



Motilal Nehru National Institute of Technology Allahabad
Prayagraj-211004 [India]

Department of Computer Science & Engineering

Programme Name: B.Tech

Semester: VII

Branch: Computer Science & Engg.

Course Code: CS17201

Course Name: Distributed Systems (Lab)

Lab Assignment 5

| Lab # | Name of Experiments | | |
|-------|--|---------------|---------|
| 5 | (i) Implement concurrent echo client-server application. Client----- | | |
| | <pre>#include <stdlib.h> #include <stdio.h> #include <sys/types.h> #include <sys/socket.h> #include <netinet/in.h> #include <string.h> #include <arpa/inet.h></pre> | | |
| | Node no | Ip Address | Port no |
| | 1 | 172.31.100.36 | 2345 |
| | 2 | 172.31.100.40 | 3128 |
| | 3 | 172.31.100.52 | 2323 |
| | <pre>#define MAXLINE 4096 /*max text line length*/ #define SERV_PORT 3000 /*port*/ int main(int argc, char **argv) { int sockfd; struct sockaddr_in servaddr; char sendline[MAXLINE], recvline[MAXLINE]; //basic check of the arguments //additional checks can be inserted if (argc !=2) { perror("Usage: TCPClient <IP address of the server>"); exit(1); } //Create a socket for the client //If sockfd<0 there was an error in the creation of the socket if ((sockfd = socket (AF_INET, SOCK_STREAM, 0)) <0) { perror("Problem in creating the socket"); exit(2); } //Creation of the socket memset(&servaddr, 0, sizeof(servaddr)); servaddr.sin_family = AF_INET; servaddr.sin_addr.s_addr= inet_addr(argv[1]); servaddr.sin_port = htons(SERV_PORT); //convert to big-endian order</pre> | | |
| | | | |
| | | | |
| | | | |
| | | | |

```

//Connection of the client to the socket
if (connect(sockfd, (struct sockaddr *) &servaddr, sizeof(servaddr))<0) {
    perror("Problem in connecting to the server");
    exit(3);
}

while (fgets(sendline, MAXLINE, stdin) != NULL) {

    send(sockfd, sendline, strlen(sendline), 0);

    if (recv(sockfd, recvline, MAXLINE,0) == 0){
        //error: server terminated prematurely
        perror("The server terminated prematurely");
        exit(4);
    }
    printf("%s", "String received from the server: ");
    fputs(recvline, stdout);
}

exit(0);
}

```

server-----

```

#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#include <unistd.h>

#define MAXLINE 4096 /*max text line length*/
#define SERV_PORT 3000 /*port*/
#define LISTENQ 8 /*maximum number of client connections*/

int main (int argc, char **argv)
{
    int listenfd, connfd, n;
    pid_t childpid;
    socklen_t clilen;
    char buf[MAXLINE];
    struct sockaddr_in cliaddr, servaddr;

    //Create a socket for the socket
    //If sockfd<0 there was an error in the creation of the socket
    if ((listenfd = socket (AF_INET, SOCK_STREAM, 0)) <0) {
        perror("Problem in creating the socket");
        exit(2);
    }

    //preparation of the socket address
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(SERV_PORT);

```

```

//bind the socket
bind (listenfd, (struct sockaddr *) &servaddr, sizeof(servaddr));

//listen to the socket by creating a connection queue, then wait for clients
listen (listenfd, LISTENQ);

printf("%s\n","Server running...waiting for connections.");

for ( ; ; ) {

    clilen = sizeof(cliaddr);
    //accept a connection
    connfd = accept (listenfd, (struct sockaddr *) &cliaddr, &clilen);

    printf("%s\n","Received request...");

    if ( (childpid = fork ()) == 0 ) { //if it's 0, it's child process
        printf ("%s\n","Child created for dealing with client requests");

        //close listening socket
        close (listenfd);

        while ( (n = recv(connfd, buf, MAXLINE,0)) > 0) {
            printf("%s","String received from and resent to the client:");
            puts(buf);
            send(connfd, buf, n, 0);
        }

        if (n < 0)
            printf("%s\n", "Read error");
        exit(0);
    }
    //close socket of the server
    close(connfd);
}
}

