LỜI CẢM ƠN

Đối với mỗi Sinh viên Đồ Án là một dạng bài tập lớn, bằng cách vận dụng những kiến thức lý thuyết đã được học cộng với sự hướng dẫn của GVHD và tự tìm tòi tham khảo để cụ thể kiến thức đó thành một sản phẩm,với Sinh Viên ngành Công Nghệ Thông Tin năm thứ tư như chúng em Đồ Án Cơ Sở Ngành Mạng là một trong những Đồ Án quan trọng nhất. Đồ Án gồm hai phần là Hệ Điều Hành và phần Mạng.

Với những môn học bổ trợ ở các kỳ trước như “ Kiến trúc máy tính và vi xử lý”, Nguyên lý hệ điều hành” hay “Mạng máy tính, Lập trình mạng”… đã mang lại những kiến thức cơ bản về để giúp chúng em trong việc chọn đề tài phù hợp với mục tiêu là hiểu rõ hơn về sự hoạt động của hệ điều hành, cũng như là sự hoạt động của các giao thức mạng. Bên cạnh đó là những môn học về giải thuật và lập trình từ trước tới giờ để cung cấp cho chúng em việc lựa chọn thuật toán hợp lý để giải quyết vấn đề, cũng như chọn ngôn ngữ phù hợp đề thực hiện.

Để hoàn thành được Đồ Án lần này chúng em xin gửi lời cảm ơn đến sự giúp đỡ của các thầy cô trong khoa, đặc biệt là Thầy **Phạm Minh Tuấn** – người trực tiếp hướng dẫn chúng em thực hiện, cảm ơn sự tận tình chỉ bao và giải đáp thắc mắc của thầy.

Xin chân thành cảm ơn !

Đà Nẵng, tháng 5 năm 2017

Sinh viên

Nguyễn Quốc Anh

Mục lục

[PHẦN I: NGUYÊN LÝ HỆ ĐIỀU HÀNH 4](#_Toc483640226)

[**TIÊU ĐỀ: XÂY DỰNG CHƯƠNG TRÌNH THEO DÕI CÁC HOẠT ĐỘNG CỦA NGƯỜI DÙNG TRÊN HỆ ĐIỀU HÀNH WINDOWS.** 4](#_Toc483640227)

[CHƯƠNG I : CƠ SỞ LÝ THUYẾT 4](#_Toc483640228)

[**I.** **Giới thiệu đề tài:** 4](#_Toc483640229)

[**II.** **Cơ chế hoạt động của hook:** 4](#_Toc483640230)

[**III.** **Hook Procedures** 5](#_Toc483640231)

[**IV.** **Cài đặt một Hook** 6](#_Toc483640232)

[**V.** **Thư viện liên kết động DLL** 8](#_Toc483640233)

[**V.1.** **Khái niệm:** 8](#_Toc483640234)

[**V.2.** **Cách thức hoạt động** 8](#_Toc483640235)

[**V.3.** **Lợi ích của DLL:** 8](#_Toc483640236)

[CHƯƠNG II: PHÂN TÍCH THIẾT Kế HỆ THỐNG 9](#_Toc483640237)

[**I.** **Khởi tạo** 9](#_Toc483640238)

[**II.** **Cài đặt hook để bắt sự kiện** 9](#_Toc483640239)

[**III.** **Xử lý** 9](#_Toc483640240)

[CHƯƠNG III: TRIỀN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ 10](#_Toc483640241)

[**I.** **Triển khai** 10](#_Toc483640242)

[I.1. Môi trường phát triển. 10](#_Toc483640243)

[I.2. Mã nguồn: Mã nguồn chương trình được đính kèm trong CD 10](#_Toc483640244)

[**II.** **Kết quả đạt được** 10](#_Toc483640245)

[**III.** **Kết quả và hướng phát triển** 10](#_Toc483640246)

[II.1. Những kết quả đạt được 10](#_Toc483640247)

[II.2. Những vấn đề tồn tại 10](#_Toc483640248)

[II.3. Hướng phát triển 10](#_Toc483640249)

[PHẦN II: LẬP TRÌNH MẠNG 12](#_Toc483640250)

[TIÊU ĐỀ: XÂY DỰNG CHƯƠNG TRÌNG FTP 12](#_Toc483640251)

[CHƯƠNG I : CƠ SỞ LÝ THUYẾT 12](#_Toc483640252)

[**I.** **GIAO THỨC TCP/IP** 12](#_Toc483640253)

[I.1. Tầng liên kết 12](#_Toc483640254)

[I.2. Tầng Internet 12](#_Toc483640255)

[I.3. Tầng giao vận 12](#_Toc483640256)

[I.4. Tầng ứng dụng 12](#_Toc483640257)

[**II.** **FILE TRANSFER PROTOCOL** 12](#_Toc483640258)

[II.1. Khái niệm: 13](#_Toc483640259)

[II.2. Mô hình kiến trúc xử lý trong giao thức FTP 13](#_Toc483640260)

[II.3. Quản lý kênh dữ liệu: 15](#_Toc483640261)

[II.4. Các kiểu dữ liệu: 17](#_Toc483640262)

[II.5. Các Command thường gặp: 17](#_Toc483640263)

[CHƯƠNG II: PHÂN TÍCH THIẾT KỆ HỆ THỐNG 18](#_Toc483640264)

[**I.** **PHÂN TÍCH YÊU CẦU** 18](#_Toc483640265)

[I.1. Yêu cầu chức năng: 18](#_Toc483640266)

[I.2 Yêu cầu về giao diện: 18](#_Toc483640267)

[**II.** **THIẾT KẾ HỆ THỐNG** 18](#_Toc483640268)

[CHƯƠNG III: TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ 19](#_Toc483640269)

[**I.** **TRIỂN KHAI** 19](#_Toc483640270)

[**I.1.** **Môi trường phát triển** 19](#_Toc483640271)

[**I.2.** **Mã nguồn:** Được đính kèm trong đĩa CD 19](#_Toc483640272)

[**II.** **KẾT QUẢ ĐẠT ĐƯỢC** 19](#_Toc483640273)

[II.1. Chức năng của chương trình 19](#_Toc483640274)

[II.2. Giao diện chính của chương trình 19](#_Toc483640275)

[II.3. Giao diện sau khi đăng nhập thành công 20](#_Toc483640276)

