Name: Rajkumar B L

Reg.No: 2047120

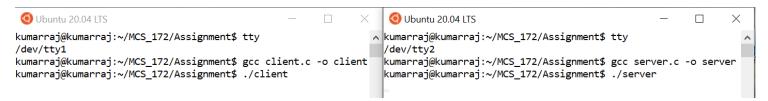
Course: MCS 172 - Unix Assignment

## Question 02: Socket Inter-Process Communication

A socket is an IPC method of point to point two-way communication between two processes: connecting two nodes on a network. To implement this, I created an application where two separate processes: server and client, communicate with each other through a socket IPC method. Here, the communication context is that a client(Rajkumar) establishes a connection with the server(Pizza-Hut) and orders pizza for his family members to celebrate Christmas.

- Creating and naming socket: socket(), bind()
- Establish Connection: listen(), accept(), connect()
- Communication: read(), write(), send(), recv()

## 1. Server.c and client.c are compiled in two separate terminals



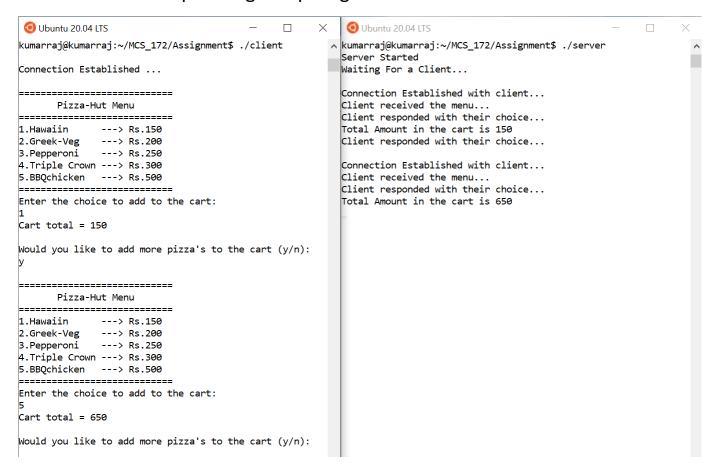
## 2. Connection is established, the server sends the menu to the client

```
O Ubuntu 20.04 LTS
                                                              O Ubuntu 20.04 LTS
                                                            kumarraj@kumarraj:~/MCS_172/Assignment$ tty
kumarraj@kumarraj:~/MCS 172/Assignment$ tty
kumarraj@kumarraj:~/MCS_172/Assignment$ gcc client.c -o client
                                                             kumarraj@kumarraj:~/MCS_172/Assignment$ gcc server.c -o server
kumarraj@kumarraj:~/MCS_172/Assignment$ ./client
                                                             kumarraj@kumarraj:~/MCS_172/Assignment$ ./server
                                                             Server Started
Connection Established ...
                                                             Waiting For a Client...
                                                             Connection Established with client...
      Pizza-Hut Menu
                                                             Client received the menu...
_____
1.Hawaiin
              ---> Rs.150
2.Greek-Veg
              ---> Rs.200
              ---> Rs.250
Pepperoni
4.Triple Crown ---> Rs.300
5.BBOchicken
-----
Enter the choice to add to the cart:
```

