ĐỒ ÁN THỰC HÀNH 1 – LẬP TRÌNH SOCKET

MÔN MẠNG MÁY TÍNH

1. Quy định chung

- Đồ án được làm theo nhóm: mỗi nhóm tối đa **3** sinh viên, tối thiểu **2** sinh viên, sinh viên tự chọn nhóm (sử dụng nhóm thực hành đã đăng ký nếu có), những sinh viên làm 1 mình phải gửi email cho GV (LT hoặc TH) và trình bày lý do. Nhóm sinh viên sẽ chọn đề tài thỏa quy định sau:

Mã đề tài = (tổng chữ số cuối cùng MSSV của các sinh viên) mod 2 + 1

- **Các bài làm giống nhau sẽ đều bị điểm 0 toàn bộ phần thực hành tất cả các nhóm liên quan (dù có điểm các bài tập, đồ án thực hành khác)**.

- Môi trường lập trình: Tự do lựa chọn ngôn ngữ lập trình, tự do lựa chọn môi trường hệ điều hành: Windows, Unix/Linux, macOS

- Ngôn ngữ lập trình GV có thể hỗ trợ: C/C++, C#, Java, Python

- Thư viện hỗ trợ lập trình socket cho phép sử dụng: Socket, CSocket, winsock. Tức là chỉ sử dụng các thư viện Socket do ngôn ngữ lập trình cung cấp, còn những phần không liên quan tới socket thì có thể dùng thư viện bên ngoài. **Đây là bài tập Socket, không phải lập trình website.**

2. Cách thức nộp bài

- **Nộp bài trực tiếp trên Website môn học, không chấp nhận nộp bài qua email hay hình thức khác.**
- Tên file: **MÃ-ĐỀ_ MSSV1_MSSV2_MSSV3.zip** (Với MSSV1 < MSSV2 < MSSV3)

Ví dụ: Nhóm gồm 2 sinh viên: 2012001, 2012002, và 2012003 làm đề 1, tên file nộp: **1_2012001_2012002_2012003.zip**

Cấu trúc file nộp gồm:

1. **Report.pdf:** chứa báo cáo về bài làm
2. **Release**: thư mục chứa file thực thi của chương trình, **nếu có** (*.exe/ ...) (nếu làm Python thì không cần tạo file exe, chỉ cần report những thư viện cài thêm trong report nếu có)
3. **Source**: thư mục chứa source code của chương trình , yêu cầu nộp cả project đã xoá bỏ thư mục Debug và các file không cần thiết khác.. *Nhóm nào chỉ nộp file *.cpp và *.h và không biên dịch được thì bị 0 điểm.*

Lưu ý: Cần thực hiện đúng các yêu cầu trên, nếu không, bài làm sẽ không được chấm.

3. Hình thức chấm bài

Chấm vấn đáp vào thời điểm kết thúc phần thực hành.

4. Tiêu chí đánh giá

Về chương trình:

- Mục tiêu của đồ án này tập trung chủ yếu vào 2 vấn đề: lập trình socket, xây dựng giao thức trao đổi giữa client và server. Do đó các tiêu chí đánh giá dựa vào các chức năng chính được liệt kê trong yêu cầu của chương trình (có ghi chú thang điểm cho từng chức năng)

Về báo cáo:

- Thông tin của nhóm.
- Đánh giá mức độ hoàn thành từ 0 – 100% (Chú thích rõ những mục làm được,chưa làm được và còn bị lỗi)
- Kịch bản giao tiếp của chương trình: Giao thức trao đổi giữa client và server, cấu trúc thông điệp, kiểu dữ liệu của thông điệp, cách tổ chức cơ sở dữ liệu (nếu có).
- Môi trường lập trình và các framework hỗ trợ để thực thi ứng dụng.
- Hướng dẫn sử dụng các tính năng chương trình.
- Bảng phân công công việc và cho biết rõ ràng ai làm việc gì một cách rõ ràng.
- Các nguồn tài liệu tham khảo.

Lưu ý: Trong báo cáo không dán các đoạn source code của chương trình. Mã chương trình chỉ trình bày nếu thật sự cần thiết và nếu cần minh họa cho các mô hình cài đặt hay các cơ chế đồng bộ (minh họa dạng mã giả, prototype hàm).

