

TEST DOCUMENT

ASSIGNMENT QUESTION :

Design and implement an emergency server. It should answer the sample clients questions:

1. Police Station number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery
6. Blood Bank Number

Constraints:

1. Clients do not know the IP address or hostname of the emergency server.
2. Emergency services should be reliable.
3. You can assume an Emergency server running on a particular port number.
4. Multiple clients may send queries parallelly to the emergency server.

Server Code :

```
GNU nano 6.2 server.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <fcntl.h> // For fcntl()
#include <errno.h> // For errno and EINTR
#include <sys/select.h> // For select()

#define PORT 5555
#define BUFFER_SIZE 1024

// Emergency service data
typedef struct {
    char *service;
    char *number;
} EmergencyService;

EmergencyService emergency_services[] = {
    {"Police Station Number", "911"},
    {"Ambulance Number", "912"},
    {"Fire Station Number", "913"},
    {"Vehicle Repair Number", "914"},
    {"Food Delivery Number", "915"},
    {"Blood Bank Number", "916"},
};

// Function to get emergency number for a given service
const char* get_emergency_number(const char *service) {
    static char response[BUFFER_SIZE]; // Static to retain memory after function call

    for (int i = 0; i < sizeof(emergency_services) / sizeof(EmergencyService); i++) {
        if (strcmp(service, emergency_services[i].service) == 0) {
            sprintf(response, "The %s is %s", service, emergency_services[i].number);
            return response;
        }
    }
    return "Invalid request. Service not available.";
}

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

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

    // Bind to the port
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    if (bind(server_socket, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
        perror("binding failed");
        close(server_socket);
        exit(EXIT_FAILURE);
    }

    printf("Emergency server started on port %d, waiting for client requests...\n", PORT);

    // Loop to receive and send data
    while (1) {
        // Use select to wait for data on the socket
        fd_set read_fds;
        FD_ZERO(&read_fds);
        FD_SET(server_socket, &read_fds);

        // Use select to wait for data on the socket with a timeout of 1 second
        int timeout = 1;
        timeout.tv_sec = 1;
        timeout.tv_usec = 0;

        int activity = select(max_fd + 1, &read_fds, NULL, NULL, &timeout);

        if (activity < 0 || errno != EINTR) {
            perror("select error");
        }

        // Check if there is data to read from the socket
        if (FD_ISSET(server_socket, &read_fds)) {
            // Receive data from client
            int bytes_received = recvfrom(server_socket, buffer, BUFFER_SIZE - 1, 0,
                                         (struct sockaddr*)&client_addr, &client_addr_len);

            if (bytes_received < 0) {
                perror("error receiving data");
                continue;
            }

            buffer[bytes_received] = '\0'; // Null-terminate the received data
            printf("Received request: %s from client %s\n", buffer, inet_ntoa(client_addr.sin_addr));

            // Get the response based on the client request
            const char *response = get_emergency_number(buffer);

            // Send response back to the client
            sendto(server_socket, response, strlen(response), 0,
                  (struct sockaddr*)&client_addr, client_addr_len);
        } else {
            // No data available to read, server is still responsive
            printf("No data available to read, server is still responsive.\n");
        }
    }

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

Fig : Server Code

Client Code :

```
GNU nano 6.2 client.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 5555
#define MAX_PACKET_SIZE "255-255-255-255"
#define BUFFER_SIZE 1024

// Array to hold service names
const char *service_list[] = {
    "Police Station Number",
    "Ambulance Number",
    "Fire Station Number",
    "Vehicle Repair Number",
    "Food Delivery Number",
    "Blood Bank Number",
};

void query_emergency_server(const char *service) {
    int client_socket;
    struct sockaddr_in server_addr;
    char buffer[BUFFER_SIZE];
    socklen_t server_addr_len = sizeof(server_addr);

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

    // Enable broadcast option for the socket
    int broadcast_enable = 1;
    if (setsockopt(client_socket, SOL_SOCKET, SO_BROADCAST, &broadcast_enable, sizeof(broadcast_enable)) < 0) {
        perror("error setting socket options for broadcast");
        close(client_socket);
        exit(EXIT_FAILURE);
    }

