

BÁO CÁO ĐỒ ÁN FTP

Github

THÀNH VIÊN

Tên	MSSV	Github
Hoàng Dân An	1612001	beohoang98
Nguyễn Phước An	1612009	phuocantd

PHÂN CHIA CÔNG VIỆC

STT	CÔNG VIỆC	Người đảm nhiệm
1	Thiết kế flow chương trình chính	Dân An
2	Thiết kế OOP cho chương trình	Dân An
3	Tạo các hàm xử lý giao thức cơ bản	Dân An
4	Phân chia thư mục và tạo project	Phước An
5	Viết các hàm cd, lcd, pwd, mdelete, mkdir, rmdir, quit, exit	Phước An
6	Viết hàm xử lý mở port, active, passive	Dân An
7	Viết các hàm put, mput	Phước An
8	Viết các hàm get, mget	Dân An
9	Viết câu lệnh help và description cho các câu lệnh ftp	Cả hai
10	Quản lý git cho mã nguồn	Dân An
11	Phụ trách document	Cả hai
12	Phụ trách ngoại giao (<i>copy ý tưởng team khác</i>)	Phước An
13	Test lỗi và tìm bug	Phước An

ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH

STT	Chức năng	Hoàn thành
1	Kết nối và login đến server	1
2	Xử lý đầy đủ các câu lệnh	1
3	Câu lệnh ls, dir	1
4	Câu lệnh cd, lcd, pwd, lpwd	1
5	put	0
6	mput	0
7	get	1
8	mget	1
9	delete	1
10	mdelete	0
11	mkdir	1
12	rmdir	1
13	Cơ chế passive và active	1

Tổng 69%

BÁO CÁO CÁC HÀM VÀ CẤU TRÚC CHƯƠNG TRÌNH

MỤC LỤC

1. CẤU TRÚC CHƯƠNG TRÌNH
 2. DANH SÁCH LỆNH
 3. CÁC HẲNG SỐ
 4. CẤU TRÚC CLASS EPTIPI
 5. SCREENSHOTS VÀ THEO DÕI GÓI TIN BẰNG WIRESHARK
-

1. CẤU TRÚC CHƯƠNG TRÌNH

1. Khởi tạo kết nối server với địa chỉ từ arguments, hoặc từ lệnh open
2. Gọi hàm login để người dùng login vào, lặp lại đến khi login thành công
3. Đọc lệnh từ bàn phím của người dùng
4. Xử lý lệnh người dùng và gọi hàm đã viết
5. Lặp về Bước 3

2. DANH SÁCH LỆNH

Các câu lệnh của chương trình (dir, ls, ...) được lưu trong một `std::map listCmd` kèm với `struct cmdDescription` là các thông tin chi tiết về lệnh đó

- Key của `listCmd` là kiểu chuỗi, là câu lệnh.
- Value của `listCmd` là một struct gồm syntax câu lệnh (title) và thông tin chi tiết (description)

Các lệnh bao gồm

- open
- ls hoặc ls [path]
- dir hoặc dir [path]
- cd [path]
- pwd
- lcd [path]
- ll
- ldir
- get [path]
- mget [expression]
- put [path]
- mput [expression]

- `del [path]`
- `mdel [expression]`
- `mkdir [name]`
- `rmdir [name]`
- `help [cmd name]` hoặc `help`
- `bye`
- `disconnect`
- `quit`

Chi tiết hơn tại [đây](#)

3. CÁC HẲNG SỐ

1. **Các status code** Các status code được lưu trong enum `FTPCode`

VD: `LOGIN_SUCCESS = 230`, `OPEN_DATA_CONNECT = 150`, ...

2. **Hằng số trạng thái mở port** Bao gồm active và passive

Là các `const int` được define bên trong **namespace** `FTPDataMode`

- `FTPDataMode::PASSIVE = 0`
- `FTPDataMode::ACTIVE = 1`
- `FTPDataMode::DEFAULT = PASSIVE` (Mặc định là passive)

3. **Hằng số trạng thái truyền dữ liệu** Bao gồm mode ASCII (truyền những file text) và mode BINARY (truyền những file nhị phân)

Định nghĩa trong **namespace** `FTPFileMode`

- `FTPFileMode::ASCII` và `FTPFileMode::BINARY`
 - Mặc định `FTPFileMode::DEFAULT = FTPFileMode::BINARY`
-

4. CẤU TRÚC CLASS EPTIPI

Dùng để xử lý các câu lệnh và các giao thức ftp nhanh hơn

Chứa thông tin socket kết nối đến server

Xử lý các hàm nhận vào từ người dùng ('ls', 'dir', 'get', ...)

