Programming exercise report

105061110 周柏宇

1. Implementation

Server-side:

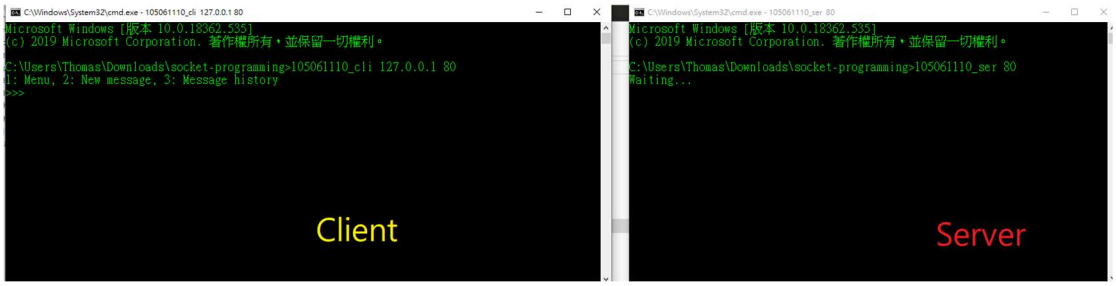
We maintain a finite state machine to define the transition of 3 states – Menu, add a new message and read all messages. In Menu state, we can either press 2 or 3 to activate the function of adding a new message and reading all messages, respectively. In Adding a message state, we accept keyboard input and store it in the 'msg_history' array. In Read all messages state, we return all the history messages but divide them into batches to save bandwidth. Since return all messages at once every time we want to read messages can be wasteful.

Client-side:

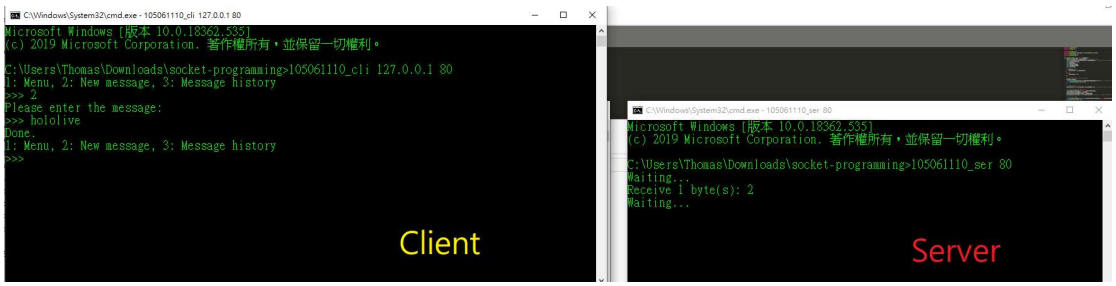
After connection is established, the client simply follows the instruction – pressing 123 to select functions and output what server returns.

2. Screenshots

- Starting screen

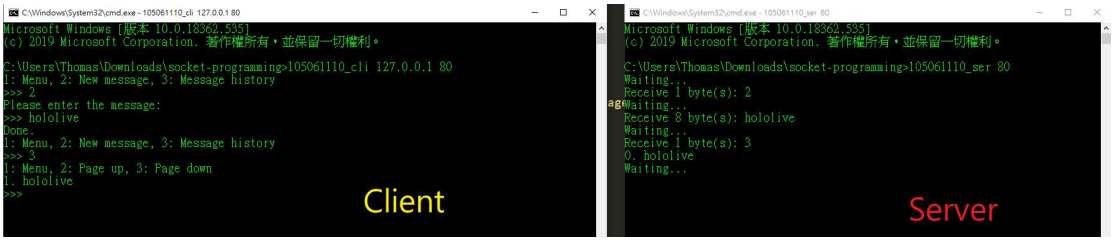


- Press 2 to add a new message and return to the starting screen after done entering.

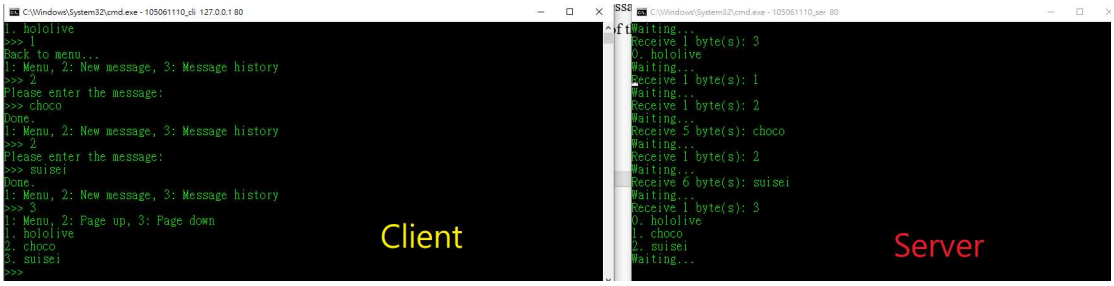


- Press 3 to browse all messages. The max messages in a page is by default

32 so that the server won't need to send all of them at once. Press 2 or 3 to page up and down.



More messages



3. Difficulties and solutions

- To accept the keyboard input, using 'scanf' will only capture the alphabets before any 'space' are type. Alternatively, I use 'fgets' to capture any message with spaces.
- To implement the page of messages, I need to add extra 70 lines of code to make the output format appealing to users.
- Some times we are lazy to type the address and port. The default port for server and client are port 80 and the default server is the local host (127.0.0.1).