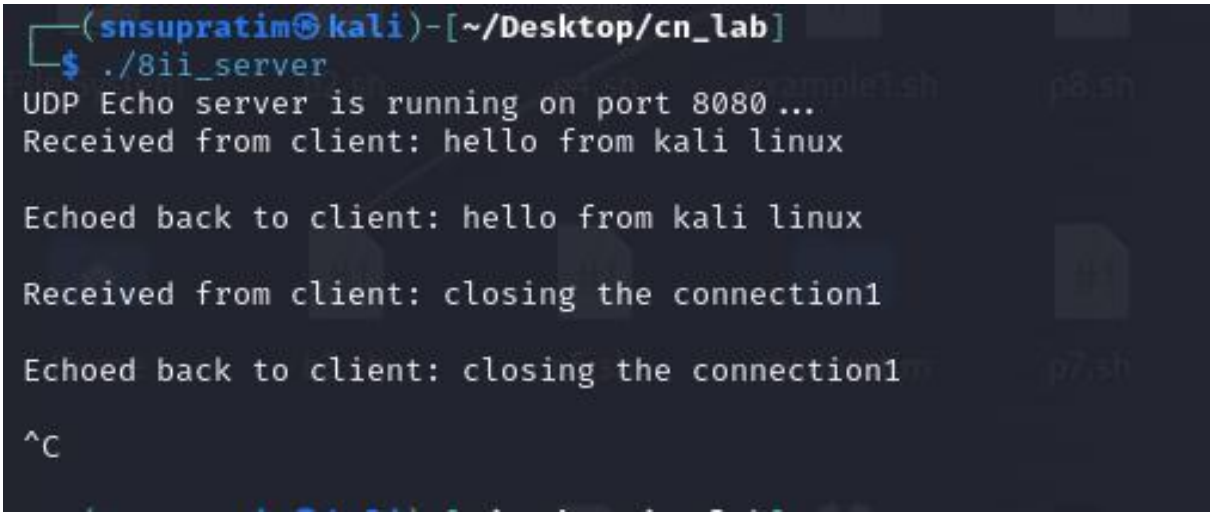
COMPUTER NETWORK LAB ASSIGNMENTS :
SUPRATIM NAG/CSE-AIM/22/57 :

Implementation of IPC in concurrent modalities:

II. Write a program to develop a concurrent echo server using UDP socket by implementing at least two clients.

SERVER SIDE CODE :

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

#define PORT 8080
#define BUFFER_SIZE 1024

int main() {
    int sockfd;
    struct sockaddr_in server_addr, client_addr;
    socklen_t client_len = sizeof(client_addr);
    char buffer[BUFFER_SIZE];

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

    // Clear and configure server address structure
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);

    // Bind the socket to the specified port
    if (bind(sockfd, (const struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("Bind failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }

    printf("UDP Echo server is running on port %d...\n", PORT);

    // Main loop to handle incoming messages
    while (1) {
        // Receive message from a client
        int len = recvfrom(sockfd, buffer, BUFFER_SIZE, 0, (struct sockaddr *)&client_addr, &client_len);
        if (len < 0) {
            perror("Receive failed");
            continue;
        }

        buffer[len] = '\0';  // Null-terminate the received string
        printf("Received from client: %s\n", buffer);

        // Send the received message back to the client
        sendto(sockfd, buffer, len, 0, (struct sockaddr *)&client_addr, client_len);
        printf("Echoed back to client: %s\n", buffer);
    }

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


(snsupratim㉿kali)-[~/Desktop/cn_lab]
$ ./8ii_server
UDP Echo server is running on port 8080 ...
Received from client: hello from kali linux
Echoed back to client: hello from kali linux
Received from client: closing the connection1
Echoed back to client: closing the connection1
^C


(snsupratim㉿kali)-[~/Desktop/cn_lab]
$ gcc 8ii_client.c -o 8ii_client
(snsupratim㉿kali)-[~/Desktop/cn_lab]
$ ./8ii_client
Enter messages to send to the server (type 'exit' to quit):
Client: hello from kali linux
Server: hello from kali linux
Client: closing the connection1
Server: closing the connection1
Client: ^C
(snsupratim㉿kali)-[~/Desktop/cn_lab]
$

CLIENT SIDE CODE :

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

#define SERVER_IP "127.0.0.1"  // IP address of the server
#define PORT 8080
#define BUFFER_SIZE 1024

int main() {
    int sockfd;
    struct sockaddr_in server_addr;
    char buffer[BUFFER_SIZE];
    socklen_t server_len = sizeof(server_addr);

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

    // Configure server address structure
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr(SERVER_IP);

    printf("Enter messages to send to the server (type 'exit' to quit):\n");

    while (1) {
        printf("Client: ");
        fgets(buffer, BUFFER_SIZE, stdin);

        // Send message to the server
        sendto(sockfd, buffer, strlen(buffer), 0, (struct sockaddr *)&server_addr, server_len);

        // If client sends "exit", break the loop and close the client
        if (strncmp(buffer, "exit", 4) == 0) {
            printf("Exiting...\n");
            break;
        }

        // Receive echo from the server
        int len = recvfrom(sockfd, buffer, BUFFER_SIZE, 0, NULL, NULL);
        if (len < 0) {
            perror("Receive failed");
            break;
        }

        buffer[len] = '\0';  // Null-terminate the received string
        printf("Server: %s", buffer);
    }

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