Chi tiết về class Eptipi

BIẾN DỮ LIỆU

std::wstring server_addr

Lưu địa chỉ server dưới dạng IP

std::string client_addr

Lưu địa chỉ client để sử dụng cho mode Active

std::string returnStr

Lưu chuỗi trả về từ server

int returnCode

Lưu return code trả về từ server Mặc định là -1

int returnPort

Lưu port trả về từ server (nếu có) Mặc định là -1

UCHAR dataMode

Lưu trạng thái mở data connect là passive hay active Mặc định là **FTPDataMode::PASSIVE**

UCHAR fileMode

Lưu trạng thái truyền dữ liệu là Binary hay ASCII Mặc định là **FTPFileMode::BINARY**

CÁC HÀM XỬ LÝ

bool Eptipi::handleCmd(std::string cmd)

Xử lý chuỗi lệnh cmd truyền vào

void Eptipi::connect(const wchar_t *)

Kết nối đến chuỗi địa chỉ server truyền vào, nếu không throw exception ra

bool Eptipi::login()

Tạo prompt đăng nhập trên màn hình console

Nếu đăng nhập không thành công, hàm trả về **false**

CÁC HÀM NỀN

void sendCmd(std::string)

Gửi lệnh thuần ftp lên server

VD:

```
sendCmd("LIST\r\n");
```

void receiveStatus()

Nhận về 1 return status từ server, sau đó cắt các thông tin cần thiết lưu về các biến

Sử dụng sau khi thực hiện lệnh sendCmd()

VD:

- Server trả về 220 Welcome ...
 - Hàm cắt ra 220 và lưu vào returnCode
 - Chuỗi trả về lưu vào returnStr
- Server trả về 227 Passive mode (15,10,19,97,69,69)
 - Chuỗi trả về sẽ lưu vào returnStr
 - Status code 227 sẽ lưu vào returnCode
 - Port server trả về bao gồm 15.10.19.97 lưu vào server_addr và $69*256 + 69 = 17733$ sẽ lưu vào returnPort

CSocket * openPassiveAndConnect()

Mở data connection đến server theo mode Passive

Trả về CSocket đã kết nối

CSocket * openActiveAndConnect()

Mở data connection đến server theo mode Active

Trả về CSocket đã Listen, đợi Accept sau.

void openDataPort(bool (*before)(CallbackParam &cb), bool (*after)(CallbackParam &cb), CallbackParam &cb)

Thực hiện công việc có sử dụng đến data connection

Param:

- `CallbackParam` &cb : chứa thông tin cmdConn, dataConn (data connection mới mở), ... cho hàm before và after
- `bool (*before)(CallbackParam &)` : tham số hàm trả về bool, nhận tham số `CallbackParam`
- `void (*after)(CallbackParam &)` : tham số hàm nhận vào `CallbackParam`

Hàm sẽ:

- Mở port (active hay passive)
- Thực hiện hàm **before**
- Nếu hàm before lỗi, return
- Nếu ok, hàm thực hiện kết nối đến data connection của server
- Trả thông tin data connection vào `CallbackParam` và gọi hàm **after**

Hàm được viết để việc viết các hàm liên quan đến kết nối data connection nhanh gọn hơn