[II.4. Giao diện download file. 21](#_Toc483640277)

[II.5. Giao diện upload file. 22](#_Toc483640278)

[II.6. Giao diện tạo mới 1 folder 23](#_Toc483640279)

[**III.** **ĐÁNH GIÁ:** 24](#_Toc483640280)

[**IV.** **HƯỚNG PHÁT TRIỂN:** 24](#_Toc483640281)

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# PHẦN I: NGUYÊN LÝ HỆ ĐIỀU HÀNH

**TIÊU ĐỀ: XÂY DỰNG CHƯƠNG TRÌNH THEO DÕI CÁC HOẠT ĐỘNG CỦA NGƯỜI DÙNG TRÊN HỆ ĐIỀU HÀNH WINDOWS.**

# CHƯƠNG I : CƠ SỞ LÝ THUYẾT

1. **Giới thiệu đề tài:**

Trong Windows, khi chúng ta thực hiện các thao tác nhấp chuột, nhấn phím… thì hệ điều hành sẽ chuyển các sự kiện này thành các thông điệp (message) rồi đưa vào hàng đợi (queue) của hệ thống. Sau đó, các thông điệp được trao lại cho từng ứng dụng cụ thể để xử lý.

Hook là một cơ chế mà một ứng dụng có thể chặn các sự kiện, như các thông điệp, thao tác chuột, bàn phím. Hàm dùng để chặn một loại sự kiện riêng biệt được gọi là hook procedure/ hook function. Mỗi khi nhận được sự kiện, hook procedure có thể thay đổi và thậm chí hủy bỏ các sự kiện đó.

Phạm vi hoạt động thì có 2 loại Hook:

- Local hook (thread hook): chỉ có ảnh hưởng trong phạm vi một thread.

- Global hook: có ảnh hưởng trong toàn hệ thống. Trường hợp này, hook procedure phải được chứa trong một thư viện DLL.

Một số ví dụ sử dụng hook:

* Tạo các chức năng debug
* Tạo các chương trình record và play back macro
* Bắt giả lập các sự kiện bàn phím và chuột
* Tạo chức năng Help(F1) cho ứng dụng
* Tạo các ứng dụng computer-based training (CBT)

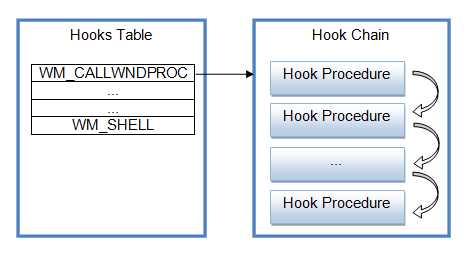
1. **Cơ chế hoạt động của hook:**

Có nhiều loại Hook và hệ điều hành luôn duy trì 1 danh sách các loại hook procedure cho mỗi loại đó. Mỗi danh sách các Hook procedure này được gọi là hook chain. Bản chất của hook chain là một dãy các con trỏ hàm trỏ đến các Hook procedure.

Khi hệ thống thực một sự kiện nào đó, nó sẽ tìm kiếm trong hook chain tương ứng với sự kiện đó. Nếu một hook procedure phù hợp được tìm thấy, hệ thống sẽ thực hiện nó và chỉ lấy lại quyền điều khiển sau khi hook chain kết thúc. Vì thế khi một hook procedure thực hiện xong, nó phải thực hiện việc chuyển quyền điều khiển cho hook procedure kế tiếp trong hook chain.

Tuy nhiên cơ chế này còn tùy thuộc vào loại hook. Như một số loại hook chỉ có thể theo dõi các thông điệp, vì vậy cho dù hook procedure có chuyển quyền điều khiển cho hook procedure kế tiếp hay không, hệ thống vẫn sẽ tự động làm việc này.

Hook là một kỹ thuật xử lý thông điệp rất mạnh cho phép chúng ta can thiệp sâu vào các tiến trình khác nhau, nhưng nó làm ảnh hưởng tới tốc độ của hệ thống, nhất là hook system-wide, vì tất cả các sự kiện của hệ thống sẽ được định hướng tới một hàm nào đó, rõ ràng điều này làm hệ thống chậm đi đáng kể. Vì thế ta chỉ hên hook những thông điệp thật cần thiết và kết thúc việc hook ngay khi không dùng đến nữa.



Hình 1.0: Cơ chế hoạt động của Hook

1. **Hook Procedures**

Hook procedure là một loại [callback function](http://en.wikipedia.org/wiki/Callback_%28computer_programming%29) (hàm hồi quy). Hệ thống sẽ gọi các hàm này khi các sự kiện, thông điệp tương ứng với loại hook. Mỗi loại hook có một hook procedure khác nhau nhưng đều có cùng tham số như cú pháp bên dưới. Với mỗi hook procedure khác nhau thì việc xét các giá trị tham số cũng khác nhau.

Lập trình viên sử dụng dụng hàm SetWindowsHookEx để cài đặt một hook procedures vào hook chain.

Cú pháp chung:

*LRESULT CALLBACK HookProc(int nCode, WPARAM wParam, LPARAM lParam)*

Tham số:

* nCode: giá trị giúp xác định cách xử lý thông điệp. Nếu *nCode* nhỏ hơn 0, hook procedure sẽ gọi CallNextHookEx. Ngược lại sẽ thực hiện các công việc xử lý trước khi gọi CallNextHookEx
* wParam và lParam: tùy theo loại hook sẽ có ý nghĩa khác nhau. Ví dụ nếu bạn hook bàn phím thì wParam sẽ có giá trị của phím được nhấn, lParam sẽ chứa các thông tin về số lần nhấn, các phím tắt, trạng thái phím,…

1. **Cài đặt một Hook**

Để sự dụng hook, chúng ta cần sử dụng ba hàm API của Windows là:

* ***SetWindowsHookEx:*** cài đặt một hook procedure vào một hook chain.
* ***CallNextHookEx:*** chuyển quyền điều khiển cùng các thông tin hook cho hook procedure kế tiếp trong hook chain. Bạn có thể không sử dụng hàm này tuy nhiên chỉ khi nào bạn muốn chặn các hook procedure còn lại trong hook chain.
* ***UnhookWindowsHookEx***: gỡ hook procedure ra khỏi hook chain được cài đặt bởi SetWindowsHookEx.

