Course: Computer Networks Professor Year S. Sun

Programming Assignment B. Server

執行環境:

系上工作站 (ubuntu) / macOS

作業執行檔案於 macOS 編譯。

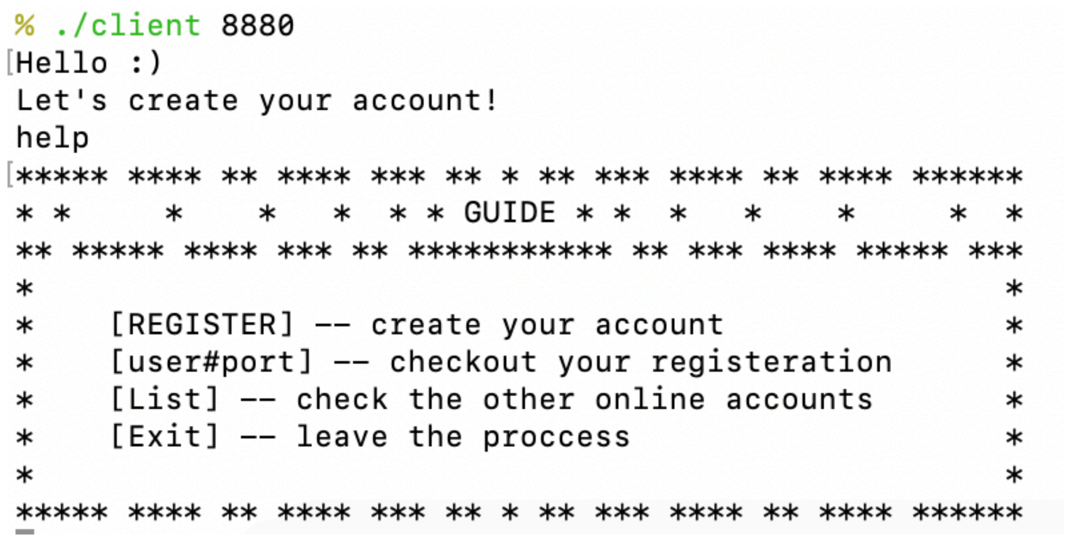
編譯方式:

打開 terminal 移到 makefile 檔案夾，下「make」指令。

make clean 指令可以把 server.out 刪除。

新增功能：

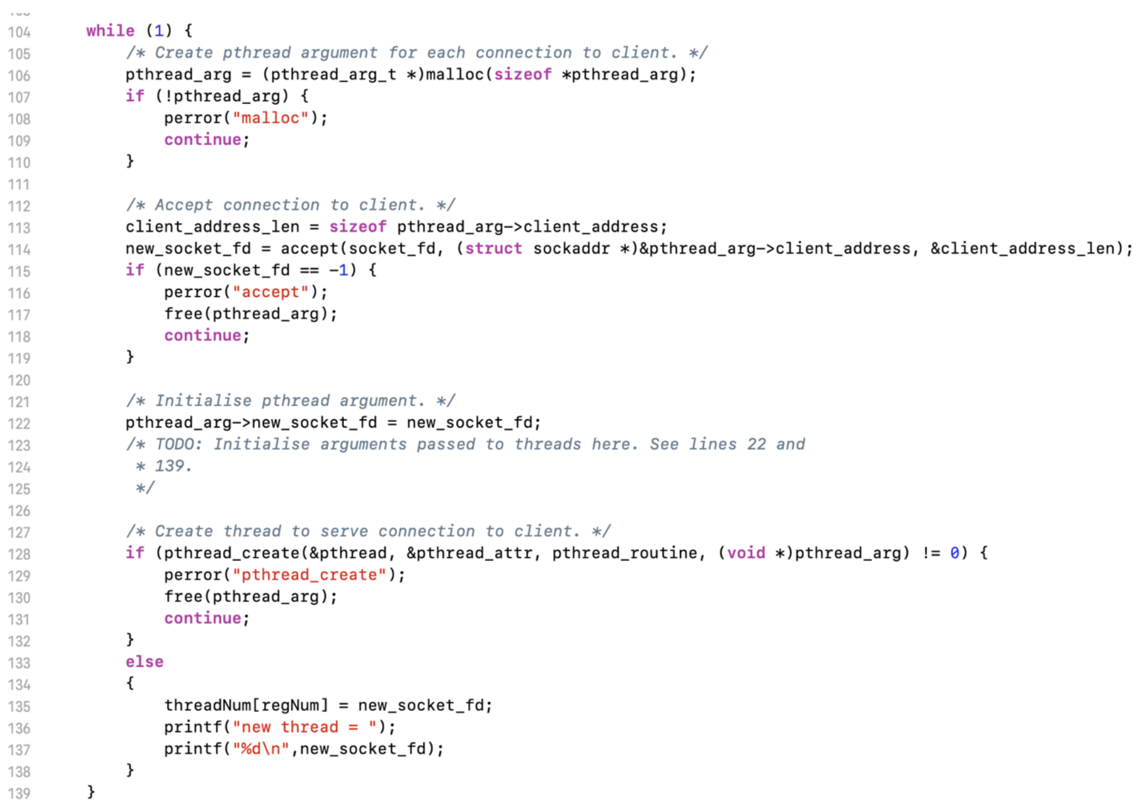
help — 印出指令



```
% ./client 8880
Hello :)
Let's create your account!
help
***** **** ** **** *** ** * ** *** **** ** **** ******
* *     *     *    *    * * GUIDE * *   *    *     *      *  *
** ***** **** *** ** *********** ** *** **** ***** ***
*                                                     *
*    [REGISTER] -- create your account                *
*    [user#port] -- checkout your registeration       *
*    [List] -- check the other online accounts        *
*    [Exit] -- leave the proccess                     *
*                                                     *
***** **** ** **** *** ** * ** *** **** ** **** ******
```