VD: Khi xử lý lệnh dir sẽ là

```
CallbackParam cb;
cb.main = this;

openDataPort([](CallbackParam &cb){
    cb.mainFTP->cmdCon.sendCmd("LIST\r\n");
    cb.mainFTP->cmdCon.receiveStatus();
}, [](CallbackParam &cb){
    if (cb.dataCon == NULL) return;

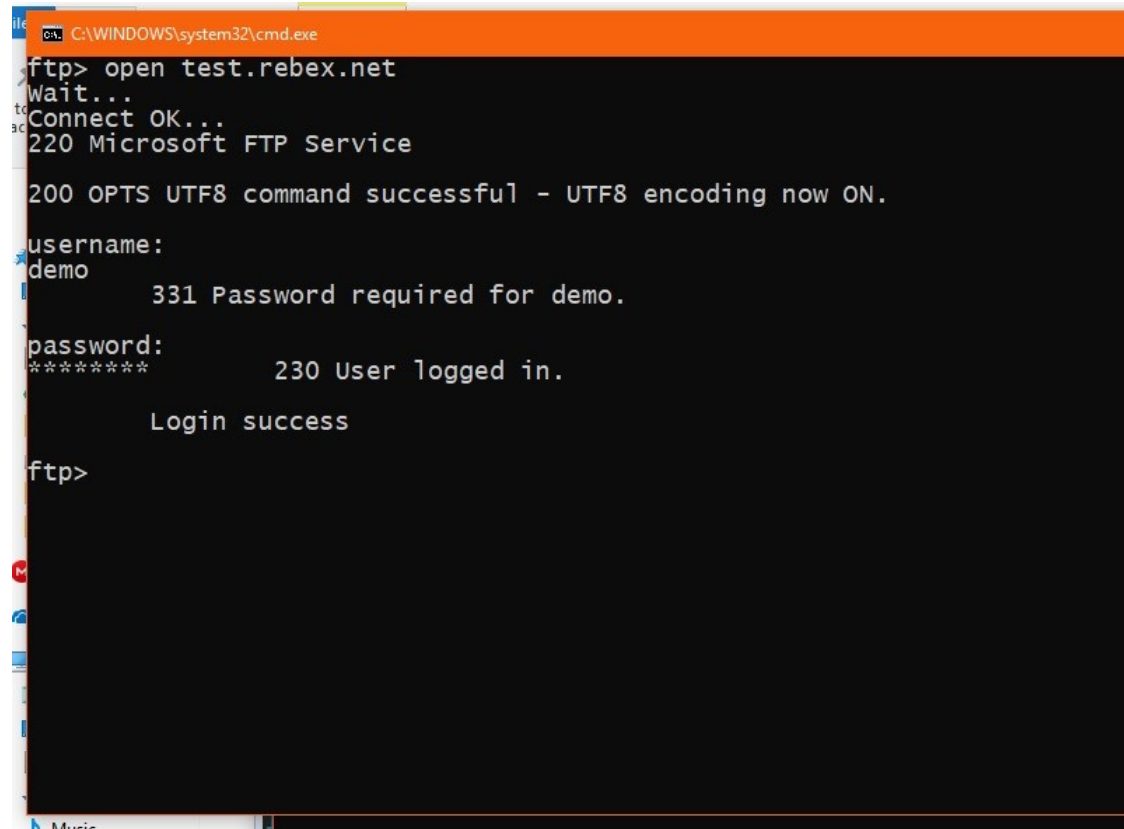
    //nhận data thông qua dataCon
    // cb.dataCon->Receive(...)
}, cb);
```

Cấu trúc dữ liệu hỗ trợ

```
struct CallbackParam
{
    Eptipi * mainFTP;
    CSocket * dataCon;
    std::string path;
    UINT64 filesize;
}
```

