Labs for Network Programming-3

BUPT/QMUL

2017-4-6

Lab on UDP

- Write the programs for data sending based on UDP 【Will be checked next week】
  - You can design the running command format, e.g.,
    - ./<exefile> <server> <data> ……
    - The server may be specified in domain name or IP address
    - ***Multiple words*** can be delivered each time
  - Each time the server receives data, ***the source (client's IP address and port number) and the data should be displayed on the screen***

Lab on UDP

Follow the step …

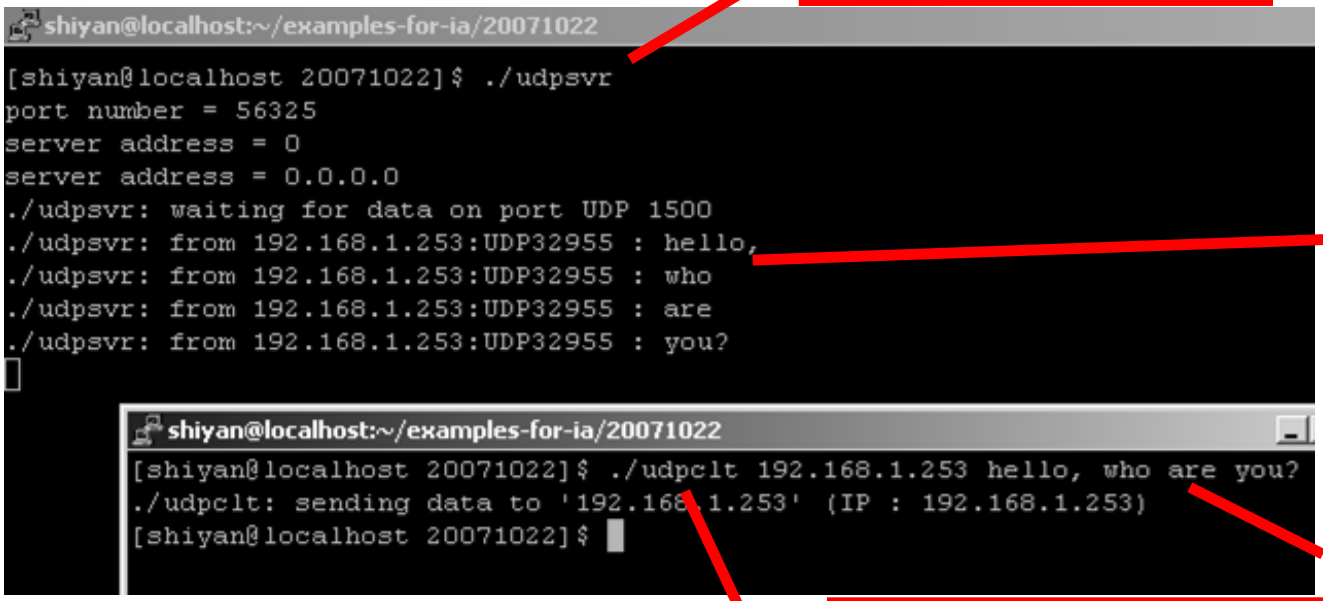
1. Type the code in the IA course slides into two *.c* files (server and client)
2. Compile and run them (Firstly, run the server program in Terminal 1 and then run the client program in Terminal 2)
3. Modify the code of both server and client ***so that the client can send multiple words in one time***. (Example is shown in the following)

Your mission

An Example for your work

- Only an example, you can design the format by yourself

1. Run the server program in Terminal 1



3. Server receives the content and print it out.

Input "IP" + "Content"

2. Run the client program in Terminal 2

TIPs

- DO NOT forget the ***#include <xxx.h>***
- Comment in ***/* xxx */*** can be ignored
- It is SEMICOLON “ ; ” but NOT COLON “ : ” in the end of each line
- Open two Xshell terminals or use ***Alt*** + F1/F2 to switch terminals in Virtualbox.
- Run the ***Server*** firstly. Then run the ***Client*** and send data
- You can use “ifconfig” to check your IP address or use “***127.0.0.1***” as your IP address when you test the program
- ***Try to read the compiling information when debug. Do not always beg help without any reading, thinking and analyzing.***

```
#include <stdio.h> /* for printf() and fprintf() */
#include <sys/socket.h> /* for socket(), bind(), sendto() and recvfrom() */
#include <arpa/inet.h> /* for sockaddr_in and inet_ntoa() */
#include <stdlib.h> /* for atoi() and exit() */
#include <string.h> /* for memset() */
#include <unistd.h> /* for close() */
#define ECHOMAX 255 /* Longest string to echo */

int main(int argc, char *argv[])
{
    int sock; /* Socket */
    struct sockaddr_in echoServAddr; /* Local address */
    struct sockaddr_in echoClntAddr; /* Client address */
    unsigned int cliAddrLen; /* Length of client address */
    char echoBuffer[ECHOMAX]; /* Buffer for echo string */
    unsigned short echoServPort; /* Server port */
    int recvMsgSize; /* Size of received message */
 if (argc != 2)
 {
     printf("Usage: %s <UDP SERVER PORT>\n", argv[0]);
     exit(1);
  }
echoServPort = atoi(argv[1]); /* First arg: local port */
```

Include necessary headers.

Define Parameters.

Count the Input.

Get the Port.

```
/* Create socket for sending/receiving datagrams */
if ((sock = socket(PF_INET, SOCK_DGRAM, 0)) < 0)
    printf("socket() failed.\n");
/* Construct local address structure */
memset(&echoServAddr, 0, sizeof(echoServAddr));
echoServAddr.sin_family = AF_INET;
echoServAddr.sin_addr.s_addr = htonl(INADDR_ANY);
echoServAddr.sin_port =htons(echoServPort);
/* Bind to the local address */
if ((bind(sock, (struct sockaddr *) &echoServAddr, sizeof(echoServAddr))) < 0)
    printf("bind() failed.\n");
for (;;) /* Run forever */
{
    /* Set the size of the in-out parameter */
    cliAddrLen = sizeof(echoClntAddr);
    /* Block until receive message from a client */
    if ((recvMsgSize = recvfrom(sock, echoBuffer, ECHOMAX, 0,(struct sockaddr
*) &echoClntAddr, &cliAddrLen)) < 0)
        printf("recvfrom() failed.\n");
    printf("Handling client %s\n",inet_ntoa(echoClntAddr.sin_addr));
    /* Send received datagram back to the client */
    if ((sendto(sock, echoBuffer, recvMsgSize, 0, (struct sockaddr *)
&echoClntAddr, sizeof(echoClntAddr))) != recvMsgSize)
        printf("sendto() sent a different number of bytes than expected.\n");
 }
}
```

Create the socket. IMPORTANT !

Construct a *structure* and write each items of it.

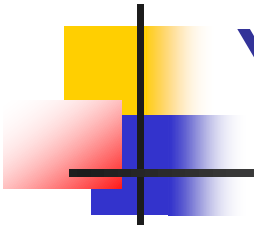
BIND the socket. Pay attention to the type.

Loop forever

Define the length of echClntAddr.

Receive the messages from the client, and save the messages received in the echoBuffer.

Send the messages in echoBuffer back to client.



Your work

- You can modify the code based on the example in the lecture.
- All your work is to ***modify the server and client to support multiple words transmission***.
- Good Luck!