Network Programming with TCP

BUPT/QMUL

2019-04-08

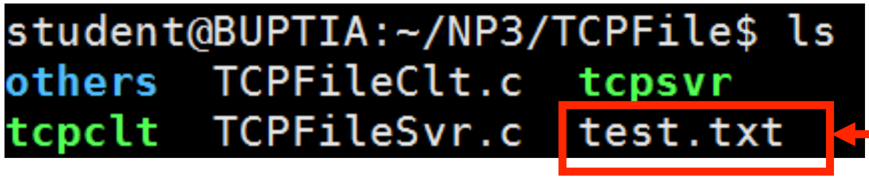
北京郵電大學
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

Electronic Engineering

Lab on TCP

- Write two programs (server and client) so that the server can transfer a local file (indicate by client) to client using TCP.
  - You can design the running command format, e.g.,
    - ./<exefile> <server> <filename>
    - The server may be specified in domain name or IP address
    - The transferred file should be indicated by client's input
    - Client should rename and save the transferred content as a new file
  - Each time the server sends data, following information has to be printed:
    - *the destination (client's IP address and port number)*
    - *Amount of data that has been sent by "Byte"*

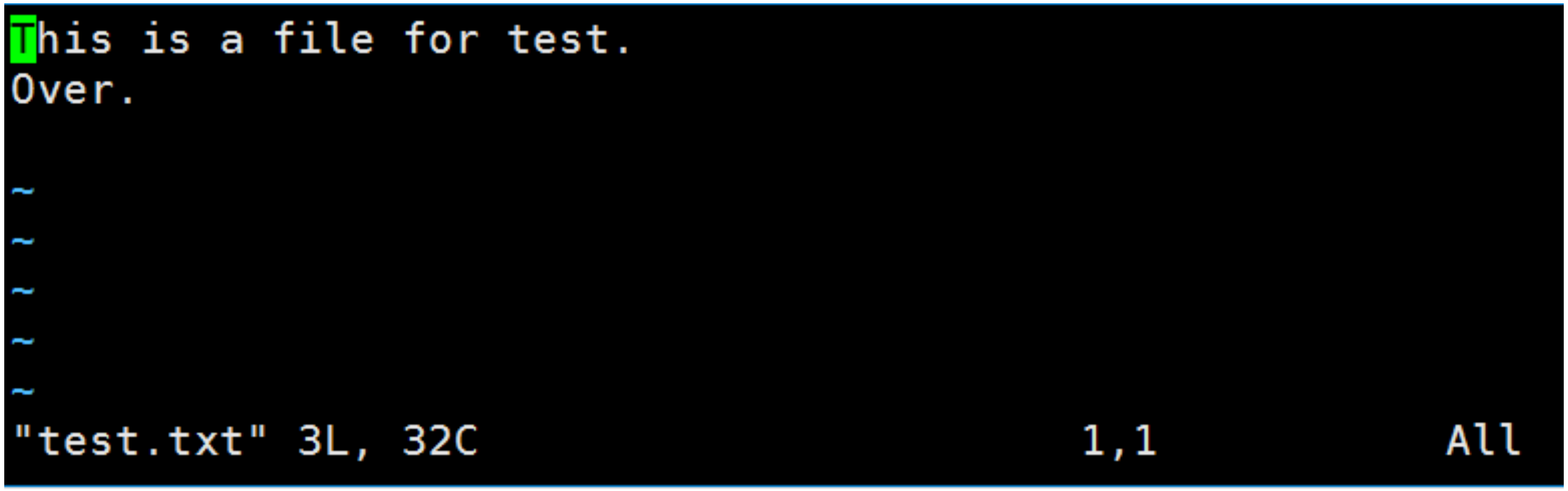
Sample Programs (1)



```
student@BUPTIA:~/NP3/TCPFile$ ls
others   TCPFileClt.c  tcpsvr
tcpclt   TCPFileSvr.c  test.txt
```

The file to be transferred

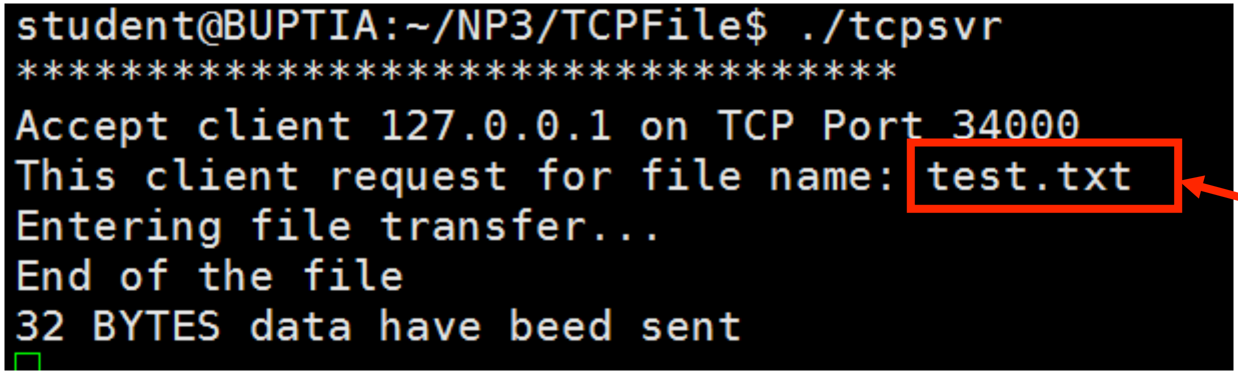
The content of file "test.txt"



```
This is a file for test.
Over.

~
~
~
~
~
"test.txt" 3L, 32C                        1,1           All
```

