DA LAB 4

SAHIL BONDRE: U18CO021

Implement echo client-server message passing application. Messages sent from the client should be displayed on the server and then the program should terminate.

1. Write a server (TCP) C Program that opens a listening socket and waits to serve client. 2. Write a client (TCP) C Program that connects with the server program knowing IP address and port number.
3. Get the input string from console on client and send it to server, server displays the same string.

Java Implementation

Server.java

```java
import java.io.*;
import java.net.*;


public class Server {


  private Socket socket = null;
  private ServerSocket server = null;
  private DataInputStream in = null;


  public Server(int port) {
    try {
      server = new ServerSocket(port);
      System.out.printf("Server started on port: %d\n", port);
```

```
    System.out.println("Waiting for a client ...");

    socket = server.accept();
    System.out.println("Client accepted");

    in =
      new DataInputStream(new BufferedInputStream(socket.getInputStream()));

    String line = "";

    line = in.readUTF();
    System.out.println(line);
    System.out.println("Closing connection");

    socket.close();
    in.close();
  } catch (IOException i) {
    System.out.println(i);
  }
}

public static void main(String args[]) {
  System.out.print("Enter a port number: ");
  int port = Integer.parseInt(System.console().readLine());
  Server server = new Server(port);
```

```
  }
}
```

Client.java

```
import java.io.*;
import java.net.*;


public class Client {


  private Socket socket = null;
  private DataInputStream input = null;
  private DataOutputStream out = null;


  public Client(String address, int port) {
    try {
      socket = new Socket(address, port);
      System.out.printf("Client connected to port: %d\n", port);


      // takes input from terminal
      input = new DataInputStream(System.in);


      // sends output to the socket
      out = new DataOutputStream(socket.getOutputStream());
    } catch (UnknownHostException u) {
      System.out.println(u);
```

```
    } catch (IOException i) {
      System.out.println(i);
    }

    String line = "";

    try {
      line = input.readLine();
      out.writeUTF(line);
    } catch (IOException i) {
      System.out.println(i);
    }

    try {
      input.close();
      out.close();
      socket.close();
    } catch (IOException i) {
      System.out.println(i);
    }
  }

  public static void main(String args[]) {
    System.out.print("Enter a port number: ");
    int port = Integer.parseInt(System.console().readLine());
    Client client = new Client("127.0.0.1", port);
```

```
  }

}
```

```
PS F:\code\github.com\godcrampy\college-notes\da\lab-04> java Server
Enter a port number: 7894
Server started on port: 7894
Waiting for a client ...
Client accepted
Hello World!
Closing connection
PS F:\code\github.com\godcrampy\college-notes\da\lab-04>
```

```
PS F:\code\github.com\godcrampy\college-notes\da\lab-04> java Client
Enter a port number: 7894
Client connected to port: 7894
Hello World!
PS F:\code\github.com\godcrampy\college-notes\da\lab-04>
```

C Implementation

server.c

```c


#include <netdb.h>

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#define MAX 80

#define SA struct sockaddr



void func(int connfd) {
```

```
  char buff[MAX];
  int n;
  bzero(buff, MAX);

  read(connfd, buff, sizeof(buff));
  printf("From client: %s", buff);
}

int main() {
  int sockfd, connfd, len;
  struct sockaddr_in servaddr, cli;
  int PORT = 8080;
  printf("Enter Port: ");
  scanf("%d", &PORT);
  getchar();

  // socket create and verification
  sockfd = socket(AF_INET, SOCK_STREAM, 0);
  if (sockfd == -1) {
    printf("socket creation failed...\n");
    exit(0);
  } else
    printf("Socket successfully created..\n");
  bzero(&servaddr, sizeof(servaddr));

  // assign IP, PORT
```

```
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(PORT);


// Binding newly created socket to given IP and verification
if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
  printf("socket bind failed...\n");
  exit(0);
} else
  printf("Socket successfully binded..\n");


if ((listen(sockfd, 5)) != 0) {
  printf("Listen failed...\n");
  exit(0);
} else
  printf("Server listening..\n");
len = sizeof(cli);


connfd = accept(sockfd, (SA*)&cli, &len);
if (connfd < 0) {
  printf("server accept failed...\n");
  exit(0);
} else
  printf("server accept the client...\n");


func(connfd);
```

```
  close(sockfd);
}
```

client.c

```
#include <netdb.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#define MAX 80
#define SA struct sockaddr

void func(int sockfd) {
  char buff[MAX];
  int n;
  bzero(buff, sizeof(buff));
  printf("Enter the string : ");
  n = 0;
  while ((buff[n++] = getchar()) != '\n')
    ;
  write(sockfd, buff, sizeof(buff));
}
```

```c
int main() {
  int sockfd, connfd;
  struct sockaddr_in servaddr, cli;
  int PORT = 8080;
  printf("Enter Port: ");
  scanf("%d", &PORT);
  getchar();



  // socket create and verification
  sockfd = socket(AF_INET, SOCK_STREAM, 0);
  if (sockfd == -1) {
    printf("socket creation failed...\n");
    exit(0);
  } else
    printf("Socket successfully created..\n");
  bzero(&servaddr, sizeof(servaddr));


  // assign IP, PORT
  servaddr.sin_family = AF_INET;
  servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
  servaddr.sin_port = htons(PORT);


  // connect the client socket to server socket
  if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0) {
    printf("connection with the server failed...\n");
```

```
    exit(0);
  } else
    printf("connected to the server..\n");


  func(sockfd);


  close(sockfd);
}
```

```
→  lab-04 git:(master) ./client
Enter Port: 7889
Socket successfully created..
connected to the server..
Enter the string : Hello
→  lab-04 git:(master)
```

```
→  lab-04 git:(master) ./server
Enter Port: 7889
Socket successfully created..
Socket successfully binded..
Server listening..
server accept the client...
From client: Hello
→  lab-04 git:(master)
```