Về vấn đáp:

- Chuẩn bị thiết bị, chương trình, báo cáo đầy đủ (không cần in).
- Trả lời các câu hỏi từ GV
- Trường hợp trả lời sai hoặc không trả lời được sẽ trừ trực tiếp điểm vào tổng điểm đồ án.

Lưu ý: Tất cả thành viên của nhóm phải tham gia buổi vấn đáp. Thành viên vắng mặt sẽ xử lý theo quy định sau:

- Có phép (gửi email xin phép trước buổi vấn đáp): trừ điểm vấn đáp trực tiếp
- Không phép: 0 điểm toàn đồ án.

HyperText Transport Protocol (HTTP)

Web browser (Chrome, IE, Firefox, Safari …) sử dụng HyperText Transport Protocol (HTTP) để truy xuất các trang web (page) từ một máy chủ (Web Server). Browser tạo kết nối TCP/IP đến WebServer, gửi một HTTP request cho máy chủ thông qua kết nối đó, đọc kết quả trả về (HTTP response) và hiển thị nội dung trang web (page) cho người dùng. Cả hai HTTP requests và HTTP responses đều là dạng text, có thể dễ dàng đọc hiểu nội dung.

Một HTTP request bao gồm:

- Một dòng request line
- Tiếp theo là một hay nhiều header lines chứa những nội dung bổ sung
- Phần body nếu có

(Xem thêm slide bài giảng lý thuyết).

Để truy xuất một web page, web browser sử dụng "*GET*" method, chỉ định page cần truy xuất và version của HTTP protocol được sử dụng (Version hiện tại là HTTP/1.1 hoặc HTTP/2).
Ví dụ: web browser gửi yêu câu (request) "*GET /index.html HTTP/1.1*" để truy xuất page "*/index.html*" từ một máy chủ. "*GET*" request bắt buộc phải kèm theo một header để chỉ định tên của website, ví dụ: "*Host: www.fit.hcmus.edu.vn*" (trong trường hợp có nhiều website cùng chạy trên cùng một máy chủ).

Ví dụ, để truy xuất trang web http://example.com/index.html , web browser sẽ tạo một kết nối đến máy chủ example.com với port 80, và gửi HTTP request như sau:

```
GET /index.html HTTP/1.1\r\n
Host: example.com\r\n
\r\n
```

Chú ý, mỗi dòng sẽ kết thúc với “\r\n”, và toàn bộ request GET sẽ kết thúc với "\r\n", Ví dụ ở trên là yêu cầu nhỏ nhất của một HTTP request. Web browser thường sẽ kèm nhiều những header khác ngoài header "*Host:*", để điều khiển kết nối, chỉ định format hoặc ngôn ngữ, cookies, ...

Khi web server nhận một HTTP GET request cho một page tồn tại, nó sẽ trả lời "*HTTP/1.1 200 OK*" response, kèm theo một số header lines để cung cấp một số thông tin cho response, một dòng trống, và cuối cùng là body của page. Những header lines nên có một "*Content-Length:*" header, header này sẽ chỉ định kích thước body của page dưới dạng bytes.
Ví dụ một HTTP reponse ("..." là text đã rút gọn)

```
HTTP/1.1 200 OK\r\n
Accept-Ranges: bytes\r\n
Age: 471725\r\n
Cache-Control: max-age=604800\r\n
Content-Type: text/html; charset=UTF-8\r\n
Date: Wed, 12 Oct 2022 09:38:18 GMT\r\n
Etag: "3147526947"\r\n
Expires: Wed, 19 Oct 2022 09:38:18 GMT\r\n
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT\r\n
Server: ECS (oxr/8321)\r\n
```

```
Vary: Accept-Encoding\r\n
X-Cache: HIT\r\n
Content-Length: 1256\r\n
\r\n
<!doctype html>
<html>
<head>
      <title>Example Domain</title>
...
</body>
</html>
```

Chú ý, mỗi dòng sẽ kết thúc với “\r\n”, và giữa header và body sẽ cách biệt bằng "\r\n".