***SetWindowsHookEx:*** Cú pháp

HHOOK WINAPI SetWindowsHookEx(

\_In\_ int idHook,

\_In\_ HOOKPROC lpfn,

\_In\_ HINSTANCE hMod,

\_In\_ DWORD dwThreadId

);

**Tham số:**

* + idHook:  loại hook, bao gồm các giá trị sau :

Ví dụ:

• WH\_GETMESSAGE : đặt một thủ tục hook quản lý các thông điệp được post tới hàng đợi thông điệp.

• WH\_KEYBOARD : đặt một thủ tục hook quản lý các thông điệp keystroke.

• WH\_MOUSE : đặt một thủ tục hook quản lý các thông điệp chuột. …..

Mỗi giá trị trên xác định một loại hook mà ta muốn cài đặt, mỗi loại hook có một ý nghĩa và tình huống sử dụng khác nhau.

* + **lpfn:** một con trỏ đến hook procedure. Nếu là global hook thì tham số này phải trỏ đến một hook procedure trong một DLL. Ngược lại thì có thể trỏ đến một đoạn mã đóng vai trò hook procedure trong process hiện tại.
  + **hMod:** handle của DLL chứa hook procedure. Trong trường hợp local hook thì tham số này được đặt là NULL.
  + **dwThreadId:** định danh của thread mà hook procedure sẽ gắn vào. Nếu giá trị này là 0 thì mọi thread sẽ bị ảnh hưởng (global), ngược lại thì chỉ có thread được xác định là bị ảnh hưởng (local).

Ví dụ 1: Cài đặt Keyboard hook toàn cục

SetWindowsHookEx(WH\_KEYBOARD, KeyboardProc,hInstDLL, 0);

Ví dụ 2: : Cài đặt Keyboard hook cục bộ:

SetWindowsHookEx(WH\_KEYBOARD, KeyboardProc,NULL,GetCurrentThreadId());

***CallNextHookEx:*** Cú pháp

LRESULT WINAPI CallNextHookEx(

\_In\_opt\_ HHOOK hhk,

\_In\_ int nCode,

\_In\_ WPARAM wParam,

\_In\_ LPARAM lParam

);

***Tham số:***

- hhk : là handle của hook hiện hành, giá trị này có thể lấy được từ hàm SetWindowsHookEx khi cài đặt hook.

- nCode : Giá trị của nó sẽ quyết định hook procedure có thực hiện các công việc xử lý không. Nếu giá trị này là âm, bạn cần gọi hàm CallNextHookEx để chuyển tới hook procedure kế tiếp và bỏ qua bước xử lý (xem phần hook procedure). Giá trị này có được do hệ thống quyết định và bạn không nên thay đối giá trị.

- wParam: chỉ định 16 bits thông tin mở rộng của thông điệp.

- lParam: chỉ định 32 bits thông tin mở rộng của thông điệp.

***Giá trị trả về:*** giá trị trả về là kết quả của quá trình xử lý và tùy thuộc vào thông số nCode.

Trong một số tình huống, hook procedure hiện hành có thể không muốn chuyển sự kiện cho hook procedure khác trong cùng một chuỗi. Lúc này, nếu loại hook chúng ta đang cài đặt cho phép huỷ bỏ sự kiện, và hook procedure của chúng ta cũng có cùng quyết định là hủy bỏ, nó sẽ không phải gọi hàm CallNextHookEx.

***UnhookWindowsHookEx:*** Cú pháp

BOOL WINAPI UnhookWindowsHookEx(

\_In\_ HHOOK hhk

);

Tham số: hhook chỉ ra hàm hook được dỡ bỏ. Đây là giá trị được trả vể bởi hàm SetWindowsHookEx khi hàm Hook được cài đặt.

Nếu thực hiện thành công hàm trả về giá trị true, ngược lại.

Chú ý : Hàm UnhookWindowsHookEx phải được sử dụng trong sự kết hợp với hàm SetWindowsHookEx

1. **Thư viện liên kết động DLL**

**V.1. Khái niệm:**

Là 1 thư viện mà dữ liệu có thể sử dụng bởi nhiều chương trình cùng 1 lúc.

Bằng các sử dụng DLL chương trình có thể được modul hóa thành nhiều phần riêng biệt thuận lợi trong quá trình cập nhật, sửa lỗi

**V.2. Cách thức hoạt động**

Giả sử có 2 file riêng biệt file 1 là “example.exe” - tập tin thực thi thông thường, và "library.dll" – tập tin DLL được sử dụng để thực thi. example.exe (Phải là tập tin đã import library.dll).

Khi chạy example.exe thì loader code sẽ tìm tập tin library.dll. Nếu tìm thấy thì library.dll sẽ đươc chuyển vận.

Bên trong tập tin DLL này có chứa một danh sách khác gọi là "export list", danh sách này kết nối các địa chỉ cụ thể cho từng chức năng nằm bên trong tập tin DLL. Kể từ thời điểm này, khi "example.exe" cần gọi một hàm từ "library.dll", "example.exe" chỉ cần sử dụng địa chỉ đó.

**V.3. Lợi ích của DLL:**

* Tạo khả năng tương tác giữa các ngôn ngữ lập trình.
* Có thế đóng gói và đưa vào các chương trình khác.
* Giảm không gian sử dụng bộ nhớ….

# CHƯƠNG II: PHÂN TÍCH THIẾT Kế HỆ THỐNG

1. **Khởi tạo**

Chương trình được viết bằng ngôn ngữ C#, hỗ trợ gọi đến các hàm API của window

Khi chương trình chạy người dùng gõ bàn phím ở bất kỳ cửa sổ nào thì tên cửa sổ và phím gõ sẽ được lưu trữ trong 1 tập tin XML

1. **Cài đặt hook để bắt sự kiện**

Hàm hook phải được cài đặt trong một file thư viện dll để chương trình chính gọi đến nó.

***SetWindowsHookEx, CallNextHookEx, UnhookWindowsHookEx***

Chúng ta cần phải cài đặt thêm hàm **GetActiveWindowTitle()** để lấy thông tin cửa sổ hiện hành.

1. **Xử lý**

Sau khi cài đặt được những hàm trên.

