**BÀI TẬP TUẦN 3 – LẬP TRÌNH MẠNG**

**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\_HW03.zip. Ví dụ với bài tập tuần này, cấu trúc file nén nộp như sau:

HotenSV\_MSSV\_HW03.zip

|**--** UDP\_Client

|-- Các file mã nguồn

|-- UDP\_Server

|-- Các file mã nguồn

|-- Makefile

**Mô tả bài tập**

**Viết ứng dụng phân giải tên miền sử dụng UDP socket:**

**Server:**

* Chạy ở số hiệu cổng bất kỳ dùng tham số dòng lệnh theo cú pháp sau:

**./server PortNumber**

**Ví dụ: ./server 5500**

* Nhận một thông điệp chứa tên miền hoặc địa chỉ IP do client gửi lên
* Trả lại thông điệp chứa kết quả phân giải cho client. Ghi log kết quả xử lý
* Yêu cầu bắt buộc: **Chương trình server không được tự động kết thúc trong mọi tình huống xử lý.**

**Client:**

* Khởi động chương trình với tham số dòng lệnh cho địa chỉ IP và số hiệu cổng của server sẽ gửi yêu cầu tới theo cú pháp sau

**./client IPAddress PortNumber**

**Ví dụ: ./client 127.0.0.1 5500**

* Người dùng nhập tên miền hoặc địa chỉ IP từ bàn phím
* Client gửi yêu cầu tới server
* Nhận kết quả từ server và hiển thị
* Chức năng lặp lại cho tới khi người dùng nhập vào một xâu rỗng.

Server cần ghi lại nhật ký hoạt động vào file có tên log\_MSSV.txt., ví dụ log\_20201234.txt Mỗi dòng có cấu trúc như sau:

[dd/mm/yyyy hh:mm:ss]$Yêu cầu nhận được$Kết quả truy vấn

Trong đó:

*dd/mm/yyyy*: Định dạng ngày nhận yêu cầu

*hh:mm:ss*: Định dạng thời điểm nhận yêu cầu

Ví dụ:

[31/03/2023 14:42:24]$google.com$+216.58.197.110 216.58.197.123 126.58.99.199

[31/03/2023 14:42:28]$126.58.99.199$+hkg07s22-in-f3.net hkg07s22-in-f99.net

[31/03/2023 14:43:28]$5giay.vn$+210.211.109.164

[31/03/2023 14:43:29]$aznsc.test.com$–Not found information

**Thang điểm:**

* Điểm chức năng(FS):

1. Thực hiện phân giải thuận: 3 điểm

a. Hiển thị được địa chỉ IP đầu tiên: 1 điểm

b. Hiển thị được toàn bộ địa chỉ IP: 2 điểm

1. Thực hiện phân giải ngược: 1 điểm
2. Ghi log: 1 điểm

[3] Thiết kế thông điệp trả lời có prefix: 2 điểm

[4] Một số lỗi sau đây bị trừ điểm:

* Server không phục vụ được liên tục cho nhiều client: -3 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
* 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 ≤ 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:**

**1. Thiết kế thông điệp trả lời từ server**

Kết quả trả về từ sever có 2 loại:

* Thông điệp chứa kết quả
* Thông điệp báo lỗi.

Cần thiết kế để phân biệt 2 loại thông điệp này bằng cách sử dụng prefix (Ký tự/Nhóm ký tự bắt đầu thông điệp) khác nhau. Client dựa trên prefix để xác định thông điệp có chứa kết quả dạng nào.

Một gợi ý thiết kế như sau:

* Prefix là ‘+’ để báo thành công. Ví dụ: +202.191.56.65
* Prefix là ‘–‘ để báo thất bại. Ví dụ: –Not found information

**2. Các vấn đề về tổ chức, trình bày mã nguồn**

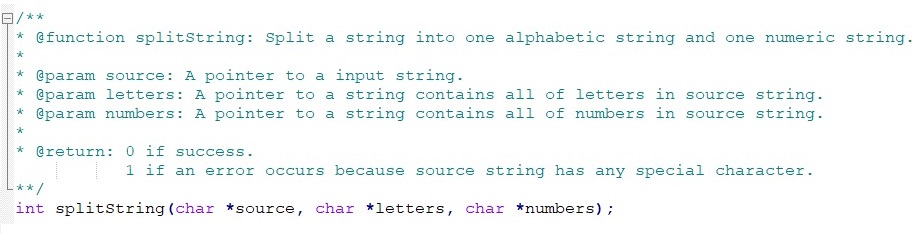
**-** Clean code: https://viblo.asia/p/tom-tat-cuon-clean-code-cua-uncle-bob-6J3Zg07MlmB

- Xây dựng chương trình thành các hàm mô-đun chức năng

- Sử dụng quy ước định danh. Ví dụ:

* Camel case: <https://en.wikipedia.org/wiki/Camel_case>
* Snake case: <https://en.wikipedia.org/wiki/Snake_case>
* Quy ước khác: https://viblo.asia/p/naming-rules-cac-quy-tac-vang-trong-lang-dat-ten-ByEZkMXE5Q0

- Comment cho các hàm mô-đun theo quy tắc thống nhất. Ví dụ:



**3. Kiểm thử**

- Kiểm thử chức năng: Sinh viên có thể tự thực hiện kiểm thử theo kịch bản sau(không bắt buộc tuân theo):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bước | Cửa sổ Command Prompt 1 | Cửa sổ Command Prompt 2 | Cửa sổ Command Prompt 3 | Cửa sổ Command Prompt 4 |
| 1 | Khởi động server |  |  |  |
| 2 |  | Khởi động client 1 |  |  |
| 3 |  |  | Khởi động client 2 |  |
| 4 |  |  |  | Khởi động client 3 |
| 5 |  |  |  | Nhập 1 xâu nào đó |
| 6 |  |  | Nhập 1 xâu nào đó |  |
| 7 |  | Nhập các xâu kiểm thử tùy ý |  |  |
| 8 |  | Nhập xâu rỗng |  |  |
| 9 |  |  | Nhấn Ctrl+C |  |
| 10 |  |  |  | Nhập các xâu kiểm thử tùy ý |
| 12 |  | Khởi động client 1 |  |  |
| 13 |  | Nhập 1 xâu nào đó |  |  |
| 14 |  | Nhấn Ctrl+C |  |  |
| 15 |  |  |  | Nhập các xâu kiểm thử tùy ý |
| 16 |  |  |  | Nhập xâu rỗng |

- Kiểm thử hiệu năng: Sinh viên có thể sử dụng chương trình udp\_test.c.

* Biên dịch: **gcc -pthread udp\_test.c -o test**
* Cú pháp: **test <#server\_port> <#threads>**

Trong đó:

#server\_port: Số hiệu cống ứng dụng của server

#threads: Số luồng kiểm thử đồng thời

Ví dụ: **test 5500 10**