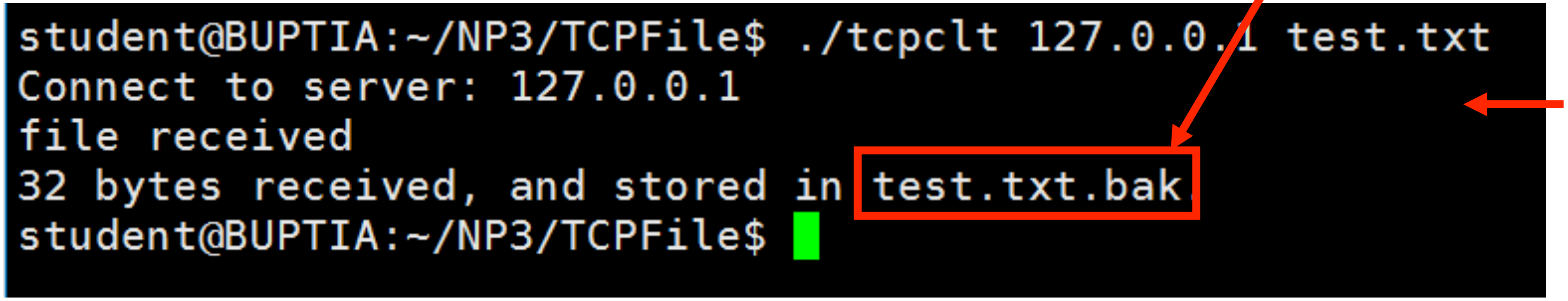
Sample Programs (2)



```
student@BUPTIA:~/NP3/TCPFile$ ./tcpsvr
**********************************
Accept client 127.0.0.1 on TCP Port 34000
This client request for file name: test.txt
Entering file transfer...
End of the file
32 BYTES data have beed sent
```

Step 1.
Run the server

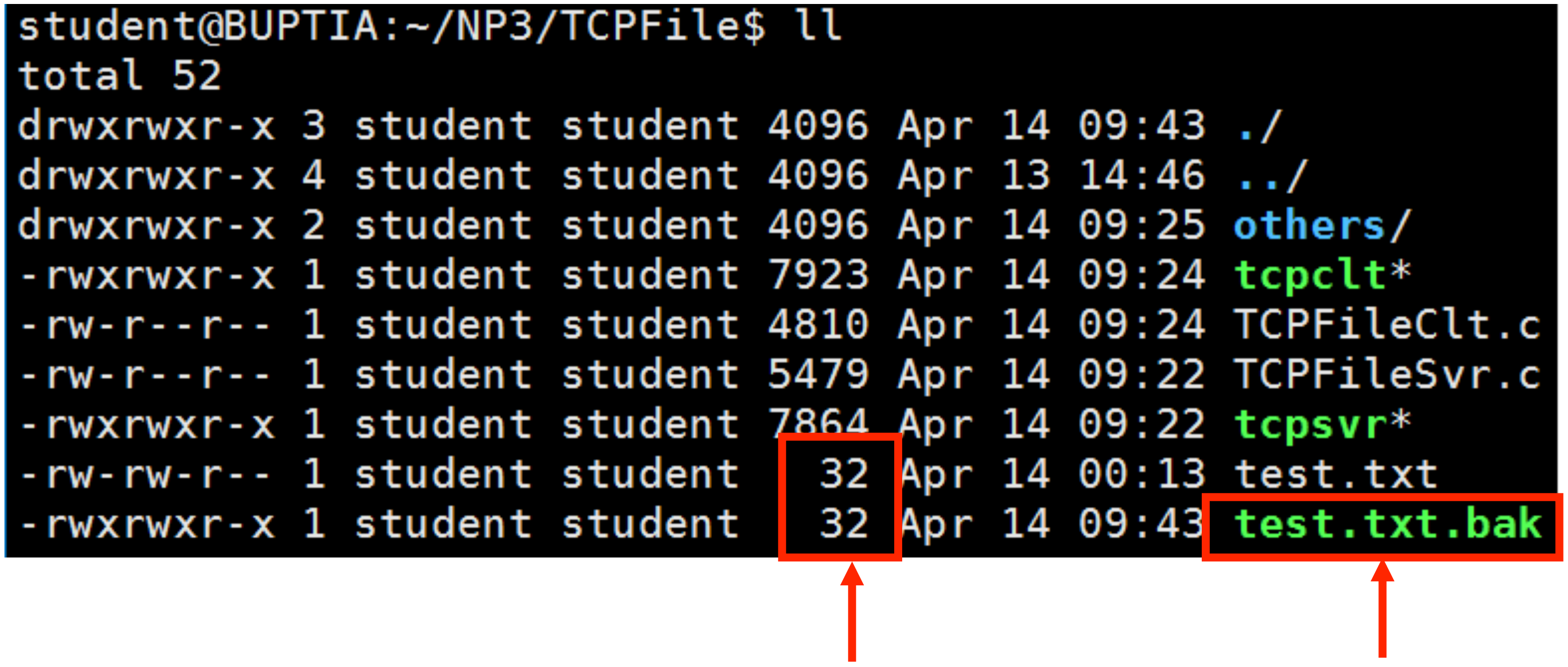
Server sends file "*test.txt*" to client, and client save it as "*test.txt.bak*"



```
student@BUPTIA:~/NP3/TCPFile$ ./tcpclt 127.0.0.1 test.txt
Connect to server: 127.0.0.1
file received
32 bytes received, and stored in test.txt.bak
student@BUPTIA:~/NP3/TCPFile$
```