5. SCREENSHOTS VÀ BẮT GÓI TIN BẰNG WIRESHARK

- open test.rebex.net

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd.exe'. The command prompt displays an FTP session. The user enters 'ftp> open test.rebex.net'. The output shows 'Wait...', 'Connect OK...', '220 Microsoft FTP Service', and '200 OPTS UTF8 command successful - UTF8 encoding now ON.'. The user then enters 'username: demo', and the output is '331 Password required for demo.'. Next, the user enters 'password: *****', and the output is '230 User logged in.'. Finally, the output shows 'Login success' and the prompt returns to 'ftp>'.

```
C:\WINDOWS\system32\cmd.exe
ftp> open test.rebex.net
Wait...
Connect OK...
220 Microsoft FTP Service

200 OPTS UTF8 command successful - UTF8 encoding now ON.

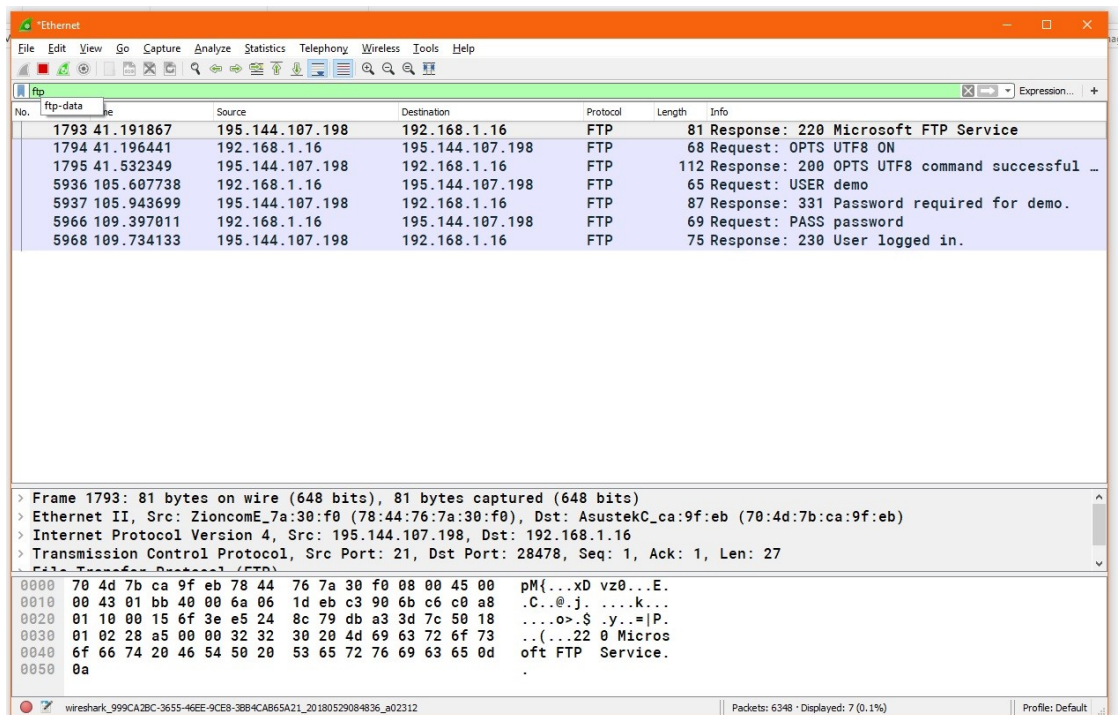
username:
demo      331 Password required for demo.

password:
*****   230 User logged in.

Login success

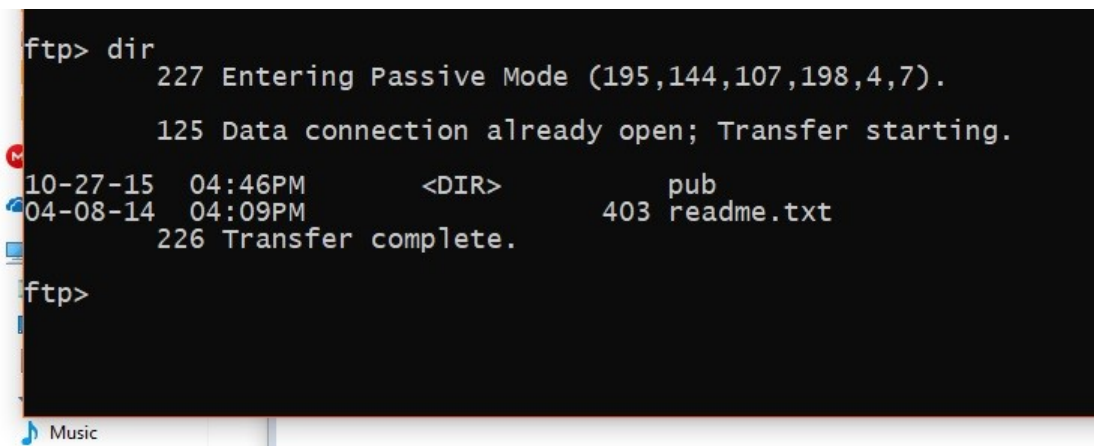
ftp>
```

