Report No.02

18.08.2021

—

Teams Name : The 4's Squad

Team members:
Akshat Bansal
Prabhanshu Raj Jain
Srishti Kulchandani
Shristi Vishwakarma

Goal

We are trying to create a multithreaded server-client program. Where multiple clients can register at once on the server. Client needs to register and give details like name and password to the server while registering which is stored in a txt file. After that, the client can login again with the same login credentials.

Main function

We created a server.txt file using ofstream (output file stream) data type for storing information of clients.

Command line:

```
ofstream serverFile("server.txt");
```

Creating and binding the socket

Here we create a socket.

Command line:

```
int servSocket = socket(AF_INET, SOCK_STREAM, 0);
  if (servSocket == -1)
  {
```

```
    serverFile << "Cannot create a socket \n";

    cout << "Cannot create a socket \n";
    return -1;
  }

  sockaddr_in servAddr;
  servAddr.sin_family = AF_INET;
  servAddr.sin_port = htons(Port);
  inet_pton(AF_INET, "0.0.0.0", &servAddr.sin_addr);
```

Binding of socket

Command line:

```
if (bind(servSocket, (sockaddr *)&servAddr, sizeof(servAddr)) < 0)
  {
    serverFile << "Error in Binding \n";

    cout << "Error in Binding \n";
    return -1;
  }
```

Listening

Command line:

```
listen(servSocket, SOMAXCONN);

  serverFile << "Started listening to : " << Port << endl;


  cout << "Started listening to : " << Port << endl;
```

pthread_t threads[MAX_CLIENT]; **(we created thread, which accommodates maximum no. of clients defined above.)**

```
  for (int i = 0; i < MAX_CLIENT; i++)
  {
    sockaddr_in clientAddr;
    socklen_t clientSize = sizeof(clientAddr);
    int clientSocket = accept(servSocket, (sockaddr *)&clientAddr, &clientSize);
    if (clientSocket < 0)
    {
```

```
            serverFile << "Problem with client connection\n";

            cout << "Problem with client connection\n";
            return -1;
        }
```

We created threads using MACRO (Replacement component)

Command line:

```
#define MAX_CLIENT 10
```

Connection handler

This function connects from the client.

Command line:

```
void *connection_handler(void *arg)
{

    clientdata *client = (clientdata *)arg;
    int clientSocket = client->socket;
    char data[4096];
    auth_client(client);
    onlineClients.push_back(client->uID);
    string str = "Welcome " + client->name + ",from the server";
    send(clientSocket, str.c_str(), str.size(), 0);

    while (true)
    {
        memset(data, 0, 2048);
        char Hello[2048] = "Hello ";
        int dataRecv = recv(clientSocket, data, 2048, 0);
        if (dataRecv < 0)
        {
```

```
            cout << "Connection issue\n";
            break;
        }
            if (dataRecv == 0)
            {
                cout << "The client dissconnected \n";
                break;
            }

        cout << "Recieved : " << data << endl;
```

(Here clients can get information like “how many clients are online?” or they can ask for their details.)

```
        if(strcmp(data,"get online clients")==0){
            string store=getonlineClients();
            send(clientSocket,store.c_str(),store.size(),0);
        }else if(strcmp(data,"Get my details")==0){
            string store=client->Getdetails();
            send(clientSocket,store.c_str(),store.size(),0);
        }
        else{
        strcat(Hello, data);
        send(clientSocket, Hello, dataRecv + 10, 0);
        }
    }
    close(clientSocket);
    pthread_exit(NULL);
}
```

Client authentication

Command line:

```
void auth_client(clientdata *client)
{
   int clientsocket = client->socket;
   char mssg[2048];
```

This command sends the message to the client and asks whether to register or login.

```
   send(clientsocket,"Register/Login",15,0);
   while (true)
   {   memset(mssg, 0, 2048); ( memset is used to clear the

      recv(clientsocket, mssg, 2048, 0);

      if (strcmp(mssg, "Register") == 0)
      {
         send(clientsocket, "format(Name:Password:uID)", 26, 0);
         client->registerClient();
         break;
      }
      else if(strcmp(mssg,"Login")==0){
         send(clientsocket,"Login Format(uId:Password)",27,0);
         client->loginClient();
         break;
      }
      else
      {
         send(clientsocket, "Retry.....", 12, 0);
      }
   }
```

```
}
```

Registration and login

Command line:

```
class clientdata
{
public:
   int socket, id, portNum;
   string name, uID;
   string password;
   string Getdetails(){
      return (this->uID+":"+to_string(this->portNum));

   }
```

This function lets the client register itself on the server.

```
   void registerClient()
   {

      //Name:Password:uId (Format for getting info)

      char msg[2048] = {0};
      recv(socket, msg, 2048, 0);
      vector<string> data = split(msg, ':');
      ofstream tracker("tracker.txt", ios::app);
      tracker << string(msg) + ":" + to_string(portNum) << endl;
      tracker.close();
      name = data[0];
      password = data[1];
      uID = data[2];
      cout<<name<<endl;
```

```
        }
```

This function lets the client login on the server after registration.

```
        void loginClient(){
          char msgdata[2048]={0};
          recv(socket,msgdata,2048,0);
          vector<string> data = split(msgdata, ':');
```

Input file stream to get information from the client and store it to server.txt file

```
          ifstream tracker("tracker.txt");
          stringstream trac;
          trac<<tracker.rdbuf();
          string temp,fulldata=trac.str();
          while(getline(trac,temp)){
            vector<string> rec = split(temp.c_str(),':');
            if(rec[2].compare(data[0])==0&&rec[1].compare(data[1])==0){
              this->name = rec[0];
              this->password = rec[1];
              this->uID = rec[2];
              tracker.close();
              ofstream tracker("tracker.txt");
              int pos=fulldata.find(rec[2],0)+rec[2].size()+1;
              fulldata.replace(pos,rec[3].size(),to_string(this->portNum));
              tracker<<fulldata;
              cout<<this->name;
              tracker.close();

            }
          }
        }
      };
```

Get online clients

Command line:

```
vector<string> onlineClients;
string getonlineClients(){
   string temp="";
   for(int i=0;i<onlineClients.size();i++){
      temp+=onlineClients[i]+":";
   }
   temp[temp.size()-1]='\0';
   return temp;
}
vector<string> split(const char str[], char a)
{
   vector<string> res;
   stringstream ss(str);
   string temp;
   while (getline(ss, temp, a))
   {
      res.push_back(temp);
   }
   return res;
}
```