Step 2.
Run the client

Sample Programs (3)



```
student@BUPTIA:~/NP3/TCPFile$ ll
total 52
drwxrwxr-x 3 student student 4096 Apr 14 09:43 ./
drwxrwxr-x 4 student student 4096 Apr 13 14:46 ../
drwxrwxr-x 2 student student 4096 Apr 14 09:25 others/
-rwxrwxr-x 1 student student 7923 Apr 14 09:24 tcpclt*
-rw-r--r-- 1 student student 4810 Apr 14 09:24 TCPFileClt.c
-rw-r--r-- 1 student student 5479 Apr 14 09:22 TCPFileSvr.c
-rwxrwxr-x 1 student student 7864 Apr 14 09:22 tcpsvr*
-rw-rw-r-- 1 student student   32 Apr 14 00:13 test.txt
-rwxrwxr-x 1 student student   32 Apr 14 09:43 test.txt.bak
```

Same size with the original file and can be opened correctly

The new file

Your work

- Refer to the **lseek()** and **O_APPEND** program. It reads /writes the content from a given file.
- Refer to the **UDP** program.
- Pay attention to following aspects
  - Difference between UDP and TCP.
  - How to judge the existence of required file and deal with it via a simplest way.
- You will be asked to transfer **at least two different files individually**, which helps to check the validity of your program.

Framework of server (only for reference)

```
#include <xxx.h>
#include <yyy.h>
……
int main (int argc, char *argv[]){
    ……
    sockfd =  socket (xx, xx, xx);   // create the socket
    bind(xx, xx, xx);     // bind
    listen(sockfd, xx); //listen
    for( ; ; ){        // Loop forever
        newsockfd = accept(xx, xx, xx);//create a new socket
        if ( recv (xx, xx, xx) ) {     // receive the file name
            fd = open(xx, xx, xx)     // Judge and open the file
            for(xx, xx, xx){    //loop to read the file
                buffer = read(fd, xx, xx);      //  read content
                send (newsockfd, buffer, xx);      //  send content
            }
        }
        close(fd);
    }
}
```

Useful headers

Some Parameters

Socket, bind and fault-tolerance

Listen

ACCEPT

Try to open the file

Loop until the file's end

Read and Transfer content

Some ***close()***

Framework of client (only for reference)

```
#include <xxx.h>
#include <yyy.h>
……
int main (int argc, char *argv[]){
      ……
      sockfd =  socket (xx, xx, xx);   // Setup the socket
      connect(xx, xx, xx)    // Connect
      send (xx, filename, xx); //  send the filename indicated by argv to server

      fd = open(xx, xx, xx)    // Open a new file and prepare to write
      for (xx; xx; xx)  {
                    if ( recv (xx, buffer, xx) ) {  //  Judge whether server close the sock
                           buffer = write(fd, xx, xx);     //  write content
                    }
     }
      close(fd);
}
```

Useful headers

Some Parameters

Socket, connect the server

Send the file name to server

open() a new file

Receive the data

Receive and write content

close() and save the file

Hints

- The framework is just a reference.
- Server will close the socket after finishing the transmission.
- Client will get a “0” from *recv* if the socket in server side has been closed.