**Name: Brijesh Rohit**

**Admission no.: U19CS009**

**DISTRIBUTED SYSTEMS - ASSIGNMENT - 5**

**Implement given extensions to the Client Server Programming.**

**1. Extend your echo Client Server message passing application to chat application.**

* **Client and Server are able to send the message to each other until one of them quits or terminates.**

**CODE=>**

**SERVER SIDE**

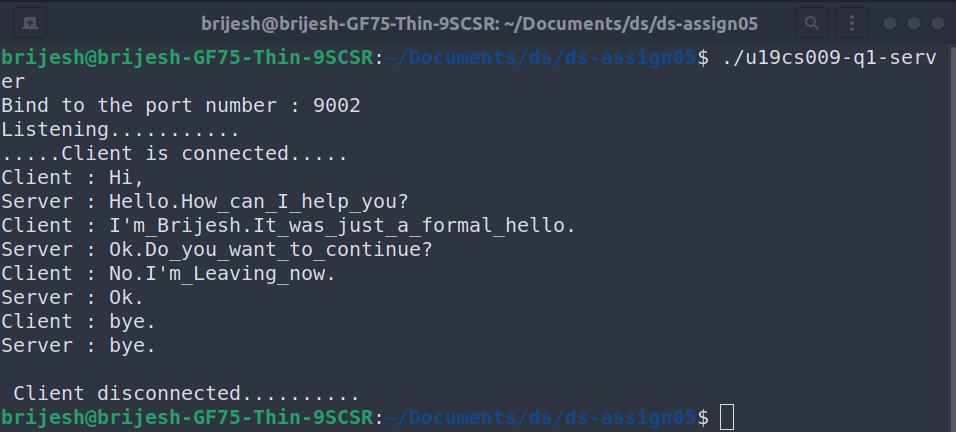
|  |
| --- |
| //U19CS009  //Brijesh Rohit  #include<stdio.h>  #include<stdlib.h>  #include<sys/types.h>  #include<sys/socket.h>  #include<unistd.h>  #include<netinet/in.h>  #include<string.h>  int main()  {  //creating a socket for server  int server\_socket, client\_socket;  server\_socket = socket(AF\_INET, SOCK\_STREAM, 0);  if (server\_socket < 0)  {  perror("--> Server error !!!\n");  exit(1);  }  //define server\_address and client\_address  struct sockaddr\_in server\_address, client\_address;  server\_address.sin\_family = AF\_INET;  server\_address.sin\_port = htons(9002); //passing port number 9002  server\_address.sin\_addr.s\_addr = INADDR\_ANY; //specifing local machine address  //binding socket with specific IP and port number  bind(server\_socket, (struct sockaddr \*)&server\_address, sizeof(server\_address));  printf("Bind to the port number : 9002");  //listening to connection  listen(server\_socket, 4);  printf("\nListening...........\n");  //accept a connection  int size\_client\_addr = sizeof(client\_address);  client\_socket = accept(server\_socket, (struct sockaddr \*)&client\_address, &size\_client\_addr);  if (client\_socket < 0)  {  printf("Error in accepting request!!\n");  exit(1);  }  printf(".....Client is connected.....\n");  //recive data from client  while (1)  {  char response\_client[256];  recv(client\_socket, &response\_client, sizeof(response\_client), 0);  //last parameter is optional so putting 0  printf("Client : ");  printf("%s", response\_client);  printf("\n");  printf("Server : ");  char send\_server[256];  scanf("%s", send\_server);  send(client\_socket, send\_server, sizeof(send\_server), 0);  if (strncmp(send\_server, "bye", 3) == 0)  break;  }  //close the socket  close(client\_socket);  close(server\_socket);  printf("\n Client disconnected..........\n");  return 0;  } |

**CLIENT SIDE**

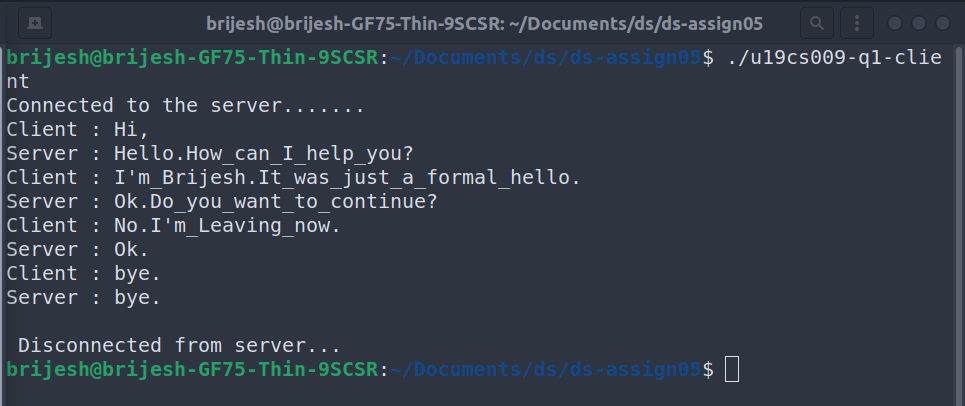
|  |
| --- |
| //U19CS009  //Brijesh Rohit  #include<stdio.h>  #include<stdlib.h>  #include<sys/types.h>  #include<sys/socket.h>  #include<unistd.h>  #include<netinet/in.h>  #include<string.h>  int main()  {  //Creating a socket for client  int no\_socket;  no\_socket = socket(AF\_INET, SOCK\_STREAM, 0);  //specifing address for client socket  struct sockaddr\_in client\_addr;  client\_addr.sin\_family = AF\_INET;  client\_addr.sin\_port = htons(9002); //passing port number 9002  client\_addr.sin\_addr.s\_addr = INADDR\_ANY; //specifing local machine address (equivalent to 0.0.0.0)  int connection\_status = connect(no\_socket, (struct sockaddr \*)&client\_addr, sizeof(client\_addr)); // 0 OK -1 error  //Checking whether there is an error in connection  if (connection\_status < 0)  {  perror("--->There was an error making connection with the remote socket\n\n");  exit(1);  }  printf("Connected to the server.......\n");  //send data to server  while (1)  {  char send\_client[256];  printf("Client : ");  scanf("%s", send\_client);  send(no\_socket, send\_client, sizeof(send\_client), 0);  char recv\_client[256];  recv(no\_socket, &recv\_client, sizeof(recv\_client), 0);  printf("Server : %s", recv\_client);  printf("\n");  if (strncmp(send\_client, "bye", 3) == 0)  break;  }  //close the connection  close(no\_socket);  printf("\n Disconnected from server...\n");  return 0;  } |

