BÀI TẬP THỰC HÀNH
MÔN LẬP TRÌNH MẠNG

Phần 2. Xây dựng ứng dụng web server

Mục đích: Sinh viên sử dụng kiến thức đã tìm hiểu trong bài thực hành trước để lập trình tạo một web server trả về cấu trúc thư mục của ổ đĩa trên server. Tùy thuộc vào trình duyệt gửi yêu cầu nội dung của thư mục hay của tập tin mà server sẽ trả về dữ liệu thích hợp.

I. Sử dụng hàm Windows API để tìm danh sách thư mục/file

- **Mục đích:** Sử dụng hàm **FindFirstFileA()** và **FindNextFileA()** để lấy danh sách thư mục và tập tin của một thư mục bất kỳ.
- Cấu trúc **WIN32_FIND_DATAA** được sử dụng để lưu dữ liệu trả về bởi 2 hàm trên.

```
typedef struct _WIN32_FIND_DATA {
  DWORD    dwFileAttributes;          // Thuộc tính file
  FILETIME ftCreationTime;
  FILETIME ftLastAccessTime;
  FILETIME ftLastWriteTime;
  DWORD    nFileSizeHigh;             // Kích thước file
  DWORD    nFileSizeLow;              // Kích thước file
  DWORD    dwReserved0;
  DWORD    dwReserved1;
  TCHAR    cFileName[MAX_PATH];       // Tên file
  TCHAR    cAlternateFileName[14];
} WIN32_FIND_DATA, *PWIN32_FIND_DATA, *LPWIN32_FIND_DATA;
```

- Các trường cần quan tâm gồm **dwFileAttributes** chứa thuộc tính của thư mục/tập tin, được sử dụng để phân biệt thư mục hoặc tập tin, **cFileName** chứa tên của thư mục/tập tin, **nFileSizeHigh** và **nFileSizeLow** chứa kích thước của tập tin.
- Kích thước của file được xác định theo công thức:

```
(nFileSizeHigh*(MAXDWORD+1))+nFileSizeLow
```

- Đoạn mã sau được sử dụng để duyệt tất cả các thư mục và tập tin trong thư mục C:

```
WIN32_FIND_DATAA DATA;
HANDLE h = FindFirstFileA("C:\\*.*",&DATA);
do {
     if (DATA.dwFileAttributes &
FILE_ATTRIBUTE_DIRECTORY){
          // Đây là một thư mục
          // In ra tên thư mục
     } else {
          // Đây là một file
          // In ra tên file và kích thước file
     }
} while (FindNextFileA(h, &DATA));
```

- **Yêu cầu:** Sử dụng đoạn mã trên để in ra màn hình nội dung của một thư mục bất kỳ (bao gồm tên các thư mục con, tập tin và kích thước).
- **Kết quả cần báo cáo:** Mã nguồn chương trình và kết quả chạy thử

II. Dùng Netcat tìm hiểu cách thức web server trả lời các yêu cầu

Mục đích: Sinh viên sử dụng công cụ Netcat để gửi yêu cầu đến website, sau đó quan sát dữ liệu trả về để tìm hiểu cách thức web server trả lời yêu cầu từ các client.

- **Bước 1**: Sử dụng công cụ Netcat để tạo client kết nối đến website genk.vn ở cổng 80. Do dữ liệu trả về có thể bao gồm nhiều dòng, vượt quá khả năng hiển thị của màn hình dòng lệnh, sinh viên sử dụng lệnh để chuyển hướng kết quả vào file output.txt

```
nc.exe –v genk.vn 80 > c:\output.txt
```

- **Bước 2**: Nhập lệnh để gửi yêu cầu đến web server

```
GET / HTTP/1.1<ENTER>
Host: genk.vn<ENTER>
<ENTER>
```

- **Bước 3:** Mở file **c:\output.txt** ra xem đoạn đầu tiên sẽ thấy nội dung tương tự như sau:

```
HTTP/1.1 200 OK
Server: Fengine
```

```
Date: Sat, 19 Nov 2016 04:25:59 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: close
Vary: Accept-Encoding
Set-Cookie: device_env=4; expires=Sat, 26-Nov-2016
04:25:59 GMT; path=/; domain=genk.vn
Cache-Control: max-age=120
Last-Modified: Sat, 19 Nov 2016 04:25:04 GMT
ETag: 9d681c0d138c06c067235bdbc401b17e
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
server: web_142-39

ff00
<!DOCTYPE html>
<html>...</html>
```

Trong các đoạn text trên phần bôi đậm là quan trọng nhất mà một web server phải trả về cho trình duyệt.

HTTP/1.1 200 OK để báo cho trình duyệt biết là yêu cầu đã được phục vụ.

Content-Type: text/html báo cho trình duyệt biết kiểu dữ liệu sẽ trả về là html.

<html>...</html> là nội dung của trang web mà trình duyệt sẽ hiển thị.

- **Kết quả cần báo cáo:** Các bước thực hiện và kết quả đạt được.

III. Lập trình một HTTP File server

Mục đích: Sinh viên lập trình tạo một web server thực hiện công việc sau: Khi trình duyệt nhận được địa chỉ ***localhost:port*** từ người sử dụng, web server sẽ nhận được yêu cầu thư mục gốc: **GET / HTTP/1.1** thì trả về nội dung bao gồm toàn bộ tên các thư mục và tập tin có trong ổ **C:**, mỗi tên trên một dòng.

- **Bước 1:** Khi nhận được yêu cầu từ trình duyệt có định dạng **GET x HTTP/1.1** thì phân tích chuỗi **x** để xem trình duyệt yêu cầu gì?
- **Bước 2:** Nếu x bằng "/" thì có nghĩa trình duyệt yêu cầu trang chủ, trong trường hợp này ta coi đó là thư mục gốc (ổ C:) và thực hiện bước tiếp theo, nếu x khác "/" thì đóng kết nối, không xử lý yêu cầu.
- **Bước 3:** Khi x bằng "/" thì duyệt nội dung ổ C: như ở bài I và tạo ra một chuỗi html để chứa tên các thư mục và file. Ví dụ:

  `"<html><b>Temp</b><br><i>log.txt</i><br></html>"`

  Trong đó: <html> và </html> là các tag mở và đóng đoạn mã html
   là cặp tag để in đậm nội dung bên trong

 tag để xuống dòng
   là cặp tag để in nghiêng nội dung bên trong

  Dựa vào ví dụ trên, hãy hiển thị mỗi thư mục hoặc tập tin trên một dòng, tên thư mục in đậm, tên tập tin in nghiêng.
- **Bước 4:** Gửi phản hồi cho trình duyệt như mô phỏng trong phần II. Ghép chuỗi html đã tạo ở bước 3 với các headers. Ví dụ:

  ```
  "HTTP/1.1 200 OK\nContent-Type: text/html;
  charset=utf-
  8\n\n<html><b>Temp</b><br><i>log.txt</i><br></html>"
  ```
- **Bước 5:** Đóng kết nối.
- **Bước 6:** Chạy chương trình, sử dụng trình duyệt để gửi yêu cầu đến server đã tạo và kiểm tra kết quả.

- **Kết quả cần báo cáo:** Mã nguồn chương trình và kết quả chạy thử

Chú ý: Sau khi kết thúc bài thực hành, sinh viên nộp các kết quả đạt được và mã nguồn vào Google Form theo địa chỉ:

https://goo.gl/forms/lQpXOrRHHCX0Mcab2