Computer Network

Report for Hw1

資工三 B00902064 宋昊恩

1. Implementation:

I first implemented both server and client parts with the simplest functionality, which can just inform each other with strings. To let server support multi-client service, I used the concept of `epoll`, which can be easily created, added, and waited.

I will describe my design for server, client and executor, which is the handler for commands from client, in the following.

For the server, I first get the socket, bind myself to one of the port, and listen to the client connection. Then, I use `epoll` here to check whether my input is from client or from finished jobs. Once I notice that there is a client asking me for connection, I will accept it and add it to my care-list. Or, if I notice that it is a client requesting me to do something, I will do some simple parsing first. And, if I can handle it, I will do it myself, or I will tune the pipe direction and fork the executor, throw the entire parsed command to it, and wait for it to response. If the response is immediately, I will soon send it back to client or write in the certain file, or I will keep the pipe for later reading. I spent the majority of my work time here to deal with the parsing and postpone issue.

For the client, it is much easier. I first get the socket, and try to connect to the certain address with certain port. I used a simple test here to judge whether the address is a real `IP` address or just a hostname. After that, I use `epoll` here again to check whether the input information comes from standard input`,` or from the socket. If it is from standard input, I will transmit it to server, or I will receive the message and print it on the standard output`.`

For the executor, I used a quite interesting implementation – I execute the command recursively. To be more clear, I first divide the command into to-be-done part and remaining part. If the remaining part is not empty, I will tune the pipe direction and fork another executor to execute the remaining command. After that, I will execute the to-be-done command directly. There are two situations here, if the command is legal, everything just go right, if the command is illegal, it will send the error information back to the server but keep doing the remaining part of commands.

For different clients, I keep a structure of information for each of them, including the socket file descriptor, postponing requests, and path information.

2. Encountered difficulties:

First, I did not close the pipe carefully, and thus spent a lot of time on finding the solution of not blocking `grep` (the solution is `stdbuf).` Second, I found out that `grep` cannot parse single and double quote, so I have to deal with it myself. Third, I spent a lot of time on changing the vector and map to alternatives. I still want to comment that not using the `STL` library is real an inefficient and unwise implementation method, however, I will respect to the requirement from teaching assistants.