connect server cli



connect server ws

- dir



dir in console

5937	105.943699	195.144.107.198	192.168.1.16	FTP	87 Response: 331 Password required for demo.
5966	109.397011	192.168.1.16	195.144.107.198	FTP	69 Request: PASS password
5968	109.734133	195.144.107.198	192.168.1.16	FTP	75 Response: 230 User logged in.
6457	210.625412	192.168.1.16	195.144.107.198	FTP	62 Request: TYPE A
6458	210.961528	195.144.107.198	192.168.1.16	FTP	74 Response: 200 Type set to A.
6459	210.962589	192.168.1.16	195.144.107.198	FTP	60 Request: PASV
6460	211.299491	195.144.107.198	192.168.1.16	FTP	104 Response: 227 Entering Passive Mode (195,14...
6465	211.643785	192.168.1.16	195.144.107.198	FTP	61 Request: LIST
6468	211.980662	195.144.107.198	192.168.1.16	FTP	108 Response: 125 Data connection already open;...
6469	211.980662	195.144.107.198	192.168.1.16	FTP	78 Response: 226 Transfer complete.

> Frame 1793: 81 bytes on wire (648 bits), 81 bytes captured (648 bits)					
> Ethernet II, Src: ZioncomE_7a:30:f0 (78:44:76:7a:30:f0), Dst: AsustekC_ca:9f:eb (70:d4:7b:ca:9f:eb)					
> Internet Protocol Version 4, Src: 195.144.107.198, Dst: 192.168.1.16					
> Transmission Control Protocol, Src Port: 21, Dst Port: 28478, Seq: 1, Ack: 1, Len: 27					

0000	70 4d 7b ca 9f eb 78 44	76 7a 30 f0 08 00 45 00	pM(...xD vz0...E.
0010	00 43 01 bb 40 00 6a 06	1d eb c3 90 6b c6 c0 a8	.C..@.j.k...
0020	01 10 00 15 6f 3e e5 24	8c 79 db a3 3d 7c 50 18	...o>.\$.y..= P.
0030	01 02 28 a5 00 00 32 32	30 20 4d 69 63 72 6f 73	..(...22 0 Micros
0040	6f 66 74 20 46 54 50 20	53 65 72 76 69 63 65 0d	oft FTP Service.
0050	0a		.

wireshark_999CA2BC-3655-46EE-9CE8-3BB4CAB65A21_20180529084836_a02312

Packets: 6664 • Displayed: 14 (0.2%)

Profile: Default

dir show in wireshark

- cd pub/example

```
ftp> dir
227 Entering Passive Mode (195
125 Data connection already op
10-27-15 04:46PM <DIR>
04-08-14 04:09PM 403
226 Transfer complete.

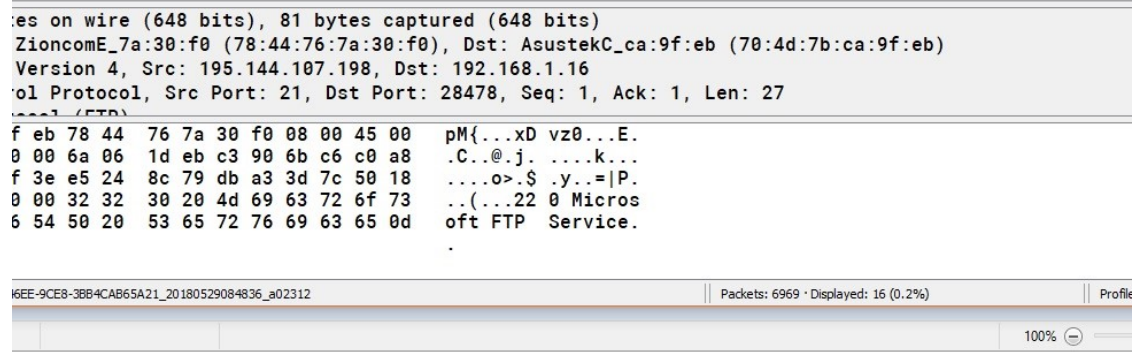
ftp> cd pub/example
250 CWD command successful.

ftp> _
```