* Đầu tiên phải tiến hành lấy thông tin cửa sổ hiện tại đưa vào tập tin XML
* Tiếp theo lấy thông tin những phím được nhấn tại cửa sổ đó
* Sau khi người dùng kết thúc cửa sổ đó. (Tức là chuyển sang cửa sổ khác ).In tất cả những phím bấm do người dùng nhập vào XML tại cửa sổ trước đó.
* Quay trở lại bước đầu.

# CHƯƠNG III: TRIỀN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

1. **Triển khai**

### I.1. Môi trường phát triển.

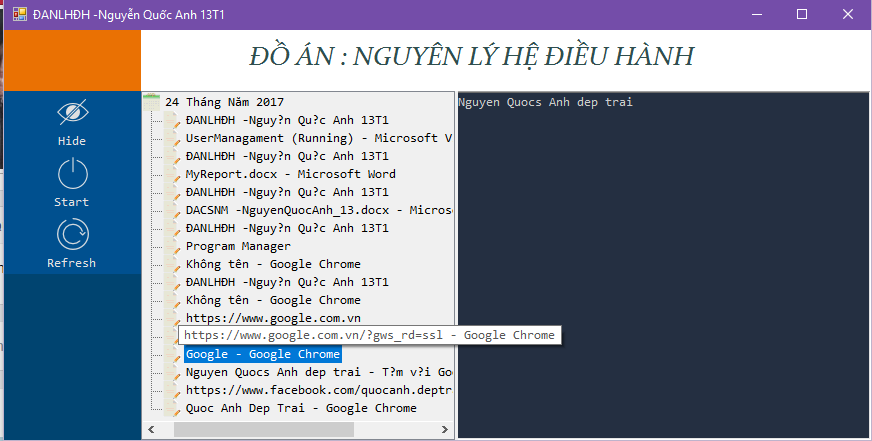
Ứng dụng được phát triển trên

Microsoft Visual Studio Community2015 Version 14.0.25431.01

Microsoft .NET Frameword Version 4.7.02046

### I.2. Mã nguồn: Mã nguồn chương trình được đính kèm trong CD

1. **Kết quả đạt được**



1. **Kết quả và hướng phát triển**

### II.1. Những kết quả đạt được

Chương trình ghi lại 100% thao tác phím

Các tập tin ghi lại theo ngày và tên ứng dụng mà người dùng gõ phím thông tin chính xác 100%

Dữ liệu ghi ra file XML được hiển thị đầy đủ trên chương trình.

Cho phép người dùng dễ dàng theo dõi những hoạt động trên máy cá nhân.

### II.2. Những vấn đề tồn tại

Lỗi font chữ khi lấy tiêu đề cửa sỗ hiện hành vẫn chưa khắc phục được.

### II.3. Hướng phát triển

Cho phép hiển thị Unicode trong bản XML

Tích hợp gủi mail cho chương trình để tự động gửi các bản log về sau 1 ngày.

Chụp màn hình, quay video….

# PHẦN II: LẬP TRÌNH MẠNG

# TIÊU ĐỀ: XÂY DỰNG CHƯƠNG TRÌNG FTP

# CHƯƠNG I : CƠ SỞ LÝ THUYẾT

1. **GIAO THỨC TCP/IP**

### I.1. Tầng liên kết

Là tầng thấp nhất trong mô hình TCP/IP, bao gồm các thiết bị giao tiếp mạng và chương trình cung cấp các thông tin cần thiết để có thể hoạt động, truy nhập đường truyền vật lý qua thiết bị giao tiếp mạng đó.

### I.2. Tầng Internet

Xử lý quá trình truyền gói tin qua mạng. Các giao thức của tầng này bao gồm IP (Internet Protocol), ICMP (Internet Control Message Protocol), IGMP (Internet Group Messages Protocol).

### I.3. Tầng giao vận

Tầng giao vận phụ trách luồng dữ liệu giữa hai trạm thực hiện các ứng dụng của tầng trên. Tầng này có hai giao thức chính: TCP (Transmission Control Protocol) và UDP (User Datagram Protocol) TCP cung cấp một luồng dữ liệu tin cậy giữa hai trạm, nó sử dụng các cơ chế như chia nhỏ các gói tin của tầng trên thành các gói tin có kích thước thích hợp cho tầng mạng bên dưới, báo nhận gói tin, đặt hạn chế thời gian time-out để đảm bảo bên nhận biết được các gói tin đã gửi đi. Do tầng này đảm bảo tính tin cậy, tầng trên sẽ không cần quan tâm đến nữa.

UDP cung cấp một dịch vụ đơn giản hơn cho tầng ứng dụng. Nó chỉ gửi các gói dữ liệu từ trạm này tới trạm kia mà không đảm bảo các gói tin đến được tới đích. Các cơ chế đảm bảo độ tin cậy cần được thực hiện bởi tầng trên.

### I.4. Tầng ứng dụng

Tầng ứng dụng là tầng trên cùng của mô hình TCP/IP bao gồm các tiến trình và các ứng dụng cung cấp cho người sử dụng để truy cập mạng. Có rất nhiều ứng dụng được cung cấp trong tầng này, mà phổ biến là: Telnet: sử dụng trong việc truy cập mạng từ xa, FTP (File Transfer Protocol): dịch vụ truyền tệp, Email: dịch vụ thư tín điện tử, WWW (World Wide Web).

1. **FILE TRANSFER PROTOCOL**

### II.1. Khái niệm:

**FTP (File Transfer Protocol)** là 1 giao thức được phát triển để cho phép truyền tệp tin giữa máy khách (client) và máy chủ (server) dựa trên giao thức TCP/IP .

FTP thường chạy trên cổng 20 (data port) và cổng 21 (command port) và chỉ chạy riêng trên nền tảng TCP.

### II.2. Mô hình kiến trúc xử lý trong giao thức FTP

Giao thức FTP sử dụng kết nối TCP. Nhưng nó không chỉ dùng một kênh TCP như các giao thức truyền thông khác. Nó chia quá trình truyền thông giữa server và client ra làm 2 kênh logic:

* Kênh điều khiển: Kênh này được dùng để khởi tạo 1 phiên kết nối FTP. Nó được duy trì xuyên suốt phiên kết nối FTP và chỉ để truyền các thông tin điều khiển
* Kênh dữ liệu: Mỗi khi dữ liệu được truyền đổi qua lại giữa client và server 1 kênh kết nối TCP nhất định lại được khởi tạo giữa chúng. Dữ liệu được truyền đi qua kênh kết nối này. Khi tệp tin được truyền xong kênh này được ngắt.

