2023-07-26

Fshare1.0: Simple FTP-like File Sharing System

Overview

This mission asks you to construct a pair of server and client programs to share a directory over network. Using the client program, a user can look up which files are stored in the shared directory at the server side, and download a file from the shared directory, and upload a file from a local directory to the shared directory. The server program receives requests from one or multiple client programs simultaneously and serves each request by sending the list of contained files (for a listing request), sending the file data (for a downloading request), and writing files with the transferred data (for an uploading request).

This program must use TCP for connecting the server and the clients. The server program and the client program must use multithreading to transfer multiple files concurrently.

Background Study

- Study the basics of the TCP/IP networking with a simple server-client example
  - https://youtu.be/ruxT8hy0M9I
  - https://github.com/hongshin/OperatingSystem/tree/sysprog/IPC
- Q1. Change server.c and client.c to use multithreading instead of multiprocessing.
- Q2. Change server.c such that it regards the message received from a client as text file name and answer back to the client by sending the content of the indicated text file.

Program design

1. Command-line interface

(a) Server: fshared.c

- As command-line arguments, the server program receives a port number (with option -p) and a directory to share with the client (with option -d)
- Example

```
$ls
.       ..      files
$ls files
.       ..      a.out           hello.txt           main.c
$fshared -p 8080 -d files &
$
```

(b) Client: fshare.c

- The client program receives the IP address of a host machine with a port number (separated with ":") and a user command as command-line arguments.

- Example

```
$ls
.      ..     hi.txt
$fshare 192.168.0.0.1:8080 list
a.out
hello.txt
main.c
$fshare 192.168.0.0.1:8080 get hello.txt
$ls
.      ..     hello.txt    hi.txt
$fshare 192.168.0.0.1:8080 put hi.txt
$fshare 192.168.0.0.1:8080 list
a.out
hello.txt
hi.txt
main.c
$
```

2. Operation

The client program first sends a fixed-size header that indicates the type of a user command and the size of the payload data (i.e., the remaining part of the transmitted data). Following the header, the client transmits data according to the header if necessary, and then shutdowns the writing channel.

Once the connection with a client is established, the server program reads the fixed number of bytes to read the header. Based on the command type, the client reads the payload data and sends a response message. The response message may be the list of the contained files (for a list request), the content of a specific file (for a get request), nothing (for a put request), or an error message. A response message consists of a header and payload. Note that the header for a request (i.e., client sent) and the header for a response (i.e., the server sent) would have different structures.

Work Plan

1. Implement the command-line interface part of fshared.c and fshare.c first
   - Use Makefile to manage build process
   - Test the interfaces thoroughly considering exceptional cases
2. Design protocol (e.g., header structure, message data formats) for each request type
   - Your implementations must be compatible with each other, even they are written by different persons.
3. Complete the features for the listing request (in both client and server), and test your implementation to see they are compatible with each other.
4. List up all cases that the server rejects a user request and sends an error message.
5. Implement the remaining features