CHAUDHARY HAMDAN

1905387

Networks Lab 2

15/07/2021

1. Server.c and Client.c files were explained in class.

Code (server.c):

/\*

NETWORK PROGRAMMING WITH SOCKETS

We show the communication between a server

process and a client process.

Since many server processes may be running in a system, we identify the

desired server process by a "port number". Standard server processes have

a worldwide unique port number associated with it. For example, the port

number of SMTP (the sendmail process) is 25. To see a list of server

processes and their port numbers see the file /etc/services

In this program, we choose port number 6000 for our server process. Here we

shall demonstrate TCP connections only. For details and for other types of

connections see:

Unix Network Programming

-- W. Richard Stevens, Prentice Hall India.

To create a TCP server process, we first need to open a "socket" using the

socket() system call. This is similar to opening a file, and returns a socket

descriptor. The socket is then bound to the desired port number. After this

the process waits to "accept" client connections.

\*/

#include<stdio.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<fcntl.h>

#include<string.h>

main()

{

int sockfd, fd1, length, i;

char buf[100]; /\* We will use this buffer for communication \*/

struct sockaddr\_in sa\_addr, cl\_addr;

/\* The following system call opens a socket. The first parameter

indicates the family of the protocol to be followed. For internet

protocols we use AF\_INET. For TCP sockets the second parameter

is SOCK\_STREAM. The third parameter is set to 0 for user

applications.\*/

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

/\* The structure "sockaddr\_in" is defined in <netinet/in.h> for the

internet family of protocols. This has three main fields. The

field "sin\_family" specifies the family and is therefore AF\_INET

for the internet family. The field "sin\_addr" specifies the

internet address of the server. This field is set to INADDR\_ANY

for machines having a single IP address. The field "sin\_port"

specifies the port number of the server.\*/

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

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

sa\_addr.sin\_port = htons(6000);

memset(sa\_addr.sin\_zero, '\0', sizeof sa\_addr.sin\_zero);

/\* With the information provided in serv\_addr, we associate the server

with its port using the bind() system call. \*/

i = bind(sockfd, (struct sockaddr \*)&sa\_addr, sizeof(sa\_addr));

printf("test %d%d\n", sockfd, i);

/\* This specifies that up to 5 concurrent client

requests will be queued up while the system is

executing the "accept" system call below.\*/

listen(sockfd, 5);

/\* The accept() system call accepts a client connection.

It blocks the server until a client request comes.

The accept() system call fills up the client's details

in a struct sockaddr which is passed as a parameter.

The length of the structure is noted in clilen. Note

that the new socket descriptor returned by the accept()

system call is stored in "fd1".\*/

length = sizeof(cl\_addr);

fd1 = accept(sockfd, (struct sockaddr \*) &cl\_addr, &length);

/\* We initialize the buffer, copy the message to it,

and send the message to the client. \*/

for (i = 0; i < 100; i++)

buf[i] = '\0';

strcpy(buf, "Message from server");

send(fd1, buf, 100, 0);

/\* We again initialize the buffer, and receive a

message from the client. \*/

for (i = 0; i < 100; i++)

buf[i] = '\0';

recv(fd1, buf, 100, 0);

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

close(fd1);

}

Code (client.c):

/\* THE CLIENT PROCESS

Please read the file server.c before you read this file. To run this,

you must first change the IP address specified in the line:

serv\_addr.sin\_addr.s\_addr = inet\_addr("144.16.202.221");

to the IP-address of the machine where you are running the server.

\*/

#include<stdio.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<arpa/inet.h>

#include<fcntl.h>

#include<string.h>

main()

{

int i, sockfd;

char buf[100];

struct sockaddr\_in sa\_addr;

/\* Opening a socket is exactly similar to the server process \*/

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

/\* Recall that we specified INADDR\_ANY when we specified the server

address in the server. Since the client can run on a different

machine, we must specify the IP address of the server.

TO RUN THIS CLIENT, YOU MUST CHANGE THE IP ADDRESS SPECIFIED

BELOW TO THE IP ADDRESS OF THE MACHINE WHERE YOU ARE RUNNING

THE SERVER.\*/

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

sa\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1"); //Loop back IP address

sa\_addr.sin\_port = htons(6000);

memset(sa\_addr.sin\_zero, '\0', sizeof sa\_addr.sin\_zero);

/\* With the information specified in serv\_addr, the connect()

system call establishes a connection with the server process.\*/

i = connect(sockfd, (struct sockaddr \*)&sa\_addr, sizeof(sa\_addr));

/\* After connection, the client can send or receive messages.

However, please note that recv() will block when the

server is not sending and vice versa. Similarly send() will

block when the server is not receiving and vice versa. For

non-blocking modes, refer to the online man pages.\*/

for (i = 0; i < 100; i++)

buf[i] = '\0';

recv(sockfd, buf, 100, 0);

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

for (i = 0; i < 100; i++)

buf[i] = '\0';

strcpy(buf, "Message from client");

send(sockfd, buf, 100, 0);

close(sockfd);

}

Output:

