**BÁO CÁO ĐỒ ÁN SOCKET**

**XÂY DỰNG CHƯƠNG TRÌNH FTP CLIENT**

**--**

1. PHÂN CÔNG CÔNG VIỆC

Thành phần nhóm:

Phạm Minh Chiến 1612052

Lê Hoàng Sang 1612554

Đoàn Quang Tuấn 1612780

|  |  |
| --- | --- |
| **Tên công việc** | **Người người thực hiện** |
| Vẽ UML |  |
| Thiết kế lớp |  |
| Login() |  |
| Connect() |  |
| Send() |  |
| Receive() |  |
| displayMessage() |  |
| Action() |  |
| LS, DIR |  |
| PWD |  |
| CD |  |
| LCD |  |
| GET |  |
| PUT |  |
| MGET |  |
| MPUT |  |
| DEL |  |
| MDEL |  |
| MKDIR |  |
| RMDIR |  |
| PASV |  |
| QUIT |  |
| UESR |  |
| PASS |  |
| CLEAR |  |
| HELP |  |
| standardizedCMD |  |
| defineOrder() |  |
| getClauses() |  |
| getCmd() |  |
| getDataPort() |  |
| getServerCode() |  |
| openPort() |  |
| openPassiveConnect() |  |
| openActiveConnect() |  |

1. NHỮNG CHỨC NĂNG CHÍNH VÀ CẤU TRÚC TỔ CHỨC
   1. Chức năng chương trình: hỗ trợ đầy đủ các tính năng của một FTP-Client, cả chế độ Active và Passive (mặc định chương trình sẽ ở chế độ active):

* Login : đăng nhập vào chương trình
* Open : mở kết nối tới server
* ls, dir : liệt kê các tập tin và thư mục trền server
* pwd : hiện đường dẫn hiện tại trên server
* cd : thay đổi đường dẫn trên server
* lcd : thay đổi dường dẫn dưới client
* get : tải xuống một file về từ server
* mget : tải xuống nhiều file về từ server
* put : tải một file lên server
* mput : tải nhiều file lên server
* del : xóa tập tin, thư mục trên server
* mdel : xóa nhiều tập tin, thư mục trên server
* mkdir : tạo thư mục mới
* rmdir : xóa thư mục
* pasv : chuyển qua trạng thái passive
* clear : xóa màn hình
* help : hiện thông báo trợ giúp
  1. Cấu trúc chương trình
     1. Tổ chức chương trình thành một **lớp FTPClien**t chứa:
* cmdClient: CSocket - là command Client để gửi lệnh lên server.
* user: string – lưu tên người dùng.
* password: string – lưu mật khẩu người dùng.
* Request: string – chứa lệnh cần gửi lên server.
* Respone: string – nới chứa phản hồi của server.
* Mode: bool – cờ hiệu chế độ hiện tại 0 là passive, 1 là active.
* isLogined: bool – cờ kiểm tra đăng nhập thành công.
* isConnected: bool – cờ kiểm tra kết nối thành công tới server.
* Argument: vector<string> - chứa các tham số của lệnh cmd.
* dataClient được lấy từ hàm openPort tùy thuộc vào chế độ passive hay active mà phát sinh một dataClient tương ứng.
  + 1. Cung cấp các hàm:  
       Chỉ public một hàm duy nhất là hàm getCmd() nhận vào một chuỗi và xữ lí chuỗi đó bằng những hàm chức năng tương ứng.
    2. Bảng mô tả các hàm chức năng:

Các tham số truyền vào chủ yếu là tên file sẽ được push vào vector argument nên class chỉ vào kiểm tra argument có những tham số nào và pop ra để sử dụng nên hàm chức năng sẽ không có tham số truyền vào.

|  |  |
| --- | --- |
| **Tên hàm** | **Chức năng** |
| void getCmd(); | Thực hiện một vòng lặp vô tận cho tới khi nhận được lệnh Quit. Đầu tiên yêu cầu đăng nhập, kiểm tra Login và nhận vào chuỗi người dùng nhập, gọi tiếp đến hàm defineOrder() để xác định hàm nào sẽ được gọi (hoặc báo không hợp lệ). Gọi tiếp hàm getClauses() để push các tham số (chủ yếu là tên files/folders) vào vector argument. Sau khi có được order của lệnh sẽ gọi đến lệnh đó. |
| void getClauses(string cmd); | Nhận vào một chuỗi và push các thành phần của chuỗi (tên file, tên thư mục) vào vector argument. |
| int defineOrder(string cmd); | Trả về một số nguyên xác định vị trí của chuỗi cmd truyền vào, nếu chuỗi cmd không hợp lệ trả về -1. |
| String standardizedCMD(string cmd) | Hàm này sẽ chuẩn hóa chuỗi cmd người dùng nhập vào như cắt bỏ khoảng trắng dư thừa, chuyển hết chữ cái về dạng lowercase. Trả về chuỗi đã chuẩn hóa. |
| int getDataPort(); | Hàm này xử lí chuỗi this->respone mà server gửi về có dạng “….(ipadress, a, b)” hàm sẽ tách a, b ra và trả về giá trị a\*256 + b. Nếu chuỗi trong this->respone không đúng cấu trúc, hàm trả về 0. |
| int getServerCode(); | Hàm xử lí chuỗi this->respone được nhận cmdClient nhận về từ server có dạng “xxx Chuỗi”. Hàm trả về giá trị xxx. |
| CSocket\* openPort(); | Hàm gọi đến 1 trong hai hàm openPasssiveConnect() hoặc OpenActiveConnect() tùy thuộc vào cờ this->mode là 0 hay 1. |
| CSocket\* openPassiveConnect(); | Tạo một CSocket dataClient mới, cmdClient gửi lện pasv lên server. dataClient sẽ kết nối đến địa chỉ this->hostIP tại port có giá trị bằng this->getDataPort(). Hàm trả về địa chỉ của dataClient vừa tạo. |
| CSocket\* openActiveConnect(); | Tạo một CSocket dataClient mới. cmdClient lấy port của dataClient và gửi lên server để báo server là dataClient đang listen. Hàm trả về địa chỉ của dataClient vừa tạo. |
| bool connect(); | Kiểm tra xem hiện tại có đang kết nối vào server nào hay không. |
| bool login(); | Kiểm tra xem người dùng đã login vào server hay chưa. |
| void cmd\_user(); | Gửi lệnh USER lên server và nhận phản hồi. |
| void cmd\_pass(); | Gửi lệnh PASS lên server và nhận phản hồi. |
| void send(); | Gửi lệnh được lưu trong this->request lên server. |
| int receive(); | Nhận thông điệp từ server vào gán vào this->respone. |
| void displayMessage(); | Xử lí thông điệp trong respone và xuất ra màn hình. |
| void action(); | Gọi liên tiếp hàm this->send(); this->receive(); this->displayMessage(); |
| void cmd\_ls(); | Mở port, cmdClient gửi lệnh NLST lên server và dataClient nhận dữ liệu từ server và xuất ra màn hình. |
| void cmd\_pwd(); | cmdClinet gửi lệnh PWD lên server và nhận phản hồi. |
| void cmd\_cd(); | cmdClinet gửi lệnh CWD lên server và nhận phản hồi. |
| void cmd\_lcd(); | Thay đổi địa chỉ hiện tại của FTP-Client bằng hàm setCurrentDirectory() của window. |
| void cmd\_get\_core(string file); | * Mở sẵn một file mới có tên “file” để ghi tại thư mục hiện hành của FTP Client. * cmdClient gửi lệnh RETR + file lên server và chờ phản hồi, nếu phản hồi thành công sẽ mở port. * dataClient sẽ nhận dữ liệu từ server (dạng stream) và ghi vào file đã mở sẵn. |
| void cmd\_get(); | Gọi lại hàm cmd\_get\_put(argument[0]). |
| void cmd\_mget(); | Gọi lại hàm cmd\_get với tham số truyền vào là lần lượt các giá trị có trong vector argument. |
| bool cmd\_put\_core(string file); | * Mở sẵn một file có tên “file” để đọc nhị phân tại thư mục hiện hành. * cmdClient gửi lệnh STOR + file lên server và chờ phản hồi, nếu phản hồi thành công sẽ mở port. * dataClient sẽ đọc cho tới hết dữ liệu trong file đã mở và truyền lên server bằng lệnh send(). |
| void cmd\_put(); | Gọi lại hàm cmd\_put\_core với tham số là this->argument[0]. |
| void cmd\_mput(); | Gọi lại hàm cmd\_put với tham số là lần lượt các giá trị có trong vector argument. |
| void cmd\_del(); | Gửi lệnh DELE + tham số this->argument[0] lên server và chờ phản hồi. |
| void cmd\_mdel(); | Gọi lại n lần gàm cmd\_del với mỗi lần là từng giá trị trong vector argument. |
| void cmd\_mkdir(); | Gửi lên server lệnh MKD + chuỗi this->argument[0] và chờ phản hồi. |
| void cmd\_rmdir(); | Gửi lên server lệnh MKD + chuỗi this->argument[0] và chờ phản hồi. |
| void cmd\_pasv(); | Gửi lệnh pasv lên server nếu là IPv4 hoặc epsv nếu là IPv6 và chờ phản hồi. |
| void cmd\_quit(); | Gửi lệnh quit lên server và chờ phản hồi. Sau đó đóng chương trình. |
| void cmd\_help(); | Xuất ra tất cả lệnh mà chương trình hỗ trợ. |
| void cmd\_clear(); | Xóa màn hình hiện tại. |