**OUTPUT=>**

**SERVER SIDE**



**CLIENT SIDE**



**2. Using the Client-Server communication mechanism get the load status of other nodes in your network (identify the states of other nodes in the system – Overload, Moderate, Lightly).**

* **Implement the Client-Server model. Run the client and server instance on same machine and pass the message from client to server or server to client**
* **Get the CPU load of the client or server and state that either it is under loaded or overloaded.**

**The client server communication mechanism has the limitation that it only handles one connection at a time and then terminates. A real-world server should run indefinitely and should have the capability of handling a number of simultaneous connections, each in its own process.**

**CODE=>**

**SERVER SIDE**

|  |
| --- |
| //U19CS009  //Brijesh Rohit  #include<stdio.h>  #include<stdlib.h>  #include<sys/types.h>  #include<sys/socket.h>  #include<unistd.h>  #include<netinet/in.h>  #include<string.h>  int main() {  //creating a socket for server  int server\_socket, client\_socket;  server\_socket = socket(AF\_INET, SOCK\_STREAM, 0);  if (server\_socket < 0)  {  perror("--> Server error !!!\n");  exit(1);  }  //define server\_address and client\_address  struct sockaddr\_in server\_address, client\_address;  server\_address.sin\_family = AF\_INET;  server\_address.sin\_port = htons(9002); //passing port number 9002  server\_address.sin\_addr.s\_addr = INADDR\_ANY; //specifing local machine address  //binding socket with specific IP and port number  bind(server\_socket, (struct sockaddr \*)&server\_address, sizeof(server\_address));  printf("Bind to the port number : 9002");  //listening to connection  listen(server\_socket, 4);  printf("\nListening...........\n");  //accept a connection  int size\_client\_addr = sizeof(client\_address);  client\_socket = accept(server\_socket, (struct sockaddr \*)&client\_address, &size\_client\_addr);  if (client\_socket < 0)  {  printf("Error in accepting request!!\n");  exit(1);  }  printf(".....Client is connected.....\n");  char response\_client[256];  recv(client\_socket, &response\_client, sizeof(response\_client), 0);  //last parameter is optional so putting 0  printf("Client message : ");  printf("%s", response\_client);  printf("\n");  //close the socket  close(client\_socket);  close(server\_socket);  printf("\n Client disconnected..........\n");  return 0;  } |

**CLIENT SIDE**

|  |
| --- |
| //U19CS009  //Brijesh Rohit  #include<stdio.h>  #include<stdlib.h>  #include<sys/types.h>  #include<sys/socket.h>  #include<unistd.h>  #include<netinet/in.h>  #include<string.h>  int main()  {  //Creating a socket for client  int no\_socket;  no\_socket = socket(AF\_INET, SOCK\_STREAM, 0);  //specifing address for client socket  struct sockaddr\_in client\_addr;  client\_addr.sin\_family = AF\_INET;  client\_addr.sin\_port = htons(9002); //passing port number 9002  client\_addr.sin\_addr.s\_addr = INADDR\_ANY; //specifing local machine address (equivalent to 0.0.0.0)  int connection\_status = connect(no\_socket, (struct sockaddr \*)&client\_addr, sizeof(client\_addr)); // 0 OK -1 error  //Checking whether there is an error in connection  if (connection\_status < 0)  {  perror("--->There was an error making connection with the remote socket\n\n");  exit(1);  }  printf("Connected to the server.......\n");  printf("Client information : ");  FILE \*client;  char send\_client[256];  client = popen("./u19cs009-ds-assign05-q2.sh", "r");  if (client != NULL)  {  while (1)  {  char \*line;  line = fgets(send\_client, sizeof(send\_client), client);  if (line == NULL)  break;  printf("%s", line); /\* line includes '\n' \*/  }  pclose(client);  }  //send data to server  send(no\_socket, send\_client, sizeof(send\_client), 0);  //close the connection  close(no\_socket);  printf("\n Disconnected from server...\n");  return 0;  } |

**OUTPUT=>**

**SERVER SIDE**



**CLIENT SIDE**