Do các chức năng điều khiển và dữ liệu sử dụng các kênh khác nhau, nên mô hình điều khiển cũng chia phần mềm trên mỗi thiết bị ra làm 2 phần logic tương ứng với mỗi kênh. Thành phần **Protocol Interpreter** ( PI) là thành phần quản lý kênh điều khiển với chức năng phát và nhận lệnh. Thành phần **Data Tranfer Process** (DTP) là thành phần quản lý kênh truyền và nhận dữ liệu. Ngoài ra còn cung cấp tiến trình bên phía người dùng còn có thêm thành phần thứ 3 là giao diện người dùng FTP - thành phần này không có bên phía server. Dưới đây là mô hình kiến trúc xử lý của FTP.

* **Các tiến trình bên phía Server:**

**Server Protocol Interpreter** (Server - PI) : chịu trách nhiệm quản lý kênh điều khiển trên server. Nó lắng nghe yêu cầu kết nối tới từ user trên công dành riêng khi kết nối được thiết lập, Nó sẽ nhận lệnh từ User-PI, trả lời lại và quản lý tiến trình truyền dữ liệu trên server

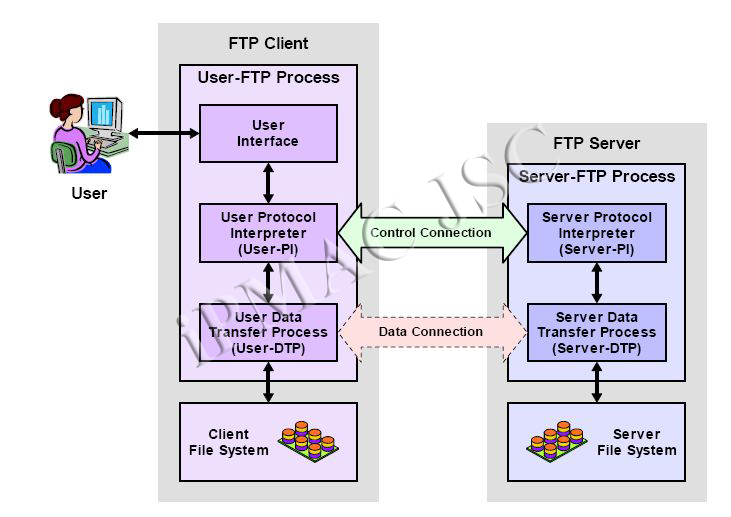
**Server DataTrasfer Process** (Server - DTP): làm nhiệm vụ gửi hoặc nhận file từ bộ phận User-DTP. Server-DTP vừa làm nhiệm vụ thiết lập kết nối kênh dữ liệu và lắng nghe một kết nối kênh dữ liệu từ user.

* **Các tiến trình bên phía Client:**

**User interface**: cung cấp giao diện xử lý cho người dùng. Nó cho phép người dùng điều khiển phiên FTP theo dõi các thông tin kết quả diễn ra trong tiến trình.

**User Protocol Interperter** (User-PI): Chịu trách nhiệm quản lý kênh điều khiển phía Client. Khới tạo kết nối FTP bằng việc phát ra yêu cầu tới phía Server – PI. Khi kết nối đã được thiết lập, nó xử lý các lệnh nhận được trên giao diện người dùng, gửi chúng tới Server-PI, và sau đó nhận phải hồi từ phía Server-PI. Nó cũng quản lý tiến trình User-DTP

**User Data Transfer Process** (User-DTP): làm nhiệm vụ gửi hoặc nhận dữ liệu từ Server-DTP. User-DTP có thể thiết lập hoặc lắng nghe yêu cầu kết nối kênh dữ liệu trên server. Nó tương tác với các thiết bị lưu trữ file phía client.



**Hình 2.0 Tiến trình trong FTP**

* Trình tự truy cập và chứng thực:

Bước 1: Người dùng cung cấp gửi Username từ User-PI → Server-PI bằng lệnh USER . Sau đó người dùng tiếp tục cung cấp Password cho phía Server-PI bằng lệnh PASS .

Bước 2: Server sẽ kiểm tra xem Username và Password có chính xác hay không . Nếu chính xác Server sẽ gửi trả 1 thông điệp rằng phiên làm việc đã được mở. Nếu không chính xác thì Server yêu cầu người dùng thực hiện lại việc chứng thực. Sau một số lần chứng thực sai thì Server sẽ chủ động ngắt kết nối.

Giả sử quá trình chứng thực đã thành công, server sau đó sẽ thiết lập kết nối để cho phép từng loại truy cập đối với người dùng được cấp quyền. Một số người dùng chỉ có thể truy cập vào một số file nhất định. Một số server có thể cấp quyền cho một số người dùng đọc và viết lên server, trong khi chỉ cho phép đọc đối với những người dùng khác. Người quản trị mạng có thể nhờ đó mà đáp ứng đúng các nhu cầu truy cập FTP.

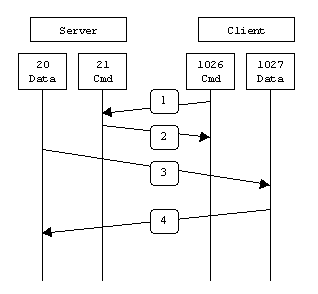
Một khi kết nối đã được thiết lập, server có thể thực hiện các lựa chọn tài nguyên dựa vào nhận diện người dùng.

### II.3. Quản lý kênh dữ liệu:

#### **II.3.1. Kết nối dữ liệu dạng chủ động:**

Trong dạng kết nối này phía User-PI sẽ gửi lệnh PORT (kèm với cổng chỉ định dành riêng cho việc truyền nhận file ) cho phía Server-PI .FTP Server sẽ sử dụng cổng 20 để kết nối đến FTP Client thông qua cổng port đã được chỉ định đó (lớn hơn 1024) . Vì lý do Server chủ động yêu cầu kết nối đến cổng port này bên phía client nên dữ liệu sẽ có thể bị chặn bởi tường lửa. (Nếu không có tường lửa thì việc vào ra bất hợp pháp một cách thoải mái trên mọi cổng của client sẽ rất dễ bị khai thác). Giả sử client châp nhận kết nối được khởi tạo (bỏ qua yếu tố tường lửa) , dữ liệu sẽ được truyền đi.