* 1. Mô tả các hàm:

Các hàm chức năng trong chương trình chia làm hai loại:

* Loại 1: các hàm command: del, mdel, pwd, cd, lcd, quit, help, user, pass.

CSocket cmdClient chỉ việc gửi lệnh lên và nhận về phản hồi.

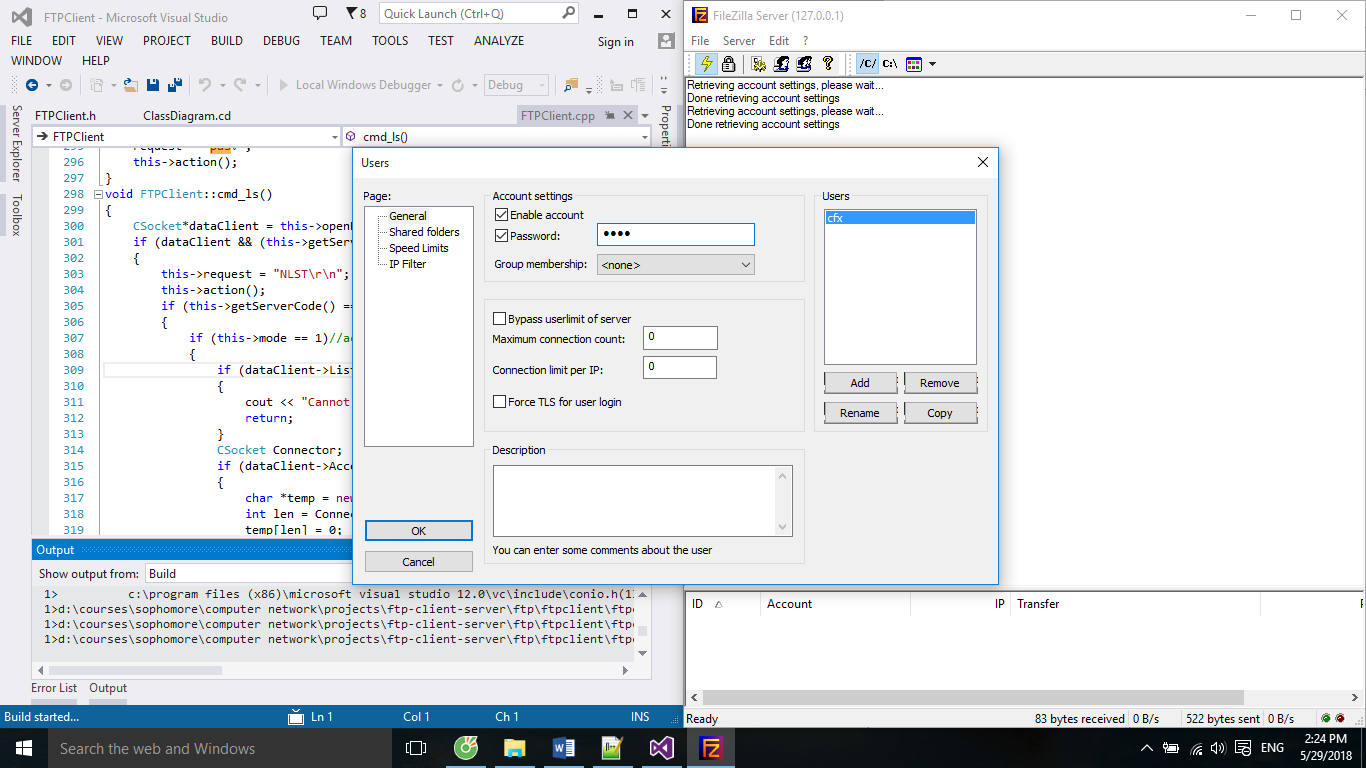
* Loại 2: các hàm cần truyền tải dữ liệu: ls, get, put

