(buf, "%d", result);
if (sendto(s, buf, strlen(buf), 0, (struct sockaddr*) &si_other, slen) == -1) {
    error("sendto");
}
}

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

**Output:**





```
oslab@oslab-VirtualBox: ~/Desktop
oslab@oslab-VirtualBox:~/Desktop$ gcc server.c -o server
oslab@oslab-VirtualBox:~/Desktop$ ./server
Waiting for data...
Waiting for data...
Waiting for data...
Waiting for data...
Waiting for data...
Waiting for data...

oslab@oslab-VirtualBox: ~/Desktop
oslab@oslab-VirtualBox:~/Desktop$ gcc client.c -o client
oslab@oslab-VirtualBox:~/Desktop$ ./client
Enter first number: 23
Enter operator (+, -, *, /, %): +
Enter second number: 22
23 + 22 = 45 + 22
oslab@oslab-VirtualBox:~/Desktop$
```