```

```

C client.c x C server.c
as5 > q1 > C client.c
50 //error: server terminated prematurely
51 perror("The server terminated prematurely");
52 exit(4);
53 }
54 }
55 printf("hell " "String received from the server: ");

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
1: client, server

Avinashs-MacBook-Air:q1 anurag$ ./client 127.0.0.1
hello i am sugandh mishra 20204211
String received from the server: hello i am sugandh
mishra 20204211
how are you
String received from the server: how are you
ugandh mishra 20204211
good
String received from the server: good
re you
ugandh mishra 20204211

Avinashs-MacBook-Air:q1 anurag$ ./server
Server running...waiting for connections.
Received request...
Child created for dealing with client requests
String received from and resent to the client:hel
i am sugandh mishra 20204211
j0
String received from and resent to the client:how
e you
ugandh mishra 20204211
j0
String received from and resent to the client:good
re you
ugandh mishra 20204211
j0

```

- (ii) Implement a client-server program in which the server accepts a connection from a client and updates it own Master table by adding the client information and send the updated table to client, so client can update their own table.

Table format:

Node No Ip Address Port no

1 172.31.100.36 2345

2 172.31.100.40 3128

3 172.31.100.52 2323

client----

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <unistd.h>
#define MAX_CLIENTS 10

struct ClientInfo {
    int node_no;
    char ip_address[20];
    int port_no;
};

int main() {
    int client_socket;
    struct sockaddr_in server_addr;
    struct ClientInfo my_info;

    // Create a socket
    client_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (client_socket < 0) {
        perror("Socket creation error");
        exit(1);
    }

```

```

}

server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(12345); // Use the same port as the server
server_addr.sin_addr.s_addr = inet_addr("127.0.0.1"); // Use the server's IP

// Connect to the server
if (connect(client_socket, (struct sockaddr*)&server_addr,
sizeof(server_addr)) < 0) {
    perror("Connection error");
    exit(1);
}

// Prepare and send client information to the server
my_info.node_no = 4; // Change this to your node number
strncpy(my_info.ip_address, "192.168.0.10", sizeof(my_info.ip_address)); //
Change to your IP
my_info.port_no = 8080; // Change to your port

send(client_socket, &my_info.node_no, sizeof(my_info.node_no), 0);
send(client_socket, my_info.ip_address, sizeof(my_info.ip_address), 0);
send(client_socket, &my_info.port_no, sizeof(my_info.port_no), 0);

// Receive and update the client's master table
struct ClientInfo updated_table[MAX_CLIENTS];
recv(client_socket, updated_table, sizeof(updated_table), 0);

printf("Updated Master Table:\n");
for (int i = 0; i < MAX_CLIENTS; i++) {
    if (updated_table[i].node_no != 0) {
        printf("Node No: %d, IP Address: %s, Port No: %d\n",
updated_table[i].node_no, updated_table[i].ip_address, updated_table[i].port_no);
    }
}

close(client_socket);

return 0;
}

```

server---

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <unistd.h>

#define MAX_CLIENTS 10

struct ClientInfo {
    int node_no;
    char ip_address[20];
    int port_no;
};

```

```

struct ClientInfo master_table[MAX_CLIENTS];
int num_clients = 0;

void updateMasterTable(int node_no, char* ip_address, int port_no) {
    if (num_clients < MAX_CLIENTS) {
        master_table[num_clients].node_no = node_no;
        strncpy(master_table[num_clients].ip_address, ip_address,
sizeof(master_table[num_clients].ip_address));
        master_table[num_clients].port_no = port_no;
        num_clients++;
    } else {
        printf("Master table is full.\n");
    }
}

int main() {
    int server_socket, client_socket;
    struct sockaddr_in server_addr, client_addr;
    socklen_t addr_size;
    int node_no;
    char client_ip[20];
    int client_port;

    // Initialize master table (you can pre-fill this with initial data)

    // Create a socket
    server_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (server_socket < 0) {
        perror("Socket creation error");
        exit(1);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(12345); // Use your desired port
    server_addr.sin_addr.s_addr = INADDR_ANY;

    // Bind the socket
    if (bind(server_socket, (struct sockaddr*)&server_addr, sizeof(server_addr)) <
0) {
        perror("Binding error");
        exit(1);
    }

    // Listen for clients
    if (listen(server_socket, 10) == 0) {
        printf("Listening...\n");
    } else {
        perror("Listening error");
        exit(1);
    }

    // Accept client connections and update the master table
    addr_size = sizeof(client_addr);
    while (1) {

```

```

        client_socket = accept(server_socket, (struct sockaddr*)&client_addr,
&addr_size);

    // Receive client information
    recv(client_socket, &node_no, sizeof(node_no), 0);
    recv(client_socket, client_ip, sizeof(client_ip), 0);
    recv(client_socket, &client_port, sizeof(client_port), 0);

    // Update master table with client information
    updateMasterTable(node_no, client_ip, client_port);

    // Send the updated table back to the client
    send(client_socket, master_table, sizeof(master_table), 0);

    close(client_socket);
}

close(server_socket);

return 0;
}