Sau đây là sơ đồ kết nối:



Hình 2.1: Sơ đồ kết nối dạng chủ động.

**Bước 1:** Client sử dụng 1 cổng ngẫu nhiên gọi là N (N>1024) dùng để khởi tạo kết nối đến port 21 của FTP Server có kèm theo giá trị PORT N+1 (data port . Giả sử cổng dùng để truyền nhận dữ liệu là N+1)

Ví dụ bên phía client N=1026

Port 21 của Server và Port 1026 của client là 2 port truyền câu lệnh ( command port) Port 1027 sẻ là ( data port)

**Bước 2:** Server gửi xác nhận ACK về cổng lệnh (command port ) của client (Ở đây là gửi về port 1026 của client)

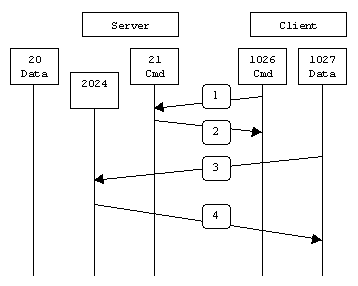
**Bước 3:** Server khởi tạo kết nối từ port 20 của mình đến cổng dữ liệu mà client đã khai báo trước đó (1027)

**Bước 4:** Client gửi ACK phản hồi cho Server

Khi FTP Server hoạt động ở chế độ active. Client không tạo kết nối thật sự vào cổng dữ liệu của FTP Server , mà chỉ đơn giản là thông báo cho Server biết rằng nó đang lắng nghe trên cổng nào và Server sẽ kết nối ngược về client cổng đó

#### **II.3.2. Kết nối dự liệu dạng bị động:**

Trong kiểu kết nối này client sẽ thông báo cho server bằng lênh PASV để server biết rằng client muốn dùng cơ chế bị động để truyền nhận tệp tin. Server sẽ gửi 1 thông báo tới cho client bao gồm địa chỉ và cổng port mà server sẽ thực hiện việc truyền nhận dữ liệu. Client sẽ sử dụng thông tin này để chủ động kết nối đến cổng mong muốn bên server. Vì client chủ động kết nối đến nên vần đề về tường lửa đã được khắc phục.



Hình 2.2: Sơ đồ kết nối dạng bị động.

**Bước 1 :** Client kết nối vào cổng lệnh của server phát lệnh PASV (lệnh mà client báo cho Server biết nó đang ở chế độ passive)

**Bước 2 :** Server trả lời bằng Port 2024 và cho client biết cổng 2024 đang mở để nhận kết nối dữ liệu

**Bước 3 :** Client tạo kết nối truyền dữ liệu từ cổng N+1 là 1027 đến cổng 2024 của Server

**Bước 4 :** Server xác nhận ACK cho cổng dữ liệu của Client

### II.4. Các kiểu dữ liệu:

Tệp tin được tao ra từ nhiều phần cứng và hệ điều hành khác nhau sẽ được chuyển đổi thành 4 kiểu dữ liệu trung gian để truyền đi và được chuyển đổi về kiểu dữ liệu cục bộ tại đích.

* ASCII A NVT-ASCII (Mã ASCII)
* EBCDIC E Văn bản EBCDIC
* IMAGE I Dòng các số nhị phân chưa được xử lý theo từng đoạn
* LOCAL L Các số nhị phân chưa được xử lý tính theo byte

Client gửi thông báo cho phía server biết kiểu dữ liệu sẽ được sử dụng

Kiểu dữ liệu mặc định là ASCII

### II.5. Các Command thường gặp:

* ABOR – ngưng lệnh đang thực hiện
* CWD – thay đổi thư mục làm việc
* DELE – xoá file
* LIST – xem danh sách files
* MDTM – trả về thông tin sửa đổi của files
* MKD – tạo thư mục mới
* PASS – gửi nội dung password
* NLST – danh sách tên các thư mục
* PASV – chế độ PASV
* PORT – mở một cổng dữ liệu
* PWD – hỏi thư mục đang thực hiện
* QUIT – tắt phiên kết nối
* RETR – nhận một file về local
* RMD – xoá một thư mục
* RNFR – đổi tên từ
* RNTO – đổi tên tới
* SITE – hỏi các command riêng của site
* SIZE – kích thước một file
* STOR – lưu trữ file lên server
* TYPE – kiểu truyền của dữ liệu (binary/ ascii)
* USER – gửi thông tin username

# CHƯƠNG II: PHÂN TÍCH THIẾT KỆ HỆ THỐNG

1. **PHÂN TÍCH YÊU CẦU**

### I.1. Yêu cầu chức năng:

* + Kết nối đến 1 FTP – Server
  + Lắng nghe các thông điệp từ Server trả về và hiển thị lên TextArea
  + Show danh sách các file/folder trên Server
  + Lắng nghe các sự kiện người dùng
    - Download file
    - Upload file
    - Tạo mới thư mục
    - Xóa 1 hoặc nhiều file
    - Xóa folder rỗng
    - Đổi tên file

### I.2 Yêu cầu về giao diện:

* Đăng nhập: Gồm các textvỉew username/ password/ Địa chỉ ftp/ Port/ Button Connection / Disconnect
* Server reply: Gồm 1 textArea
* Danh sách các file/folder: Gồm 1 table hiển thị danh sách các file / folder trên server
* Menu: Gồm có 4 lựa chọn: Download/ upload/ Delete/ Refresh/Tạo folder

1. **THIẾT KẾ HỆ THỐNG**

* Chương trình xây dựng là 1 FTP-Client và được kết nối đến 1 FTP-Server nào đó trên mạng
* FTP-Client sẽ được thiết kế gồm 3 tiến trình:
* **User Interface**: GuiMain.java, GuiNewFolder.java, GuiRenameFile.java, GuiUploadFile.java, GuiDownloadFile.java
* **User PI:** ControllerFtpClientPI.java
* **User transfer:** ControllerFtpClientTranfer.java

# CHƯƠNG III: TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

1. **TRIỂN KHAI**

**I.1. Môi trường phát triển**

Ứng dụng được phát triển trên

Eclipse IDE for Java Developers

Version: Neon.3 Release (4.6.3)