Ví dụ ở trên, "Content-Length:" là 1256 bytes, nghĩa là sẽ có chính xác 1256 bytes trong phần body của reponse (Tính từ ký tự "<" của dòng "<!doctype html>" và kết thúc với ">" của dòng "</html>").
Nếu một request truy cập một page không tồn tại, server có thể response with mã lỗi 404 "file not found". Sẽ có "Content-Type: text/html" header, còn body của response chứa nội dung HTML thông báo lỗi.

```
HTTP/1.1 404 Not Found\r\n
Cache-Control: max-age=604800\r\n
Content-Type: text/html; charset=UTF-8\r\n
Date: Wed, 12 Oct 2022 10:25:15 GMT\r\n
```

```
Expires: Wed, 19 Oct 2022 10:25:15 GMT\r\n
Server: EOS (vny/0452)\r\n
Vary: Accept-Encoding\r\n
Content-Length: 1256\r\n
\r\n
<!doctype html>
<html>
<head>
      <title>Example Domain</title>
...
</body>
</html>
```

Ngoài ra còn có những reponse khác, được phân biệt dựa vào con response code trong dòng đầu tiên của response. (*Xem thêm slide bài giảng về response code: 200, 201, 400, 404, 500, 502…*).

ĐỀ 1

Web Server

Nội dung:

Mục tiêu là viết một web server đơn giản. Webserver sẽ bind trên **port 8080**. Bạn có thể dùng bất kỳ trình duyệt web nào làm client (Firefox, Chrome, Safari, ...) để gửi request đến web server và webserver sẽ gửi lại response cho client (nội dung của page, hoặc là lỗi). Nội dung của trang web đã được lập trình sẵn, web server sẽ đọc nội dung file và trả về cho client (src_html_de01 folder)

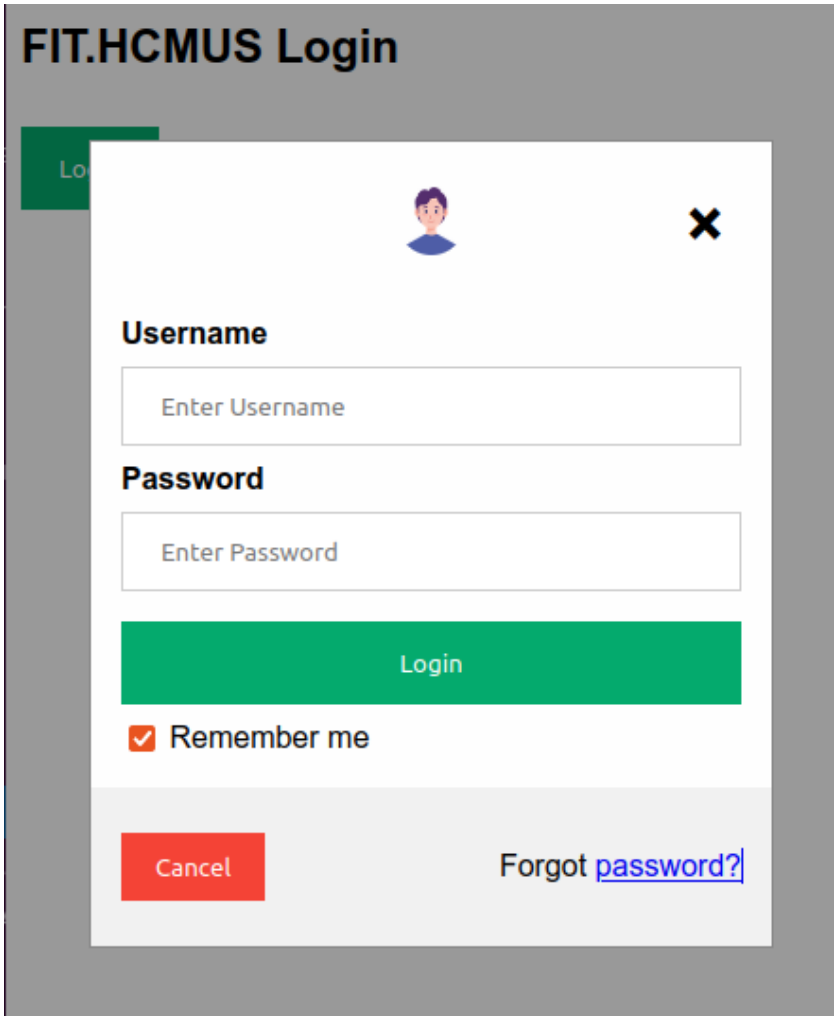
Handle connection cơ bản:

Server tạo một TCP socket, bind trên port 8080, sau đó listen và accept connection từ browser. Khi chấp nhận kết nối từ browser, server đọc và parse request. Dòng đầu tiên của data (request line) sẽ kết thúc bằng (\r\n) để xác định được kiểu của HTTP request tạo bởi browser.

- Nếu request bắt đầu với "GET" và theo sau đó là "/" hoặc tên filename (ví dụ index.html, css/style.css, ...), cuối cùng là HTTP/1.1, server cần phải parse thông tin của request để biết được cần phải load file nào cho request đó. Nếu là "/" thì coi như "index.html". Tên filename sẽ liên quan đến thư mục của máy chủ tương ứng (i.e. css/style.css, avatars/1.png, ...). Sau khi parse request để xác định filename, server sẽ gửi reponse với HTTP header tương ứng, kèm theo là body (nội dung của file). **Sau đó đóng kết nối (không giữ connection - không Keep alive)**
    - Nếu filename tồn tại, thì trả về thành công "200 OK" kèm theo nội dung của file

```
HTTP/1.1 200 OK\r\n
Content-Type: text/html\r\n
Connection: close\r\n
\r\n
<!DOCTYPE html>
<html>
...
```

Phần "..." trong mẫu trên là những phần nội dung tiếp theo của trang HTML. Nội dung của body phải đọc từ file. Ví dụ sau khi web browser nhận data:



- Nếu file không tồn tại, trả về "404 File Not Found". Đây là một ví dụ của một response "404 File Not Found"

```
HTTP/1.1 404 Not Found\r\n
```

```
Content-Type: text/html\r\n
Connection: close\r\n
\r\n
<!DOCTYPE html>
<html>
<head>
<title> 404 Not Found </title>
</head>
<body>
<p> The requested file cannot be found. </p>
</body>
</html>
```

- Nếu request bắt đầu với "POST", thì server sẽ nhận được thông tin "uname" và "psw" kèm theo trong body của request. Server cần parse thông tin này ra và kiểm tra nếu "uname" là "admin" và "psw" là "123456" thì trả về nội dung của trang "images.html". Nếu không thì trả về "401 Unauthorized". Đây là một ví dụ của một response "401 Unauthorized"

```
HTTP/1.1 401 Unauthorized\r\n
Content-Type: text/html\r\n
Connection: close\r\n
\r\n
<!DOCTYPE html>
<h1>401 Unauthorized</h1><p>This is a private area.</p>
```

Images

Chỉ định content type:

Trong nội dung của các page, có kèm theo hình ảnh (png và jpg), file CSS, được link vào trong HTML của pages. Để cho browser nhận dạng được những images hoặc file CSS, bạn cần phải kèm theo "*Content-Type:"* header tương ứng trong response của server. "*Content-Type:*" phục thuộc vào loại file của filename.

Ví dụ:

Filename	Content-Type:
*.html, *.htm	Content-Type: text/html
∗.txt	Content-Type: text/plain
∗.jpg, ∗.jpeg	Content-Type: image/jpeg
∗.gif	Content-Type: image/gif
*.png	Content-Type: image/png
*.css	Content-Type: text/css
(unknown)	Content-Type: application/octet-stream

Bạn sẽ cần phải parse filename trong HTTP GET request để xác định được loại file, sau đó điền "*Content-Type:*" tương ứng khi gửi response.