```

The screenshot shows a code editor with two files: `client.c` and `server.c`. The `client.c` file contains the following code:

```

46 // Receive and update the client's master table
47 struct ClientInfo updated_table[MAX_CLIENTS];
48 recv(client_socket, updated_table, sizeof(updated_table), 0);
49
50 printf("Updated Master Table:\n");
51 for (int i = 0; i < MAX_CLIENTS; i++) {
52     if (updated_table[i].node_no != 0) {
53         printf("Node No: %d, IP Address: %s, Port No: %d\n", updated_table[i].node_no
54     }
55 }
56

```

The terminal window shows the following commands and output:

```

Avinashs-MacBook-Air:q2 anurag$ gcc -o client client.c
Avinashs-MacBook-Air:q2 anurag$ ./client
Updated Master Table:
Node No: 4, IP Address: 192.168.0.10, Port No: 8080
Avinashs-MacBook-Air:q2 anurag$

```

The terminal also shows the output of the `server.c` program:

```

Child created for dealing with client r
Avinashs-MacBook-Air:q1 anurag$ cd ..
Avinashs-MacBook-Air:as5 anurag$ cd q2
Avinashs-MacBook-Air:q2 anurag$ gcc -o server server.c
Avinashs-MacBook-Air:q2 anurag$ ./server
Listening...

```

(iii) Develop a client-server program to implement a date-time server and client. Upon connection establishment, the server should send its current date, time and CPU load information to its clients.

Client----

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

```

```

#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define PORT 8080

int main() {
    int client_socket;
    struct sockaddr_in server_addr;
    char buffer[1024];

    client_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (client_socket < 0) {
        perror("Error in socket");
        exit(1);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = PORT;
    server_addr.sin_addr.s_addr = INADDR_ANY;

    if (connect(client_socket, (struct sockaddr*)&server_addr,
sizeof(server_addr)) < 0) {
        perror("Error in connection");
        exit(1);
    }

    recv(client_socket, buffer, sizeof(buffer), 0);
    printf("Server Response: %s\n", buffer);

    close(client_socket);

    return 0;
}

```

server----

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define PORT 8080

int main() {
    int server_fd, new_socket;
    struct sockaddr_in server_addr, new_addr;
    socklen_t addr_size;
    char buffer[1024];
    time_t rawtime;

```



```

struct tm *info;

server_fd = socket(AF_INET, SOCK_STREAM, 0);
if (server_fd < 0) {
    perror("Error in socket");
    exit(1);
}

server_addr.sin_family = AF_INET;
server_addr.sin_port = PORT;
server_addr.sin_addr.s_addr = INADDR_ANY;

if (bind(server_fd, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0)
{
    perror("Error in bind");
    exit(1);
}

if (listen(server_fd, 10) == 0) {
    printf("Listening...\n");
} else {
    perror("Error in listen");
    exit(1);
}

addr_size = sizeof(new_addr);
new_socket = accept(server_fd, (struct sockaddr*)&new_addr, &addr_size);
time(&rawtime);
info = localtime(&rawtime);
snprintf(buffer, sizeof(buffer), "Date and Time: %sCPU Load: 0.75\n",
asctime(info));
send(new_socket, buffer, strlen(buffer), 0);

close(new_socket);
close(server_fd);

return 0;
}

```

The screenshot displays a macOS desktop environment with the Visual Studio Code (VS Code) editor open. The editor has two tabs: `client.c` and `server.c`. The `client.c` tab is active, showing a C program with a line number 37. The terminal window at the bottom is titled `1: bash, bash` and shows the following commands and output:

```
Avinashs-MacBook-Air:q3 anurag$ gcc -o client client.c
Avinashs-MacBook-Air:q3 anurag$ ./client
Server Response: Date and Time: Thu Oct 19 19:14:35 2023
CPU Load: 0.75
Avinashs-MacBook-Air:q3 anurag$
```

The right side of the terminal window shows the output of the server program:

```
Avinashs-MacBook-Air:q3 anurag$ gcc -o server server.c
Avinashs-MacBook-Air:q3 anurag$ ./server
Listening...
Avinashs-MacBook-Air:q3 anurag$
```

The status bar at the bottom of the VS Code window indicates the current position is `Ln 37, Col 2`, with `Spaces: 4`, `UTF-8`, `LF`, and `C` encoding. The macOS dock at the bottom contains various application icons, including Finder, Safari, Mail, and VS Code.