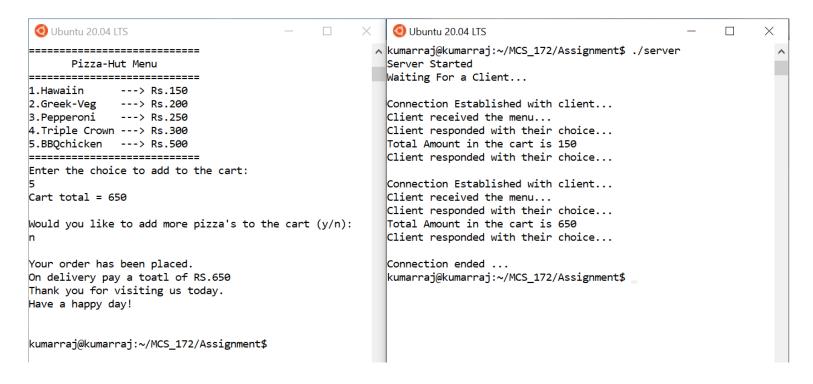
3.Client receives the menu and replies to the server with the pizza choice that he wants to add to the cart. The server calculates the pizza's cost based on the client's preference and responds with the total cart value. And asks if the client wants to add more pizzas to the cart?

```
O Ubuntu 20.04 LTS
                                                               O Ubuntu 20.04 LTS
kumarraj@kumarraj:~/MCS_172/Assignment$ tty
                                                            ^ kumarraj@kumarraj:~/MCS_172/Assignment$ tty
kumarraj@kumarraj:~/MCS_172/Assignment$ gcc client.c -o client
                                                              kumarraj@kumarraj:~/MCS_172/Assignment$ gcc server.c -o server
kumarraj@kumarraj:~/MCS_172/Assignment$ ./client
                                                              kumarraj@kumarraj:~/MCS_172/Assignment$ ./server
                                                              Server Started
Connection Established ...
                                                              Waiting For a Client...
                                                              Connection Established with client...
      Pizza-Hut Menu
                                                              Client received the menu...
                                                              Client responded with their choice...
_____
1.Hawaiin
              ---> Rs.150
                                                              Total Amount in the cart is 150
2.Greek-Veg
              ---> Rs.200
             ---> Rs.250
Pepperoni
4.Triple Crown ---> Rs.300
5.BBQchicken ---> Rs.500
_____
Enter the choice to add to the cart:
Cart total = 150
Would you like to add more pizza's to the cart (y/n):
```

4. Server sends the menu again as the client responded with yes for adding more items to the cart. We loop through step 3 again.



5.After adding desired pizzas to the cart, the client now responds to the server as no. The server accepts the client's response and replies with an endnote stating the client to be ready with the cart total at the time of delivery. With this, the connection comes to an end.