Nâng cao:

- Cho phép gửi nhiều request cho mỗi kết nối: Hiện tại web browser mở một TCP connection cho mỗi request khi nhiều file cần được truy cập => ko hiệu quả, HTTP cho phép gửi nhiều request trên một kết nối trước đó. Nếu server không có header "*Connection: close*" trong response, client có thể giữ kết nối, và tiếp tục gửi những request tiếp theo. Để cho phép Client có thể phân biệt data từ nhiều request, server bắt buộc phải có "*Content-Length:*" header để chỉ định kích thước của data trả về cho client. Kết nối chỉ đóng lại khi client đóng kết nối, khi đó hàm "*recv*" trên server sẽ return 0 khi kết nối bị đóng.

- Cho phép nhiều clients truy cập cùng lúc (Handling Multiple Connections in Parallel, concurrency by thread for each connection, ...)

Yêu cầu:

	Chức năng	Ý nghĩa	Test-cases
1	**KẾT NỐI**	**0,5 điểm** Cho phép client kết nối đến server thông qua kết nối TCP	
2	**QUẢN LÝ KẾT NỐI**	**0,5 điểm** Khi client hoặc server mất kết nối đột ngột, không làm chương trình treo hay xảy ra lỗi	
3	**Tải được page index.html**	**3.5 điểm** Browser có thể render lên đầy đủ nội dung của trang index.html. Trang index.html hiển thị 1 HTML form cho phép đăng nhập (yêu cầu 4). Form này có phần action trỏ đến Web server. Tham khảo:	http://localhost:8080 http://localhost:8080/index.html http://<ip>:8080 http://<ip>:8080/index.html

		https://www.w3schools.com/html/html_forms.asp	
4	**ĐĂNG NHẬP**	**2 điểm** **POST** method, gửi "**uname**" là "**admin**" và "**psw**" là "**123456**"	• Kiểm tra login trả 401 nếu không đúng (0.5đ) • Kiểm tra login (0.5đ) • Load được trang images.html khi đăng nhập đúng (1đ)
5	**Lỗi page**	**0.5 điểm** Trả 404 khi load page không đúng	http://localhost:8080/gacon http://localhost:8080/fit.html http://<ip>:8080/gacon http://<ip>:8080/fit.html ...
6	**Multiple requests**	**1 điểm** gửi nhiều requests trong một connection	
7	**Multiple connection**	**1 điểm** Concurrent, handle nhiều client cùng lúc	

8	**Report**	**1 điểm** Theo quy định ở trên	

ĐỀ 2

Web Client

Nội dung:

Mục tiêu là viết một web client đơn giản. Web client sẽ gửi request đến web server, port 80 để tại nội dung của page và lưu vào file

Handle connection cơ bản:

Client tạo một TCP socket, kết nối đến **port 80** của web server. Sau khi kết nối đến web server, client sẽ gửi request đến web server để tải page. Client yêu cầu sử dụng HTTP/1.1 và "*Connection: keep-alive*", khi đó hàm "recv" sẽ vẫn chờ nhận dữ liệu mặc dù đã nhận đủ nội dung file. Sinh viên cần tính toán để biết khi nào đã nhận xong data và đóng kết nối. Vậy làm sao Client có thể biết được là đã nhận đủ data, thì bạn dựa vào một trong các trường hợp:

- "*Content-Length:*" cho biết nội dung của body hoặc
- "*Transfer-Encoding: chunked*", khi đó webserver sẽ trả về dữ liệu từng chunk, làm sao nhận biết được độ dài 1 chunk và khi nào nhận hết các chunk, Tham khảo: https://en.wikipedia.org/wiki/Chunked_transfer_encoding, https://bunny.net/academy/http/what-is-chunked-encoding/

Yêu cầu tải và lưu file

- File download hay folder download sẽ nằm cùng cấp với binary
- Với các request là "/" gốc thì mặc định tải và lưu thành file “<domain>_index.html”, Ví dụ

