BÀI TẬP TUẦN SỐ 6 – IT4062

Yêu cầu nộp bài:

- Đặt mã nguồn của mỗi chương trình vào thư mục riêng rẽ có tên như gợi ý ở dưới.
- Tạo Makefile để biên dịch đồng thời các chương trình với tên file chạy lần lượt là **server** và **client**
- Đóng gói các thư mục này vào file nén có tên theo dịnh dạng HotenSV_MSSV_HW06.zip. Ví dụ với bài tập tuần này, cấu trúc file nén nộp như sau:

```
HotenSV_MSSV_HW06.zip
      |-- TCP_Client
           |-- Các file mã nguồn
      |-- TCP_Server
           |-- Các file mã nguồn
      |-- Makefile
```

Mô tả bài tập

Sử dụng TCP socket và kỹ thuật đa luồng để xây dựng ứng dụng đăng bài cho người dùng.

- Server khởi động với số hiệu cổng là giá trị truyền qua tham số dòng lệnh:

  `$./server Port_Number`

  Ví dụ: `$./server 5500`

- Client khởi động với địa chỉ server là các giá trị truyền qua tham số dòng lệnh có cú pháp như sau:

  `$./client IP_Addr Port_Number`

  Ví dụ: `$./client 10.0.0.1 5500`

Yêu cầu:

- Sử dụng giao thức được mô tả ở mục sau đây.
- Trên server, tài khoản người dùng lưu trong file văn bản account.txt, mỗi dòng một tài khoản dạng(xem file ví dụ):

  `username status`

  Trong đó giá trị status là 0 nếu tài khoản bị khóa, là 1 nếu tài khoản hoạt động.
- Người dùng cần phải đăng nhập bằng tên tài khoản đã có trên hệ thống trước khi gửi bài đăng. Người dùng có thể gửi nhiều bài đăng trong một phiên.
- Người dùng có thể kết thúc phiên bằng cách gửi yêu cầu đăng xuất hoặc tắt chương trình client.
- Trên mỗi cửa sổ chương trình client, người dùng chỉ đăng nhập được 1 tài khoản.
- **Mỗi tài khoản chỉ được đăng nhập trên 1 client**
- Chương trình client cần phải có giao diện để giao tiếp người dùng.

Giao thức của ứng dụng:

Client và server trao đổi các thông điệp có cú pháp như sau:

Chức năng	Thông điệp yêu cầu	Thông điệp trả lời
		100: khi kết nối tới dịch vụ thành công
Đăng nhập	USER *username* Trong đó *username* là tên tài khoản người dùng nhập từ bàn phím	110: nếu tài khoản tồn tại và hoạt động 211: nếu tài khoản bị khóa 212: nếu tài khoản không tồn tại **213: nếu tài khoản đã được đăng nhập trên client khác** 300: nếu không xác định được kiểu thông điệp yêu cầu
Đăng bài	POST *article* Trong đó *article* là nội dung bài đăng người dùng nhập từ bàn phím.	120: nếu đăng bài thành công 221: nếu chưa đăng nhập 300: nếu không xác định được kiểu thông điệp yêu cầu
Đăng xuất	BYE	130: nếu đăng xuất thành công 221: nếu chưa đăng nhập 300: nếu không xác định được kiểu thông điệp yêu cầu

Lưu ý: Sinh viên tự quyết định thông điệp định dạng(theo chuỗi byte hay chuỗi ký tự) và kỹ thuật xử lý truyền dòng.

Thang điểm:

- Điểm chức năng(FS):

  [1] Sử dụng đúng kỹ thuật đa luồng: 2 điểm

  [2] Xử lý truyền dòng thành công: 1 điểm

  [3] Xử lý đăng nhập đúng theo giao thức: 1 điểm

  [4] Xử lý đăng bài đúng theo giao thức: 1 điểm

  [5] Xử lý đăng xuất đúng theo giao thức: 1 điểm

  [6] Có điều độ tiến trình: 1 điểm

  [4] Một số lỗi sau đây bị trừ điểm:

  - Server không phục vụ được liên tục cho nhiều client: -2 điểm
  - Các tình huống khiến server không thể xử lý thêm các yêu cầu: -2 điểm
  - Lỗi runtime error khiến client kết thúc: -1 điểm
  - Lỗi runtime error khiến server kết thúc nhưng vẫn kiểm thử được các chức năng: -3 điểm
  - Client không có giao diện phù hợp, thông báo giao tiếp người dùng: -1 điểm
  - Không sử dụng kỹ thuật đa luồng và giao thức đã mô tả: -100%
  - Lỗi runtime error khiến không thể kiểm thử được tất cả chức năng: -100%
  - Lỗi biên dịch: -100%
  - Các lỗi khác: trừ điểm tùy theo mức độ nghiêm trọng của lỗi

- Điểm Tổ chức và trình bày mã nguồn(SS):

  Nếu $0 \leq FS < 1$: $SS = 0$

Nếu 1 ≤ FS < 3: SS tối đa là 1 điểm

Nếu 3 ≤ FS ≤ 5: SS tối đa là 2 điểm

Nếu 5 < FS ≤ 7: SS tối đa là 3 điểm

Yêu cầu môi trường:

- Hệ điều hành: Ubuntu 20.04
- Trình biên dịch: GCC

Gợi ý:

- Trạng thái đăng nhập trong phiên cần phải được quản lý tại server là luôn kiểm tra mỗi khi có yêu cầu từ client. Mỗi phiên cung cấp dịch vụ cho 1 client trên server nên được lưu trữ dưới dạng cấu trúc. Cấu trúc nên có các trường thông tin sau:
  - Định danh socket trên server
  - Địa chỉ của client
  - Tên tài khoản đã đăng nhập tại client
  - Trạng thái đăng nhập

  Trạng thái đăng nhập cần được cập nhật khi xử lý thành công các yêu cầu đăng nhập và đăng xuất.
- Một kịch bản kiểm thử các chức năng có thể như sau(không bắt buộc tuân theo):

Bước	Cửa sổ Client 1	Cửa sổ Client 2	Cửa sổ Client 3
1	Khởi động client 1		
2		Khởi động client 2	
3			Khởi động client 3
4			(1) Yêu cầu: Đăng nhập với tài khoản tungbt Kết quả: Thành công (2) Yêu cầu: Đăng nhập với tài khoản admin Kết quả: Thất bại (3) Yêu cầu: Đăng bài với số lần tùy ý Kết quả: Thành công (4) Yêu cầu: Đăng xuất Kết quả: Thành công (5) Yêu cầu: Đăng nhập với tài khoản tungbt

			Kết quả: Thành công (6) Đóng cửa sổ
5		(1) Yêu cầu: Đăng bài Kết quả: Thất bại (2) Yêu cầu: Đăng xuất Kết quả: Thất bại (3) Yêu cầu: Đăng nhập với tài khoản tungbt Kết quả: Thành công	
6	(1) Yêu cầu: Đăng nhập với tài khoản ductq Kết quả: Thất bại (2)Yêu cầu: Đăng nhập với tài khoản admin Kết quả: Thành công (3) Yêu cầu: Đăng bài với số lần tùy ý Kết quả: Thành công		
7		(1) Yêu cầu: Đăng bài với số lần tùy ý Kết quả: Thành công	
8	(1) Yêu cầu: Đăng xuất Kết quả: Thành công		
9		(1) Yêu cầu: Đăng xuất Kết quả: Thành công	

Sử dụng chương trình kiểm thử: Chương trình kiểm thử cho trường hợp sử dụng định dạng theo chuỗi ký tự và xử lý truyền dòng bằng mẫu kết thúc "\r\n". Sinh viên thay đổi cho phù hợp với kỹ thuật đã sử dụng nếu cần.

- Biên dịch: **`gcc -pthread tcp_test.c -o test`**
- Cú pháp: **`test <#server_port> <#threads>`**

  Trong đó:

  #server_port: Số hiệu cổng ứng dụng của server

  #threads: Số luồng kiểm thử đồng thời

  Ví dụ: **`test 5500 10`**