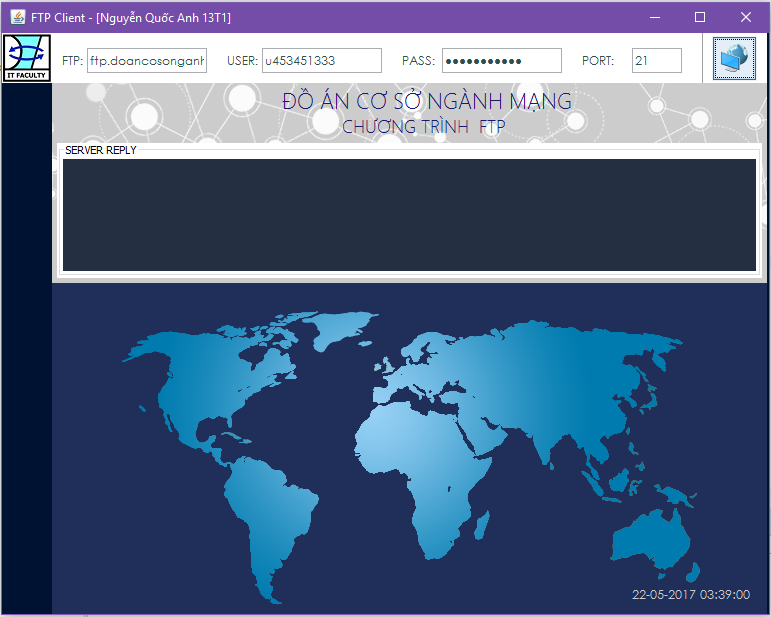
**I.2. Mã nguồn:** Được đính kèm trong đĩa CD

1. **KẾT QUẢ ĐẠT ĐƯỢC**

### II.1. Chức năng của chương trình

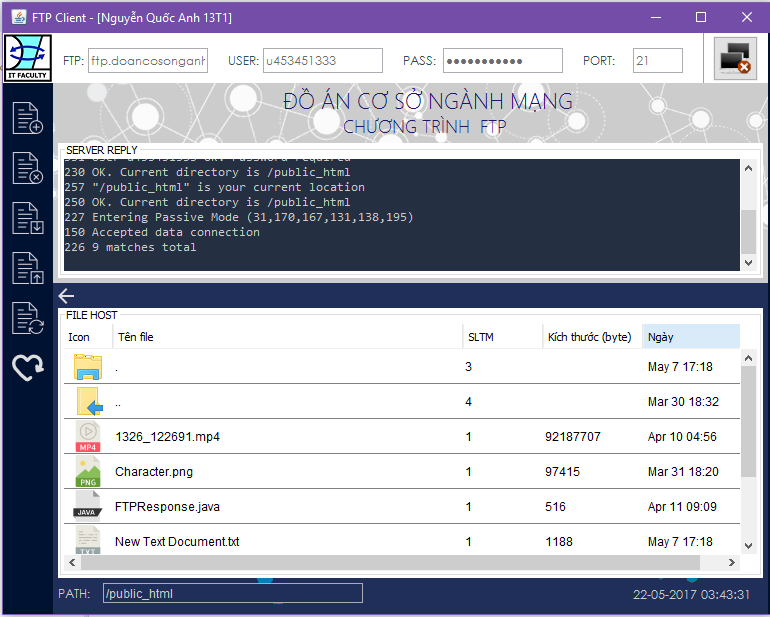
* Chương trình cho phép người dùng download / upload 1 file.
* Cho phép xóa thư mục. tệp tin
* Tạo mới 1 thưc mục

### II.2. Giao diện chính của chương trình



Hình 3.1: Giao diện chính của chương trình

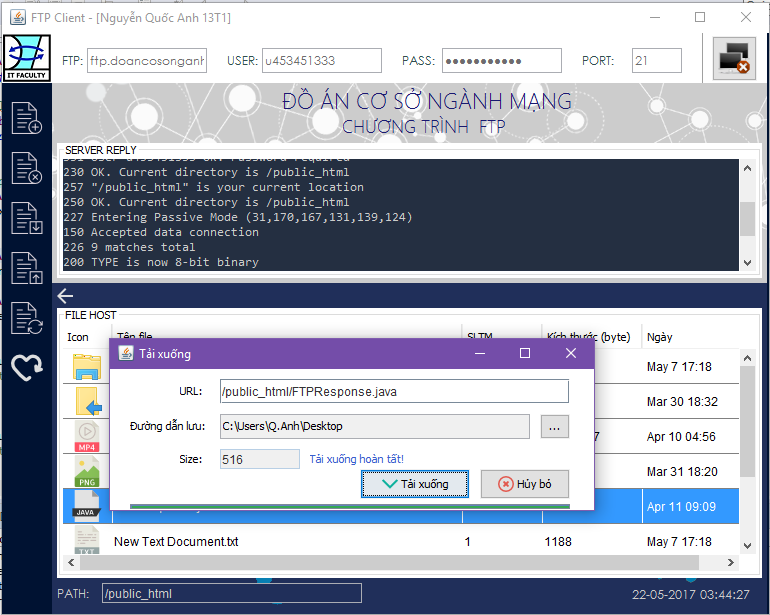
### II.3. Giao diện sau khi đăng nhập thành công



Hình 3.2: Chương trình sau khi đăng nhập thành công

* **Chức năng:**
* Nếu click đúp chuột vào 1 thư mục, thư mục sẽ được mở ra với tất cả các tệp tin và thư mục bên trong thư mục đó.
* Thư mục có tên (..) có nhiệm vụ điều hướng trở về thư mục trước đó
* Thư mục có tên (.) có nhiệm vụ điều hướng trở về thư mục root
* Button điều hướng back có nhiệm vụ điều hướng về thư mục trước đó (Button này được thêm vào do trong 1 số các FTP server sẽ không có các thư mục (..) hoặc (.) )
* Hiển thị thông điệp từ server: Khi nhận được 1 Response từ server, thông điệp này sẽ được hiển thị lên TextArea
* Nếu click chuột phải vào 1 tệp tin thì sẽ show ra menu lựa chọn bao gồm:
* Download 1 file
* Upload 1 file
* Tạo folder mới
* Xóa file/ folder (folder được xóa phải là 1 folder rỗng. Có thể xóa nhiều file cùng lúc).