    - http://example.com hoặc http://example.com/, thì client sẽ tải và lưu thành file "example.com_index.html"
    - http://www.bing.com hoặc http://www.bing.com/, thì client sẽ tải và lưu thành file "www.bing.com_index.html"
- Với các request tải một file, ví dụ *.html, *.pdf, *.jpg ..., thì client sẽ tải và lưu thành tên file tương ứng kèm theo domain theo cấu trúc: "*<domain>_<tenfile>"*. Ví dụ:
    - http://example.com/index.html, thì sẽ tải và lưu thành file “example.com_index.html”
    - http://web.stanford.edu/dept/its/support/techtraining/techbriefing-media/TB-080610-ProtectYourData.ppt, thì sẽ tải và lưu thành file “web.stanford.edu_TB-080610-ProtectYourData.ppt”
- Với request là các subfolder, thì download hết tất cả các file trong folder đó kèm domain theo cấu trúc: "*<domain>_<tenfolder>"*. Ví dụ:
    - http://web.stanford.edu/class/cs224w/slides/ thì tải và lưu hết tất cả các file vào folder “web.stanford.edu_slides”, trong folder đó sẽ chứa các file tương ứng mà không có domain kèm theo.

Nâng cao:

- Cho phép gửi nhiều request cho mỗi kết nối: Hiện tại client mở một TCP connection cho mỗi request khi nhiều file cần được truy cập => ko hiệu quả, HTTP cho phép gửi nhiều request trên một kết nối trước đó. Nếu server không có header "*Connection: close*" trong response, client có thể giữ kết nối, và tiếp tục gửi những request tiếp theo nếu client gửi có header "*Connection: keep-alive*". Để cho phép Client có thể phân biệt data từ nhiều request, server bắt buộc phải có "*Content-Length:*" hoặc "*Transfer-Encoding: chunked"* header để chỉ định kích thước của data trả về cho client. Kết nối chỉ đóng được khi client đóng kết nối, khi đã tải xong hoàn toàn dữ liệu. Áp dụng khi download nhiều file trong subfolder. Ví dụ:
    - ./<tên file chạy> http://web.stanford.edu/class/cs224w/slides/
- Cho phép client tải nhiều trang cùng lúc (Handling Multiple Connections in Parallel, concurrency by thread for each connection, ...). Áp dụng khi Client có nhiều link download cùng lúc. Ví dụ:
    - ./<tên file chạy> http://web.stanford.edu/class/cs231a/project.html http://web.stanford.edu/class/cs224w/slides/08-GNN-application.pdf ...

Yêu cầu:

	Chức năng	Ý nghĩa	Test-cases
1	**KẾT NỐI**	**0,5 điểm** Cho phép client kết nối đến server thông qua kết nối TCP	
2	**QUẢN LÝ KẾT NỐI**	**0,5 điểm** Khi client hoặc server mất kết nối đột ngột, không làm chương trình treo hay xảy ra lỗi	
3	**Tải và lưu thành file dạng "**Content-Length**"**	**3 điểm** Download file thành công	http://example.com/ http://example.com/index.html http://web.stanford.edu/dept/its/support/techtraining/techbriefing-media/Intro_Net_91407.ppt http://web.stanford.edu/class/cs224w/slides/01-intro.pdf http://web.stanford.edu/class/cs224w/slides/08-GNN-application.pdf http://web.stanford.edu/class/cs231a/assignments.html

			http://web.stanford.edu/class/cs231a/project.html http://gaia.cs.umass.edu/wireshark-labs/alice.txt http://www-net.cs.umass.edu/wireshark-labs/Wireshark_Intro_v8.1.docx ...
4	**Tải và lưu thành file dạng "***Transfer-Encoding: chunked***"**	**1.5 điểm** Download file thành công	http://www.google.com http://www.google.com/index.html http://www.bing.com http://anglesharp.azurewebsites.net/Chunked http://www.httpwatch.com/httpgallery/chunked/chunkedimage.aspx ...
5	**Tải các file trong folder**	**1.5 điểm** Lưu tất cả các file trong folder	http://web.stanford.edu/class/cs224w/slides/ http://web.stanford.edu/class/cs142/lectures/ http://web.stanford.edu/class/cs143/handouts/ http://web.stanford.edu/class/cs231a/course_notes/ ...

6	**Multiple requests**	**1 điểm** gửi nhiều requests trong một connection khi download file trong folder	
7	**Multiple connection**	**1 điểm** Concurrent, handle nhiều kết nối cùng lúc đến các web servers	
8	**Report**	**1 điểm** Theo quy định ở trên	