Introduction to Network Programming

Makeup Exam

Date: 2020/06/08 Time: 15:30-18:30

1) (50%) UDP file downloader

Implement a UDP Server and UDP Client. There are some files storing on the server side. Your client should be able to get multiple files it needs from the server. Your server can only bind to an assigned port and the server will wait for the client's request on this port and send the requested files to the client.

Command format:

- Server
  - **./server {port}**

  Example:

  ```
  ./server 12345
  ```

  Server can only receive the client's request using port 12345 (12345 is just for example. It can be any valid port)

- Client
  - **./client {server-ip} {server-port} {file-name1} {file-name2} {file-name3}…**

  Client can get variable number of files from the server. When the client receives the file, the received file name must be the same as the original file name.

  Example:

  ```
  ./client 127.0.0.1 12345 file1 file2
  ```

  file1 and file2 exist on the server. Client will get **file1** and **file2** from the server.

Notes:

- The specified files will locate on the same directory as the server program.
- You can assume the requested files always exist on the server.
- We will ONLY test **one** client at the same time.

2) (50%) Implement a TCP server. The server does the following functions:

- Server will give client a name by connection order. For example, the first client connects to the server, it will be named user1, the second client will be user2, the third client will be user3. **When user2 disconnect and connect to the server again, it will be named user4.**
- When client connect/disconnect to the server, **server** should output message: **New connection from {ip}:{port} {user#} / {user#} {ip}:{port} disconnected**

The client does the following functions:

- Generally, it is broadcast to everyone who is online except himself, but every user knows who they don't want to send a message to.
- The service accepts the following commands and at least 10 clients: Remember to handle incomplete client input.

Command	**Description**	**Output**
list-users	List all online users.	1. Success: user1 user2...
list-blacklist	List all users in blacklist.	2. Success: user1 user2...
blacklist <username>	Keep <username> from receiving my message.	1. Success: Blacklist <username> successfully. 2. Fail 1: User is not online: (first judgment) <username> does not exist. 3. Fail 2: Repeated blacklist: <username> is already blacklisted.

whitelist <username>	Let <username> receive my message.	1. Success: Whitelist <username> successfully. 2. Fail 1: User is not online: (first judgment) <username> does not exist. 3. Fail 2: Has not been blacklisted: <username> has not been blacklisted.
broadcast <message>	Broadcast <message> to all user except users I refused and me.	1. Online users did not be blacklisted： From < broadcaster> : <message>
exit	Disconnect from the server.	

Set up:

Download **np_makeup_exam.zip** from new E3. After extracting this file, you will see the following directory structure:

np_final_exam/

└── **setup.sh** --- This script is used to set up directories of your submission.

Submission:

Change directory to **np_makeup _exam**. Type **./setup.sh <your_student_id>** to set up the directories. Then, you will see the following directories.

<your_student_id>/

├── P1/ --- This directory is for storing your answer to the problem 1.

 ├── server/

  Put your **server codes** and a readme file that describes how to compile your program (Makefile is better) here.

 └── client/

  Put your **client codes** and a readme file that describes how to compile your program (Makefile is better) here.

└── P2/ --- This directory is for storing your answer to the problem 2.

  Put your **codes** here and a readme file that describes how to compile your program (Makefile is better) here.

Type **zip –r <your_student_id> <your_student_id>** and upload the **<your_student_id>.zip** to new E3.