Code for Client.c and Server.c is attached below: -

```
1 /************
2 * MCS 172 - Assignment
3 * Filename : server.c
4 * Author : Rajkumar B L
           : 2047120
5 * Reg.No
7 #include <unistd.h>
8 #include <stdio.h>
9 #include <sys/socket.h>
10 #include <stdlib.h>
11 #include <netinet/in.h>
12 #include <string.h>
13 #include <stdbool.h>
14 #define PORT 8080
15 int cart();
16
17 int main()
18 | {
19
      int server_fd, new_socket, valread;
20
      char info_to_customer[250];
21
      struct sockaddr_in address;
22
      int opt = 1;
23
      int addrlen = sizeof(address);
      char buffer[1024] = \{0\};
24
25
      bool s_conct = true;
26
      char menu[250] = "\n=======\n
27
  Menu\n========================\n1.Hawaiin ---> Rs.150\n2.Greek-Veg
  Rs.200\n3.Pepperoni
                        ---> Rs.250\n4.Triple Crown ---> Rs.300\n5.BBQchicken
  Rs.500\n========";
28
      int cart_amount = 0;
29
      // Creating socket file descriptor
30
      if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
31
32
      {
          perror("socket failed");
33
34
          exit(EXIT_FAILURE);
35
      }
36
37
      // Forcefully attaching socket to the port
      if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
38
  sizeof(opt)))
39
      {
40
          perror("setsockopt");
          exit(EXIT_FAILURE);
41
42
43
      address.sin_family = AF_INET;
44
      address.sin_addr.s_addr = INADDR_ANY;
45
      address.sin_port = htons(PORT);
46
      // Forcefully attaching socket to the port
47
      if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0)</pre>
48
49
          perror("bind failed");
50
51
          exit(EXIT_FAILURE);
52
53
      if (listen(server_fd, 3) < 0)</pre>
54
          perror("listen");
55
56
          exit(EXIT_FAILURE);
```

```
57
        if ((new_socket = accept(server_fd, (struct sockaddr *)&address,(socklen_t
 58
    *)&addrlen)) < 0)
 59
        {
 60
            perror("accept");
            exit(EXIT_FAILURE);
 61
 62
        }
 63
 64
        printf("Server Started\n");
 65
        printf("Waiting For a Client...\n");
 66
        while (s_conct)
 67
 68
 69
            char ch;
 70
            //Sending Menu to the client
 71
            send(new_socket, menu, strlen(menu), 0); //Send 1 Menu
 72
 73
            valread = read(new_socket, buffer, 1024); // Read 1 if the menu is received
 74
            printf("\nConnection Established with client... \n");
            printf("%s\n", buffer);
 75
 76
            memset(buffer, 0, strlen(buffer));
 77
            //Ask the user to enter the choice
 78
 79
            memset(info_to_customer, 0, strlen(info_to_customer));
            strcpy(info to customer, "Enter the choice to add to the cart:");
 80
            send(new_socket, info_to_customer, strlen(info_to_customer), 0); //Send 2 -
 81
    Enter the choice
            memset(info_to_customer, 0, strlen(info_to_customer));
 82
 83
            //Read the option selected by the client
 84
            valread = read(new_socket, buffer, 1024); // Read 2
 85
 86
            printf("Client responded with their choice...\n");
 87
            char ct_amt_st[5];
 88
 89
            cart_amount = cart_amount + cart(buffer);
            printf("Total Amount in the cart is %d\n", cart_amount);
 90
            memset(buffer, 0, sizeof(buffer));
 91
 92
            //Sending current cart total amount to client
            strcpy(info_to_customer, "Cart total = ");
 93
            snprintf(ct_amt_st, 5, "%d", cart_amount);
 94
 95
            strcat(info_to_customer, ct_amt_st);
            send(new_socket, info_to_customer, strlen(info_to_customer), 0); //Send 3 -
 96
    Sending total cart value
            memset(info_to_customer, 0, sizeof(info_to_customer));
 97
 98
            //Asking customer if they want to add more pizzas to the cart
 99
100
            strcpy(info_to_customer, "\nWould you like to add more pizza's to the cart
    (y/n):");
            send(new_socket, info_to_customer, strlen(info_to_customer), 0); //Send 4 -
101
    Asking to add more item
102
            memset(info_to_customer, 0, sizeof(info_to_customer));
103
104
            //Read the option selected by the client for more pizza
105
            valread = read(new socket, buffer, 1024); // Read 3
            printf("Client responded with their choice...\n");
106
            //printf("%s", buffer);
107
108
            ch=buffer[0];
109
            if (buffer[0] == 'y')
110
            {
111
                s_conct = true;
```

```
memset(buffer, 0, strlen(buffer));
112
113
            }
            else
114
115
            {
                memset(info_to_customer, 0, sizeof(info_to_customer));
116
                strcpy(info_to_customer, "\nYour order has been placed.\nOn delivery pay
117
    a toatl of RS.");
                snprintf(ct_amt_st, 5, "%d", cart_amount);
118
                strcat(info_to_customer, ct_amt_st);
119
                send(new_socket, info_to_customer, strlen(info_to_customer), 0); //Send
120
    end 1 - Sending total cart value
