```
//CSM19031

#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <time.h>
int init(int socket_desc, struct sockaddr_in client_addr, int
client_struct_length)
{
    char server_message[2000], client_message[2000];
    char c2[1200] = "\n Select options to know who is the current
President of that country : \n 1. India \n 2. Pakisthan \n 3. Srilanka
\n 4. Bangladesh \n 5. USA \n 6. Exit \n";
    time_t t;
    while (1)
    {
        memset(server_message, '\0', sizeof(server_message));
        memset(client_message, '\0', sizeof(client_message));
        int n;
        // Receive client's message:
        n = recvfrom(socket_desc, client_message,
sizeof(client_message), 0,
                     (struct sockaddr *)&client_addr,
&client_struct_length);
        if (n < 0)
        {
            printf("Couldn't receive\n");
            return -1;
        }
        client_message[n] = '\0';
        time(&t);
        printf("Received message from IP: %s and port: %i\t",
inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port));

        printf(" : [%s] Client : %s\n", ctime(&t), client_message);

        char c[100];
        char *ptr;
        // Respond to client:
        //  strcpy(server_message, client_message);
        int choice = (int)strtol(client_message, &ptr, 10);
        switch (choice)
        {
        case 1:
            strcpy(c, " Ram Nath Kovind \n Press any random key to see
the main menu ");
            if (sendto(socket_desc, c, strlen(c), 0,
                       (struct sockaddr *)&client_addr,
client_struct_length) < 0)
            {
                printf("Unable to send message\n");
                return -1;
            }
            break;
        case 2:
            strcpy(c, "Arif Alui \n Press any random key to see the
main menu ");
            if (sendto(socket_desc, c, strlen(c), 0,
                       (struct sockaddr *)&client_addr,
client_struct_length) < 0)
            {
                printf("Unable to send message\n");
                return -1;
            }
            break;
        case 3:
            strcpy(c, " Gotabaya Rajapaksa \n Press any random key to
see the main menu ");
            if (sendto(socket_desc, c, strlen(c), 0,
                       (struct sockaddr *)&client_addr,
client_struct_length) < 0)
            {
                printf("Unable to send message\n");
                return -1;
            }
            break;

        case 4:
            strcpy(c, " Abdul Hamid \n Press any random key to see the
main menu");
            if (sendto(socket_desc, c, strlen(c), 0,
                       (struct sockaddr *)&client_addr,
client_struct_length) < 0)
            {
                printf("Unable to send message\n");
                return -1;
            }

            break;

        case 5:
            strcpy(c, "Joe Biden \n Press any random key to see the
main menu");
            if (sendto(socket_desc, c, strlen(c), 0,
                       (struct sockaddr *)&client_addr,
client_struct_length) < 0)
            {
                printf("Unable to send message\n");
                return -1;
            }
            break;
        //
        case 6:
        //    strcpy(c, " Elon Musk \n Press any random key to see the
main menu");
        //    if (sendto(socket_desc, c, strlen(c), 0,
        //               (struct sockaddr *)&client_addr,
client_struct_length) < 0)
        //    {
        //        printf("Unable to send message\n");
        //        return -1;
        //    }
            break;
        default:
            if (sendto(socket_desc, c2, strlen(c2), 0,
                       (struct sockaddr *)&client_addr,
client_struct_length) < 0)
            {
                printf("Unable to send message\n");
                return -1;
            }
            break;
        }
        memset(server_message, '\0', sizeof(server_message));
        memset(client_message, '\0', sizeof(client_message));
        sleep(1);
        if (sendto(socket_desc, &t, sizeof(&t), 0,
                   (struct sockaddr *)&client_addr,
client_struct_length) < 0)
        {
            printf("Can't send\n");
            return -1;
        }
    }
    return 0;
}

int main(int argc, char *argv[])
{
    int socket_desc;
    struct sockaddr_in server_addr, client_addr;
    char server_message[2000], client_message[2000];
    int client_struct_length = sizeof(client_addr);

    if (argc != 2)
    {
        printf("Enter IP of the server as command line argument and
run again \n");
        exit(1);
    }
    // Clean buffers:
    memset(server_message, '\0', sizeof(server_message));
    memset(client_message, '\0', sizeof(client_message));

    // Create UDP socket:
    socket_desc = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);

    if (socket_desc < 0)
    {
        printf("Error while creating socket\n");
        return -1;
    }
    printf("Socket created successfully\n");

    // Set port and IP:
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(2000);
    server_addr.sin_addr.s_addr = inet_addr(argv[1]);
    // Bind to the set port and IP:
    if (bind(socket_desc, (struct sockaddr *)&server_addr,
sizeof(server_addr)) < 0)
    {
        printf("Couldn't bind to the port\n");
        return -1;
    }
    printf("Done with binding\n");

    printf("Listening for incoming messages...\n\n");

    init(socket_desc, client_addr, client_struct_length);

    // Close the socket:
    close(socket_desc);

    return 0;
}
```