Downloads

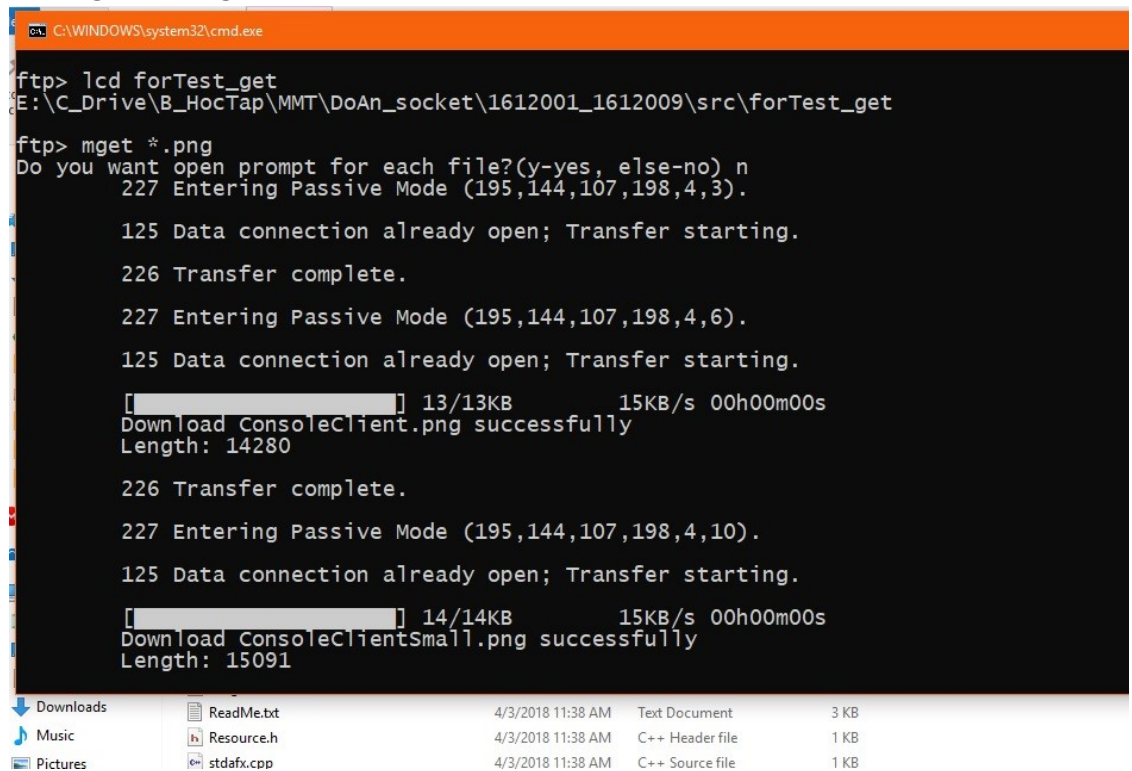
cd in console

195.144.107.198	192.168.1.16	FTP	108 Response: 125 Data connection already open; Transfer starting.
195.144.107.198	192.168.1.16	FTP	78 Response: 226 Transfer complete.
192.168.1.16	195.144.107.198	FTP	71 Request: CWD pub/example
195.144.107.198	192.168.1.16	FTP	83 Response: 250 CWD command successful.

