OS 2014 – EX6

Reliable File Transfer

Supervisor – Netanel Zakay

Due Date: 19/6/2014

Assignment Description

In this exercise you will implement reliable file transfer between a client and a server. The protocol should use TCP to transmit the file from a client to the server. You will create two programs, one that represents the server (srftp.c or srftp.cpp), and the other represents a client (clftp.c or clftp.cpp).

The Server

For running the server:

srftp server-port

Notes:

- The server receives a port number, and listens to this port. It waits for clients to connect to this port (using TCP connection) and start sending a file to the server.
- The client must supply both the name of the file and the file's content. The server saves the file locally (in the working directory). You should create a simple transmission protocol. Using this protocol, the server will be able to separate between the name of the file and the file's content.
- If the file already exists, the server will overwrite it.
- The server handles multiple clients by using the “select()” system call. It must use a single process and a single thread.
- The server doesn't finish naturally (it can be killed, e.g. by "Crl + c" in the shell).

The Client

For running the client:

crftp server-port server-hostname file-to-transfer filename-in-server

Parameters' description:

server-port	The port used by the server.
server-hostname	The host name of the server.
file-to-transfer	The path of the file to transfer from the client to the server. This can be a relative or absolute path of a file (located at the client's computer). The client will transmit the content of this file to the server. The server will write this file's content in its working directory.
filename-in-server	The name of the file that the server saves. The client should send to the server file's name in addition to the file's content.

Notes:

- The flow of the client is simple. First, it opens the *file-to-transmit.* Then, it transmits its content and the *filename-in-server* to the server.
- As mentioned above, you should create a simple transmission protocol that enables the server to separate the file's name and the file's content.
- In case of success the client program should return 0.

Error Handling

We know that the exam period is close by, and therefore we tried to keep this simple.

- You can assume that the input parameters are valid and correct. For example:
  - ❖ The port is a number
  - ❖ The path of the file to transmit is valid, the file exists, and it is not a directory.
  - ❖ The new file name (that will be saved on the server) is a valid name.
- In case of an unrecoverable error (e.g., if the program can't open a socket), the program will do the following:

    - ❖ First, print informative error message via STDERR. The message must start with a prefix of “ERROR: “.
    - ❖ Exit using “exit(1)”.

  Pay attention – you are not required to deallocate all the resources in case on an error!

Testing Your Program:

- ✓ To verify that the file was transferred properly and has the same data exactly, use “diff” command.
- ✓ Basic text: Run a server and a single client on one machine.
- ✓ Try to transfer both ASCII files (test files) and binary files (program files / images etc.).
- ✓ Also, try different files size (in particular, files that require multiple send / recv and those that require a single one).
- ✓ Try to transfer files to a single server from multiple clients simultaneously.

Useful system calls

socket, bind, connect, listen, accept, send, recv, select, close, inet_addr, htons, ntohs, gethostbyname

The client code is online

I understand that the examination period is close by. Therefore I decided to publish my own clftp.cpp. A few notes:

- Notice that my goal in writing the code is to understand the difficulties of this EX rather than to supply an “example code” for students.
- It’s your choice whether use this code without a change, changing it, or not use it at all.
- If you decide to it, you should figure out the protocol from the code by yourself. However, my code is documented and it shouldn’t be a hard task.

<u>Theoretical part (20 points)</u>

This section equal to 20 points, and should be answered in details in the README file.

- **Reliable file transfer protocol over UDP connection (10 points).**
  In this EX, you are required to implement a file transfer protocol over TCP connection. The question is how it could be done over **UDP connection**. You shall develop an **efficient** and **reliable** file transfer protocol over UDP connection. You should take into account the problems arise by using UDP, and solve them in your suggested protocol.
  You are required to describe such protocol, explain it (e.g. how it solves each problem, why it is efficient, etc.), and mention how its implementation will be different from the protocol you are required to implement.

- **Performance evaluation of your program (10 points).**
  In order to evaluate the performance of your file transfer protocol, you shall measure the time spent from the establishment of the connection and until its termination. A possible way to measure this elapsed time is by using gettimeofday(), that you are already familiar with. You are requested to create a graph that summarizes the performance of your program.
    - The X-axis of the graph is the size of the file, and the Y-axis is the time spent to transfer this file. You are requested to evaluate the performance of at least 5 different sizes.
    - You shall repeat the performance evaluation twice, under different conditions: when the server and the client are on the same computer, and when they are on different computers. Each such experiment will have a different line in the graph.
    - The graph's name is "performance.jpg".
    - You can create the graph with any program you wish (Matlab, Excel, Word, Python...). However, pay attention to have a very clear and informative graph, including a title, x-label and y-label.
    - Any detail that is not supplied here (e.g. the sizes, the files' types, etc.) is your choice.

  Please summarize the experiment and analyze the results in your README file

Submission

1. The files srftp.c \ srftp.cpp and clftp.c \ clftp.cpp.
2. A Makefile which compiles the executables. That means that a simple “make” command creates two executables: clftp and srftp. More requirements of the Makefile appear in the course guidelines.
3. performance.jpg.
4. A README

Good luck!!