    // Set the server address
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    inet_aton("127.0.0.1", &server_addr.sin_addr);

    // Send the request to the server
    sendto(client_socket, service, strlen(service), 0,
          (struct sockaddr*)&server_addr, sizeof(server_addr));

    // Receive the response from the server
    int bytes_received = recvfrom(client_socket, buffer, BUFFER_SIZE - 1, 0,
                                  (struct sockaddr*)&server_addr, &server_addr_len);

    if (bytes_received < 0) {
        perror("error receiving data from server");
    } else {
        buffer[bytes_received] = '\0'; // Null-terminate the received data
        printf("Response from server: %s\n", buffer);
    }

    close(client_socket);
}

int main() {
    int user_choice;

    // Loop to continuously prompt the user for service requests
    while (1) {
        // Display the service options
        printf("\nEnter the service you need:\n");
        printf("1. Police Station Number\n");
        printf("2. Ambulance Number\n");
        printf("3. Fire Station Number\n");
        printf("4. Vehicle Repair Number\n");
        printf("5. Food Delivery Number\n");
        printf("6. Blood Bank Number\n");
        printf("Type '0' to exit\n");
        printf("Enter your choice: ");

        // Get the user's choice
        if (scanf("%d", &user_choice) != 1) {
            // Handle invalid input
            printf("Invalid input. Please enter a number between 0 and 6.\n");
            // Clear the input buffer
            while (getchar() != '\n'); // discard invalid input
            continue;
        }

        // Check if the user wants to exit the program
        if (user_choice == 0) {
            printf("Exiting the client program.\n");
            break;
        }

        // Validate user input and send the request if valid
        if (user_choice >= 1 && user_choice <= 6) {
            query_emergency_server(service_list[user_choice - 1]);
        } else {
            printf("Invalid choice. Please enter a number between 1 and 6.\n");
        }
    }

    return 0;
}
```

Fig : Client Code

TEST CASES :

1. On starting the server first, it displays the message:
'Emergency server started on port 5555, waiting for client requests...'

```
pratyush_parth@LAPTOP-VEBSJD0U:~/Emergency$ ./server
Emergency server started on port 5555, waiting for client requests...
```

2. On starting the client, it displays the following message in the menu-driven option format where the client is asked to write the choice from 1 to 6 and 0 to terminate.

```
pratyush_parth@LAPTOP-VEBSJD0U:~/Emergency$ ./client
Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
Enter your choice: █
```

3. On the server side, if there is no data available to read from the socket, the server displays a message indicating that it is still responsive.

```
pratyush_parth@LAPTOP-VEBSJD0U:~/Emergency$ ./server
Emergency server started on port 5555, waiting for client requests...
No data available to read; server is still responsive.
```

4. Demonstration of multiple clients on a single server, namely 1 server and 2 clients (Client 1 with IP Address 172.16.108.50 and Client 2 with IP Address 172.16.108.90)

```
pratyush-parth@pratyush-parth-VirtualBox:~/Downloads/Emergency(1)/Emergency$ ./server
Emergency server started on port 5555, waiting for client requests...
No data available to read; server is still responsive.
No data available to read; server is still responsive.
No data available to read; server is still responsive.
No data available to read; server is still responsive.
No data available to read; server is still responsive.
Received request: Police Station Number from client 172.16.108.50:58051
Received request: Police Station Number from client 172.16.108.90:45000
Received request: Ambulance Number from client 172.16.108.50:56989
```

5. On the client side, output and response from the server on choosing option 1.

```
pratyush_parth@LAPTOP-VEBSJD0U:~/Emergency$ ./client
Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
Enter your choice: 1
Response from server: The Police Station Number is 911
```

6. On the server side, output when the client makes a request of contact number of option 1.

```
pratyush_parth@LAPTOP-VEBSJD0U:~/Emergency$ ./server
Emergency server started on port 5555, waiting for client requests...
No data available to read; server is still responsive.
Received request: Police Station Number from client 172.22.153.162:39380
```

7. On the client side, output and response from the server on choosing option 6.