                memset(info_to_customer, 0, sizeof(info_to_customer));
121
122
123
                s_conct = false;
                strcpy(info to customer, "\nThank you for visiting us today.\nHave a
124
    happy day!");
                send(new_socket, info_to_customer, strlen(info_to_customer), 0); //Send
125
    end 2 - sending end note
126
                memset(buffer, 0, strlen(buffer));
                printf("\nConnection ended ... \n");
127
128
            }
129
        }
130
131
        return 0;
132 }
133
134 int cart(char choice[])
135 | {
136
        int cost=0;
        switch (choice[0])
137
138
        {
            case '1':
139
                cost=150;
140
141
                break;
142
            case '2':
143
                cost = 200;
144
                break;
145
            case '3':
146
                cost = 250;
147
                break;
            case '4':
148
                cost = 300;
149
150
                break;
151
            case '5':
152
                cost = 500;
153
                break;
154
        }
155
        return cost;
156 }
157
```

```
1 /************
2 * MCS 172 - Assignment
3 * Filename : client.c
4 * Author : Rajkumar B L
5 * Reg.No
            : 2047120
  ******************************
6
8 #include <stdio.h>
9 #include <sys/socket.h>
10 #include <arpa/inet.h>
11 #include <unistd.h>
12 #include <string.h>
13 #include <stdbool.h>
14 #define PORT 8080
15
16 int main()
17 {
       int sock = 0, valread;
18
19
       char info_to_server[100];
20
       struct sockaddr_in serv_addr;
21
       char info_buff[1024] = \{0\};
22
23
       bool c_conct=true;
24
       if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)</pre>
25
26
27
           printf("\n Socket creation error \n");
28
           return -1;
29
       }
30
31
       serv_addr.sin_family = AF_INET;
32
       serv_addr.sin_port = htons(PORT);
33
34
       // Convert IPv4 and IPv6 addresses from text to binary form
35
       if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0)</pre>
36
       {
37
           printf("\nInvalid address/ Address not supported \n");
38
           return -1;
39
       }
40
41
       if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)</pre>
42
       {
43
           printf("\nConnection Failed \n");
           return -1;
44
45
       }
       else
46
47
           printf("\nConnection Established ... \n");
48
49
       }
50
       while (c_conct)
51
52
       {
53
54
           char ch;
55
           //Menu Received
           valread = read(sock, info_buff, 1024); //Read 1
56
57
           printf("%s\n", info_buff);
58
           memset(info_buff, 0, sizeof(info_buff));
59
           memset(info_to_server, 0, sizeof(info_to_server));
60
```

```
strcpy(info_to_server, "Client received the menu...");
 61
 62
            send(sock, info_to_server, strlen(info_to_server), 0); // Send 1
            memset(info to server, 0, sizeof(info to server));
 63
 64
            //Reading form server - Asking to enter the choice
 65
            valread = read(sock, info_buff, 1024); //Read 2
 66
            printf("%s\n", info_buff);
 67
            memset(info_buff, 0, sizeof(info_buff));
 68
 69
            //Sending the Choice to the server
 70
 71
            scanf("%s",info_to_server);
            send(sock, info_to_server, strlen(info_to_server), 0); // Send 2 - sending
 72
    client choice
 73
            memset(info_to_server, 0, sizeof(info_to_server));
 74
 75
            //Reading the total cart value
            valread = read(sock, info_buff, 1024); //Read 3
 76
 77
            printf("%s\n", info_buff);
 78
            memset(info_buff, 0, sizeof(info_buff));
 79
            //Reading - if to add more pizza to cart
 80
            valread = read(sock, info_buff, 1024); //Read 4
 81
            printf("%s\n", info_buff);
 82
 83
            memset(info_buff, 0, sizeof(info_buff));
 84
            //Sending the more pizza Choice to the server
 85
            scanf("%s", info_to_server);
 86
            send(sock, info_to_server, strlen(info_to_server), 0); // Send 3 - sending
 87
    client choice
            //printf("%s", info_to_server);
 88
 89
            ch = info_to_server[0];
 90
            if (info_to_server[0] == 'y')
 91
            {
 92
                c_conct = true;
 93
                memset(info_to_server, 0, sizeof(info_to_server));
 94
            }
 95
            else
96
            {
 97
                c_conct = false;
 98
                //Reading - end note
                valread = read(sock, info_buff, 1024); //Read end 1
 99
                printf("%s\n", info_buff);
100
                memset(info_buff, 0, sizeof(info_buff));
101
102
                valread = read(sock, info_buff, 1024); //Read end 2
103
                printf("%s\n\n", info_buff);
104
                memset(info_buff, 0, sizeof(info_buff));
105
                memset(info_to_server, 0, sizeof(info_to_server));
106
107
            }
108
        }
109
110
        return 0;
111 }
112
```