EX-6

Implementing remote command execution using sockets system call

Implementing remote command execution using sockets in C involves creating a server that listens for client connections and a client that sends commands to be executed on the server. The server then executes these commands and sends the output back to the client.

 ****Server****:

* The server listens on a specified port.
* When a command is received from the client, it uses popen() to execute the command and captures the output.
* The output is then sent back to the client.

 ****Client****:

* The client connects to the server using the server's IP address and port.
* It sends commands to the server and displays the output received.

Server.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 12345

#define BUFFER\_SIZE 1024

void execute\_command(char \*command, char \*result, int result\_size) {

FILE \*fp;

fp = popen(command, "r");

if (fp == NULL) {

snprintf(result, result\_size, "Failed to run command\n");

return;

}

fread(result, sizeof(char), result\_size - 1, fp);

pclose(fp);

}

int main() {

int server\_fd, client\_fd;

struct sockaddr\_in address;

int addrlen = sizeof(address);

char buffer[BUFFER\_SIZE] = {0};

char result[BUFFER\_SIZE] = {0};

// Creating socket file descriptor

if ((server\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) == 0) {

perror("socket failed");

exit(EXIT\_FAILURE);

}

// Bind the socket to the network address and port

address.sin\_family = AF\_INET;

address.sin\_addr.s\_addr = INADDR\_ANY;

address.sin\_port = htons(PORT);

if (bind(server\_fd, (struct sockaddr \*)&address, sizeof(address)) < 0) {

perror("bind failed");

close(server\_fd);

exit(EXIT\_FAILURE);

}

// Listen for incoming connections

if (listen(server\_fd, 3) < 0) {

perror("listen failed");

close(server\_fd);

exit(EXIT\_FAILURE);

}

printf("Listening on port %d...\n", PORT);

while (1) {

// Accept a connection from a client

if ((client\_fd = accept(server\_fd, (struct sockaddr \*)&address, (socklen\_t\*)&addrlen)) < 0) {

perror("accept failed");

close(server\_fd);

exit(EXIT\_FAILURE);

}

while (1) {

memset(buffer, 0, BUFFER\_SIZE);

memset(result, 0, BUFFER\_SIZE);

// Read the command from the client

int read\_size = read(client\_fd, buffer, BUFFER\_SIZE);

if (read\_size == 0) {

printf("Client disconnected\n");

break;

} else if (read\_size < 0) {

perror("read failed");

break;

}

printf("Received command: %s\n", buffer);

// Execute the command and capture the output

execute\_command(buffer, result, BUFFER\_SIZE);

// Send the output back to the client

send(client\_fd, result, strlen(result), 0);

}

// Close the client connection

close(client\_fd);

}

// Close the server socket

close(server\_fd);

return 0;

}

Client.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT 12345

#define BUFFER\_SIZE 1024

int main() {

int sock = 0;

struct sockaddr\_in serv\_addr;

char buffer[BUFFER\_SIZE] = {0};

// Create socket

if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {

printf("\n Socket creation error \n");

return -1;

}

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(PORT);

// Convert IPv4 and IPv6 addresses from text to binary form

if(inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr) <= 0) {

printf("\nInvalid address/ Address not supported \n");

return -1;

}

// Connect to the server

if (connect(sock, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) < 0) {

printf("\nConnection Failed \n");

return -1;

}

while (1) {

printf("Enter command to execute on server: ");

fgets(buffer, BUFFER\_SIZE, stdin);

buffer[strcspn(buffer, "\n")] = 0; // Remove the newline character

if (strcmp(buffer, "exit") == 0) {

break;

}

// Send the command to the server

send(sock, buffer, strlen(buffer), 0);

// Receive the result from the server

memset(buffer, 0, BUFFER\_SIZE);

read(sock, buffer, BUFFER\_SIZE);

printf("%s\n", buffer);

}

// Close the socket

close(sock);

return 0;

}

OUTPUT

SERVER

[11ca013@mcalinux network]$ cc server.c

[11ca013@mcalinux network]$ ./server

Command received : ls

1.c               4tcpcli.c     7pingclie.c  exno2a.c               hello.txt      sevan.txt

2.c               4tcpser.c               7pingser.c             exno2client.c        mca1.txt      tcpchclient.c

3tcpcli.c      4udpcli.c              8rcpcli.c      exno2server.c       menu.sh      tcpchserver.c

3tcpser.c     4udpser.c             8rcpser.c    exnola.sh              payroll.sh    third.txt

3udpcli.c      6tcpftpcli.c  9rcpcli.c              febser.sh                         rajiv

Command received : cat sample.txt

rajivgandhi

hi

how are you

Command received : date

Fri Mar  1 11:39:14 IST 2013

Command received : cal

     March 2013

Su     Mo    Tu     We    Th     Fr      Sa

                                                          1        2

          3        4        5        6        7        8        9

10      11      12      13      14      15      16

17      18      19      20      21      22      23

24      25      26      27      28      29      30

31

Command received : end

CLIENT

Cc client.c

]$ ./client

Enter the port number  :  1806

Enter the command:Enter the command:ls

Enter the command:cat sample.txt

Enter the command:date

Enter the command:cal

Enter the command:end

3 Response to "REMOTE COMMAND EXECUTION USING SOCKETS"