```
pratyush_parth@LAPTOP-VEBSJD0U:~/Emergency$ ./client
Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
Enter your choice: 6
Response from server: The Blood Bank Number is 916
```

8. On the server side, output when the client makes second request of contact number of option 6. Also, it demonstrates client with repeated IP and port sending multiple requests.

```
pratyush_parth@LAPTOP-VEBSJD0U:~/Emergency$ ./server
Emergency server started on port 5555, waiting for client requests...
No data available to read; server is still responsive.
Received request: Police Station Number from client 172.22.153.162:55092
Received request: Blood Bank Number from client 172.22.153.162:53554
```

9. On starting the client first before starting the server and choosing option 2.

```
pratyush_parth@LAPTOP-VEBSJD0U:~/Emergency_System$ ./client
Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
Enter your choice: 2
```

10. On the server side, this is the output on the request of client.

```
pratyush_parth@LAPTOP-VEBSJD0U:~/Emergency_System$ ./server
Emergency server started on port 5555, waiting for client requests...
Received request: Ambulance Number from client 127.0.0.1:60286
```

11. On the client side, this is the response from the server.

```
pratyush_parth@LAPTOP-VEBSJD0U:~/Emergency_System$ ./client
Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
Enter your choice: 2
Response from server: The Ambulance Number is 912
```

12. On the client side, output when the client chooses a text instead of appropriate numbers within 0 to 6.

```
pratyush_parth@LAPTOP-VEBSJD0U:~/Emergency$ ./client
Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
Enter your choice: abcd
Invalid input. Please enter a number between 0 and 6.
```

13. On the client side, output when the client chooses a number outside the range 1 to 6.

```
pratyush_parth@LAPTOP-VEBSJD0U:~/Emergency$ ./client
Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
Enter your choice: 8
Invalid choice. Please enter a number between 1 and 6.
```

14. On the client side, on choosing 0, client terminates .

```
pratyush_parth@LAPTOP-VEBSJD0U:~/Emergency$ ./client
Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
Enter your choice: 0
Exiting the client program.
```

Apart from these cases,

- In case of any network issue, when the server is temporarily unreachable and client sends a request, the client handles the error gracefully and displays the message 'Error receiving data from server.'
- On checking the server timeout test, if the client makes a request after a period of inactivity, the server correctly responds to the request without requiring a restart.
- In case of test with maximum data size, i.e., buffer size limit, the server handles the buffer size without crashing, responding correctly if the message size is within the limit.

DEMONSTRATION OF HANDLING MULTIPLE CLIENTS ON A SINGLE SERVER

We successfully handled multiple clients on different systems from a single server, where we had 1 server and 3 clients, namely Client 1 with IP Address 172.16.108.50, Client 2 with IP Address 172.16.108.90 and Client 3 with IP Address 172.16.108.53 respectively, which we can clearly observe from the further screenshots in the next page.


