IP : 140.113.207.51
Port : 8008

Introduction of Network Programming 2018 Autumn
Homework 2 - Social Service: Client

Description

In this assignment, you are going to create a client side program of a social service. Your program should be able to receive user command from **standard input**, send the request to the server through socket, and deal with the server response from socket.

Requirement

The service provide the following commands for user:

Command Format	Description	Example response from server (in JSON format)	

register <*id*> <*password*>	Register an account with <id> and <password>.	Success	{
			“status”: 0,
			“message”: “Success!”
			}

		Fail	{
			“status”: 1,
			“message”: “<*id*> is already used”
			}
			or
			{
			“status”: 1,
			“message”: “Usage: register <*id*> <*password*>”
			}

login <*id*> <*password*>	Login with the id and password. Server reply an access token. Your program should store users’ access tokens during the program execution.	Success	{
			“status”: 0,
			“token”: “11d28a07-2091-4c57-ba46-7fb1bc488cd6”,
			“message”: “Success!”
			}

		Fail	{
			“status”: 1,
			“message”: “No such user or password error”
			}
			or
			{
			“status”: 1,
			“message”: “Usage: login <*id*> <*password*>”
			}

delete <*user*>	Delete a user account.	Success	{
			“status”: 0,
			“message”: “Success!”
			}

		Fail	{
			“status”: 1,
			“message”: “Not login yet”
			}
			or
			{
			“status”: 1,
			“message”: “Usage: delete <*user*>”
			}

logout <*user*>	Logout the user	Success	{
			“status”: 0,
			“message”: “Bye!”
			}

		Fail	{
			“status”: 1,
			“message”: “Not login yet”
			}
			or
			{

			“status”: 1, “message”: “Usage: logout <*user*>” }

invite <*user*> <*id*>	Invite another user to become your friend.	Success	{
			“status”: 0,
			“message”: “Success!”
			}

		Fail	{
			“status”: 1,
			“message”: “<*id*> is already your friend”
			}
			or
			{
			“status”: 1,
			“message”: “<*id*> does not exist”
			}
			or
			{
			“status”: 1,
			“message”: “Not login yet”
			}
			or
			{
			“status”: 1,
			“message”: “You cannot invite yourself”
			}
			or
			{
			“status”: 1,
			“message”: “Already invited”
			}
			or
			{
			“status”: 1,
			“message”: “<id> has invited you”
			}
			or
			{
			“status”: 1,
			“message”: “Usage: invite <*user*> <*id*>”
			}

list-invite <*user*>	List all the users who invited you to become friends.	Success	{
			“status”: 0,
			“invite”: [
			user_A,
			user_B
			]
			}

		Fail	{
			“status”: 1,
			“message”: “Not login yet”
			}
			or
			{
			“status”: 1,
			“message”: “Usage: list-invite <*user*>”
			}

accept-invite <*user*> <*id*>	Accept the invitation of being friends.	Success	{
			“status”: 0,
			“message”: “Success!”
			}

		Fail	{
			“status”: 1,
			“message”: “<*id*> did not invite you”
			}
			or
			{
			“status”: 1,
			“message”: “Not login yet”
			}
			or
			{
			“status”: 1,
			“message”: “Usage: accept-invite <*user*> <*id*>”
			}

list-friend <*user*>	List all the friends.	Success	{
			“status”: 0,
			“friend”: [
			user_A,
			user_B
			]
			}

		Fail	{
			“status”: 1,
			“message”: “Not login yet”
			}
			or
			{
			“status”: 1,
			“message”: “Usage: list-friend <*user*>”
			}

post <*user*> <*message*>	Share a post with your friends	Success	{
			“status”: 0,
			“message”: “Success!”
			}

		Fail	{
			“status”: 1,
			“message”: “Not login yet”
			}
			or
			{
			“status”: 1,
			“message”: “Usage: post <*user*> <*message*>”
			}

receive-post <*user*>	Receive the posts from your friends	Success	{
			“status”: 0,
			“post”: [
			{
			“id”: user_A,
			“message”: “I have no friends”
			},
			{
			“id”: user_B,
			“message”: “I have no friends too”
			}
			]
			}

		Fail	{
			“status”: 1,
			“message”: “Not login yet”
			}
			or
			{
			“status”: 1,
			“message”: “Usage: receive-post <*user*>”
			}

General

- In order to connect to server, your client program **must** accept two **command line arguments: ip and port**.
- For the commands that need authentication, your program **must** help user maintain their **access tokens**. Before sending those commands to server, you **must** replace the <*user*> with their corresponding token. You can acquire the token via ***login*** command. Of course, you have to ***register*** the account first. For example, when user type in “list-friend userA”, your program **must** replace userA with userA’s access token, and send it to the server. If user did not login yet, you should just leave <*user*> empty.
- **Please note that you will need to serve more commands in homework 4. Take care about the extensibility and readability.**
- **Please note that you should send the request immediately once you connect to the server. Otherwise, the server will close the connection. The timeout is 100 millisecond.**

Input Command

- The message in post command **allows spaces**.
- There’s a special command ***exit***. It will clear all the token store in client side and exit client program.

Output Message

- For the responses which contain “**message**”, you **must** directly print the message with newline.
- For the responses from ***list-invite*** and ***list-friend***, you **must** print each element with newline
- For the responses from ***receive-post***, you **must** print each element in
  “<id>: <message>” format with newline.
- If the above three commands return an empty list, you **must** print messages like “**No posts**”, “**No friends**”, “**No invitations**”.

Sample Input

```
register testA 111
register testB 222
register testC 333
login testA 111
login testB 222
login testC 333
invite testA testB
accept-invite testB testA
invite testB testC
accept-invite testC testB
list-friend testA
list-friend testB
list-friend testC
post testA Hi I am A
post testB Hi I am B
post testC Hi I am C
receive-post testA
receive-post testB
receive-post testC
delete testA
delete testB
delete testC
exit
```

Sample Output

```
Success!
Success!
Success!
Success!
Success!
Success!
Success!
Success!
Success!
Success!
testB
testA
testC
testB
Success!
Success!
Success!
testB: Hi I am B
testA: Hi I am A
testC: Hi I am C
testB: Hi I am B
Success!
Success!
Success!
```

Grade (100%)

- Connect to server through command line arguments - (5%)
- ***exit*** command - (5%)
- Your program can maintain multiple users' access token - (10%)
- Each command: 8%, 10 commands in total - (80%)

Submit

Please upload a zip file called "hw2_{$student_id}.zip" that includes your source code. Submission that don't follow the rule will get 20% punishment on the grade.

Demo time will be announced before the deadline. You are not allowed to modify your code after deadline. Please submit your code on time.

If you have any questions, please ask your questions on course forum(https://e3new.nctu.edu.tw/)

Reference

1. **JSON format (https://zh.wikipedia.org/wiki/JSON)**
2. **C socket (http://man7.org/linux/man-pages/man2/socket.2.html)**
3. **Python socket (https://docs.python.org/3/library/socket.html)**
4. **C JSON (https://github.com/json-c/json-c)**
5. **Python JSON (https://docs.python.org/3/library/json.html)**