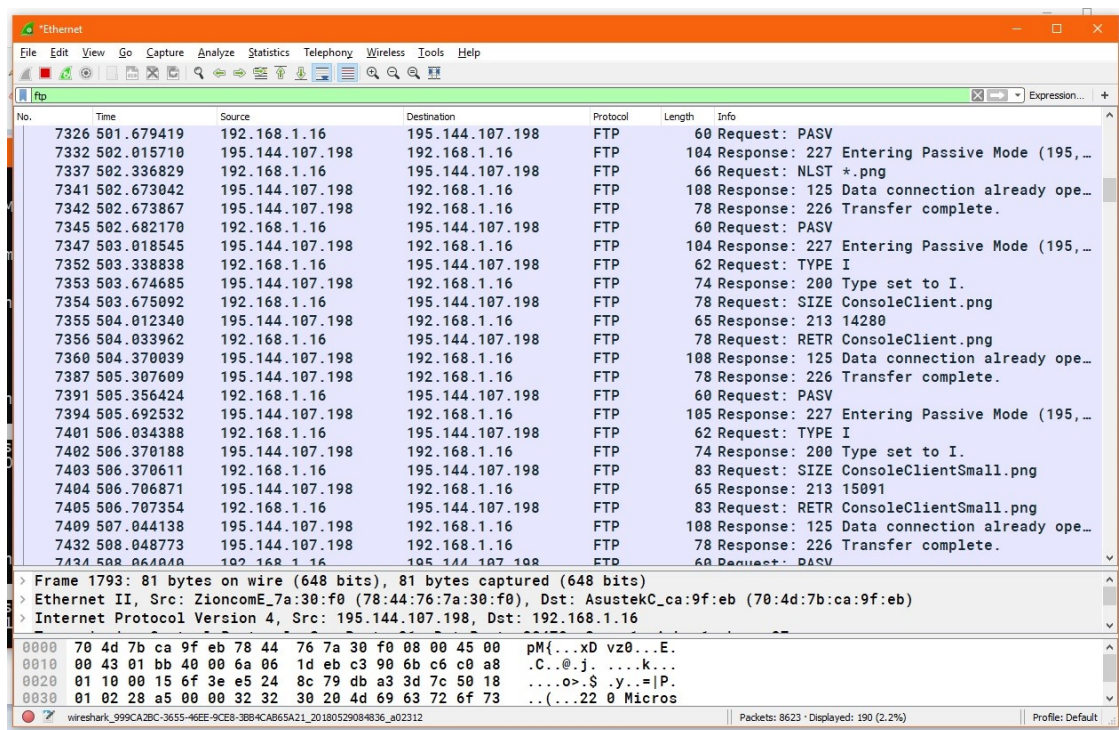


cd show in wireshark

- mget *.png

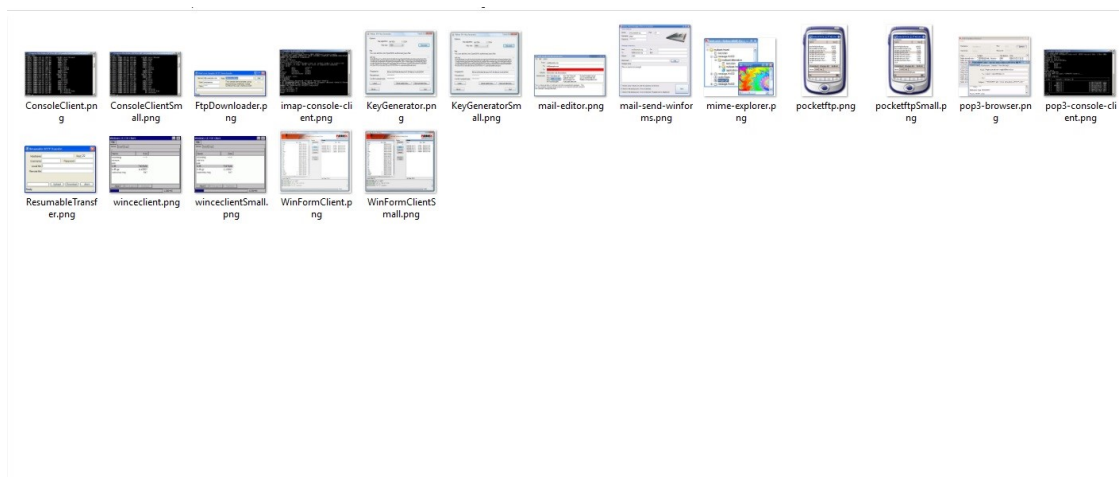


mget in console



mget show in wireshark

• Result



Kết quả

TÀI LIỆU THAM KHẢO

- [FTP Command](#)
- [FTP Return codes](#)
- [How to put file by raw ftp command](#)
- [Passive and Active in ftp](#)