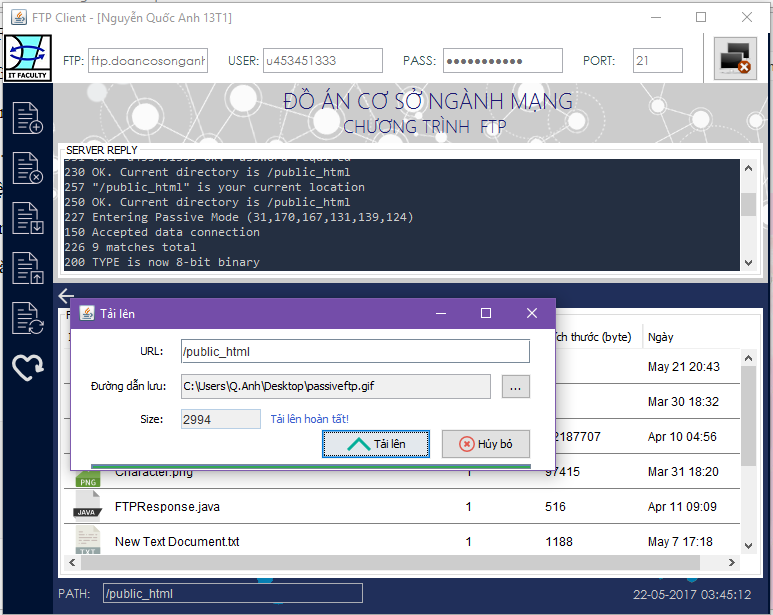
### II.4. Giao diện download file.



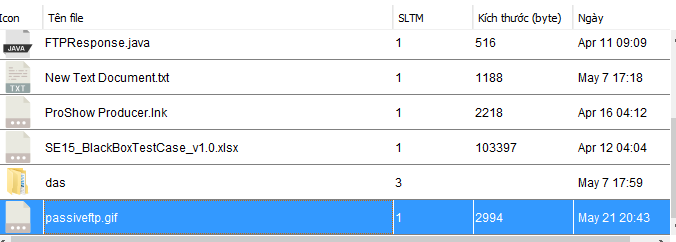
Hình 3.3: Download file

* Quy trình thực hiện của form:
* Bắt đầu tải về, Form sẽ hiển thị tiến độ tải
* Sau khi tải về hoàn tất sẽ xuất hiện thông báo :"Tải xuống hoàn tất! ” trên Form.
* Ấn nút “Hủy bỏ ” Đóng Form và Hủy bỏ quá trình tải file

### II.5. Giao diện upload file.



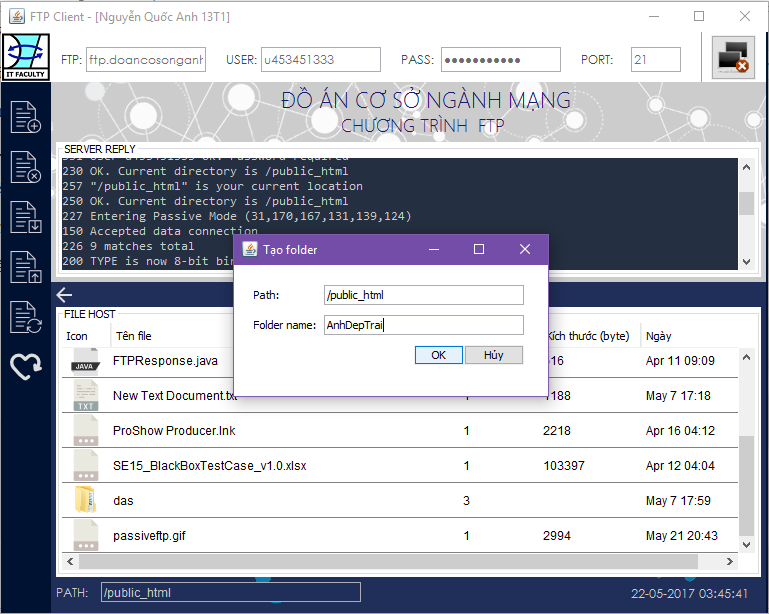
Hình 3.4: Upload file

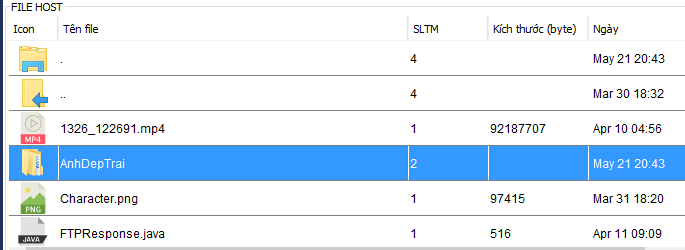


Hình 3.5: File sau khi upload lên Server

* Quy trình thực hiện của form:
* Bắt đầu tải lên, Form sẽ hiển thị tiến độ upload
* Sau khi tải lên hoàn tất sẽ xuất hiện thông báo :"Tải lên hoàn tất! ” trên Form. Đồng thời cập nhật lại dữ liệu trong bảng.
* Ấn nút “Hủy bỏ ” Đóng Form và Hủy bỏ quá trình tải file

### II.6. Giao diện tạo mới 1 folder





* Quy trình
* Right click vào 1 thư mục hoặc file bất kỳ để hiện thi menu → New Folder → Form Tạo Folder sẽ được hiển thị → đặt tên Folder muốn khởi tạo → click vào OK hoàn tất việc khởi tạo → Dữ liệu trong bảng sẽ được cập nhật lại

1. **ĐÁNH GIÁ:**

* Chương trình có giao diện thân thiên bắt mắt . dễ dàng sử dụng. Về cơ bản đã đáp ứng được 1 số chức năng của 1 chương trình FTP như download/ upload….
* Tuy nhiên vần còn nhiều bất cập như khả năng đáp ứng người dùng chưa thật sự tối ưu. Một số lệnh đang được tiến hành thực nghiệm của giao thức chưa được hiện thực trong chương trình. Chưa nghiên cứu vào bảo mật thông tin khi dùng giao thức FTP.

1. **HƯỚNG PHÁT TRIỂN:**

* Download/Upload cùng lúc nhiều file.
* Download/Upload thư mục
* Tích hợp kéo thả đối tượng từ client vào chương trình FTP….

*CD kèm theo báo cáo, đính kèm hướng dẫn triển khai.*

*-= HẾT =-*