其餘功能皆與助教 server 相同。

程式碼部分說明：



```c
104     while (1) {
105         /* Create pthread argument for each connection to client. */
106         pthread_arg = (pthread_arg_t *)malloc(sizeof *pthread_arg);
107         if (!pthread_arg) {
108             perror("malloc");
109             continue;
110         }
111 
112         /* Accept connection to client. */
113         client_address_len = sizeof pthread_arg->client_address;
114         new_socket_fd = accept(socket_fd, (struct sockaddr *)&pthread_arg->client_address, &client_address_len);
115         if (new_socket_fd == -1) {
116             perror("accept");
117             free(pthread_arg);
118             continue;
119         }
120 
121         /* Initialise pthread argument. */
122         pthread_arg->new_socket_fd = new_socket_fd;
123         /* TODO: Initialise arguments passed to threads here. See lines 22 and
124          * 139.
125          */
126 
127         /* Create thread to serve connection to client. */
128         if (pthread_create(&pthread, &pthread_attr, pthread_routine, (void *)pthread_arg) != 0) {
129             perror("pthread_create");
130             free(pthread_arg);
131             continue;
132         }
133         else
134         {
135             threadNum[regNum] = new_socket_fd;
136             printf("new thread = ");
137             printf("%d\n",new_socket_fd);
138         }
139     }
```

每個 client 成功連線後，會給一個 thread。
目前上限設為 10，可更改全域變數 BLOCKLOG。
若有 client 登出 thread 會被重複利用（thread pool）。
(抱歉 DEMO 的時候不知道 thread pool 名詞定義，如果我的理解沒錯上述是有用 thread pool 做沒錯)

```c
void *pthread_routine(void *arg) {
    pthread_arg_t *pthread_arg = (pthread_arg_t *)arg;
    int new_socket_fd = pthread_arg->new_socket_fd;
    struct sockaddr_in client_address = pthread_arg->client_address;
    /* TODO: Get arguments passed to threads here. See lines 22 and 116. */

    free(arg);

    /* TODO: Put client interaction code here. For example, use
     * write(new_socket_fd,,) and read(new_socket_fd,,) to send and receive
     * messages with the client.
     */
    char message[1024] = {};
    char client_message[1024] = {};
    int n = 0;
    n = write(new_socket_fd, "Hello :)\n", strlen("Hello :)\n"));
    n = write(new_socket_fd, "Let's create your account!\n", strlen("Let's create your account!\n"));

    //Receive msg from client
    while(recv(new_socket_fd , client_message , 1024 , 0) > 0)
    {
        if(NULL != strstr(client_message, "REGISTER"))//REGISTER
        {
            int check = TRUE;
            char *ret = strchr(client_message,'#')+1; // UserName

            for(int i = 0; i < regNum; i++)
            {
                if(0 == strcmp(ret, name[i]))
                {
                    check = FALSE;
                    n = write(new_socket_fd, "210 FAIL\n", strlen("210 FAIL\n"));
                    break;
                }
            }
```

每個連線的 client 就會跑 pthread_routine 這個 function。