```
Oct 21 00:18
pratyush-parth@pratyush-parth-VirtualBox: ~/Downloads/Emergency(1)/Emergency
pratyush-parth@pratyush-parth-VirtualBox: ~/Downloads/Emergency(1)/Emergency$ ./server
Emergency server started on port 5555, waiting for client requests...
No data available to read; server is still responsive.
No data available to read; server is still responsive.
No data available to read; server is still responsive.
No data available to read; server is still responsive.
No data available to read; server is still responsive.
Received request: Police Station Number from client 172.16.108.50:58851
Received request: Ambulance Number from client 172.16.108.50:45000
Received request: Ambulance Number from client 172.16.108.50:56989
No data available to read; server is still responsive.
No data available to read; server is still responsive.
Received request: Ambulance Number from client 172.16.108.50:35251
Received request: Fire Station Number from client 172.16.108.50:56910
Received request: Fire Station Number from client 172.16.108.50:48196
Received request: Food Delivery Number from client 172.16.108.50:41768
Received request: Vehicle Repair Number from client 172.16.108.50:37538
Received request: Blood Bank Number from client 172.16.108.50:60366
No data available to read; server is still responsive.
Received request: Vehicle Repair Number from client 172.16.108.50:51355
Received request: Police Station Number from client 172.16.108.50:43129
No data available to read; server is still responsive.
Received request: Ambulance Number from client 172.16.108.50:51247
No data available to read; server is still responsive.
Received request: Food Delivery Number from client 172.16.108.50:44252
Received request: Food Delivery Number from client 172.16.108.50:43565
Received request: Fire Station Number from client 172.16.108.50:38838
Received request: Food Delivery Number from client 172.16.108.50:59697
No data available to read; server is still responsive.
Received request: Police Station Number from client 172.16.108.50:46543
Received request: Vehicle Repair Number from client 172.16.108.50:34609
Received request: Ambulance Number from client 172.16.108.50:59443
Received request: Fire Station Number from client 172.16.108.50:59806
Received request: Blood Bank Number from client 172.16.108.50:37865
Received request: Vehicle Repair Number from client 172.16.108.50:58050
Received request: Police Station Number from client 172.16.108.50:42829
Received request: Food Delivery Number from client 172.16.108.50:68880
Received request: Fire Station Number from client 172.16.108.53:57424
Received request: Fire Station Number from client 172.16.108.50:46159
Received request: Blood Bank Number from client 172.16.108.50:44865
No data available to read; server is still responsive.
No data available to read; server is still responsive.
No data available to read; server is still responsive.
No data available to read; server is still responsive.
No data available to read; server is still responsive.
No data available to read; server is still responsive.
Received request: Blood Bank Number from client 172.16.108.50:50607
No data available to read; server is still responsive.
No data available to read; server is still responsive.
No data available to read; server is still responsive.
No data available to read; server is still responsive.
No data available to read; server is still responsive.
Received request: Vehicle Repair Number from client 172.16.108.53:47786
```

Fig: Server handling multiple clients parallelly

```
Oct 21 00:18
pratyush-parth@pratyush-parth-VirtualBox: ~/Downloads/Emergency(1)/Emergency
pratyush-parth@pratyush-parth-VirtualBox: ~/Downloads/Emergency(1)/Emergency$ ./client
Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
Enter your choice: 1
Response from server: The Police Station Number is 911

Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
Enter your choice: 2
Response from server: The Ambulance Number is 912

Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
Enter your choice: 6
Response from server: The Blood Bank Number is 916

Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
Enter your choice: 7
Invalid choice. Please enter a number between 1 and 6.

Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
```

Fig: Client 1 with IP Address 172.16.108.50


```
Oct 21 00:20
aayush@aayush-VirtualBox: ~/Downloads/Emergency-2/Emergency

client client.c server server.c
aayush@aayush-VirtualBox:~/Downloads/Emergency-2/Emergency$ gcc client.c -o client
aayush@aayush-VirtualBox:~/Downloads/Emergency-2/Emergency$ ./client

Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
Enter your choice: 1
Response from server: The Police Station Number is 911

Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
Enter your choice: 2
Response from server: The Ambulance Number is 912

Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
```

Fig: Client 2 with IP Address 172.16.108.90

```
Oct 20 22:55
client.c - Emergency - Visual Studio Code

File Edit Selection View Go Run Terminal Help
EXPLORER PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
EMERGENCY
  E d
  E client
  C client.c
  E server
  C server.c

Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
Enter your choice: 5
Response from server: The Food Delivery Number is 915

Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
Enter your choice: 3
Response from server: The Fire Station Number is 913

Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
Enter your choice: 4
Response from server: The Vehicle Repair Number is 914

Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
Enter your choice: 1
Response from server: The Police Station Number is 911

Enter the service you need:
1. Police Station Number
2. Ambulance Number
3. Fire Station Number
4. Vehicle Repair Number
5. Food Delivery Number
6. Blood Bank Number
Type '0' to exit
```

Fig: Client 3 with IP Address 172.16.108.53

*****THANK*****
*****YOU*****