Sau khi cmdClient gửi command lên server, nếu phản hồi thành công thì FPTCLient sẽ thực hiện mở port, tạo dataClient để nhận dữ liệu từ server, hai dạng passive và active có đôi chút khác nhau ở chỗ port và lúc chuẩn bị truyền tải dữ liệu, còn kĩ thuật truyền/nhận thì giống nhau hoàn toàn. Passive thì sau khi mở port sẽ dùng ngay dataClient truyền tải ngay (vì đã kết nối với server rồi). Còn Acitve thì chưa, ở chế độ này dataClient chỉ mới có thể listen kết nối của server và không thể kết nối với server được, do đó cần có một CSocket Connector khác giữ kết nối với server mỗi khi dataClient accept thành công với server, Connector này mới là đối tượng trực tiếp truyền/nhận dữ liệu với server. Và một chú ý nữa là mỗi lần gọi ls, get, put ta phải mở một port mới vì port cũ sau khi dùng xong phía server sẽ đóng kết nối.

* Hàm CSocket\* openPassiveConnect():  
  Như đã nêu bên trên, hàm này sẽ mở port ở chế độ passive. Đầu tiên ta tạo một CSocket mới có tên là dataClient với port do window xác định. Sau đó cmdClient sẽ gửi một lênh PASV lên server để thông báo rằng chương trình muốn mở một kết nối passive với server, ta cần một thông điệp của server có nội dung “xxx stringyyy (ip address, a, b). Khi nhận được thông điệp này có nghĩa là server đã cấp cho ta một port dữ liệu, nhiệm vụ của ta là cho dataClient kết nối vào port có giá trị (a\*256+b) là xong.
* Hàm openActiveConnect():  
  Ở chế độ Active ta mong muốn rằng server sẽ chủ động kết nối với ta. Tuy nhiên muốn vậy ta phải thông báo cho phía server biết ta đang chờ một kết nối từ server bằng lệnh command port “PORT ip, a, b” với ip là địa chỉ ip của Socket đang chờ server kết nối tới, a = port/256 và b = port %256, đó là quy tắc. Tóm lại ta sẽ tạo một dataClient với port do window xác định, dùng cmdClient gửi “thông tin” của dataClient cho server biết mà liên lạc với dataClient.

Khác với Passive, chế độ Active này dataClient chỉ có chức năng listen một kết nối từ server chứ tuyệt nhiên không thể truyền nhận trực tiếp với server bởi không thể connect tới server. Cho nên, mỗi lần truyền gửi (ls, get, put) ta sẽ tạo thêm một Socket Connector để giữ kết nối với server, sau lệnh accept(Connector) nếu thành công thì Connector giờ sẽ là đối tượng truyền dữ liệu trực tiếp với server, còn dataClient sẽ tiếp tục listen().

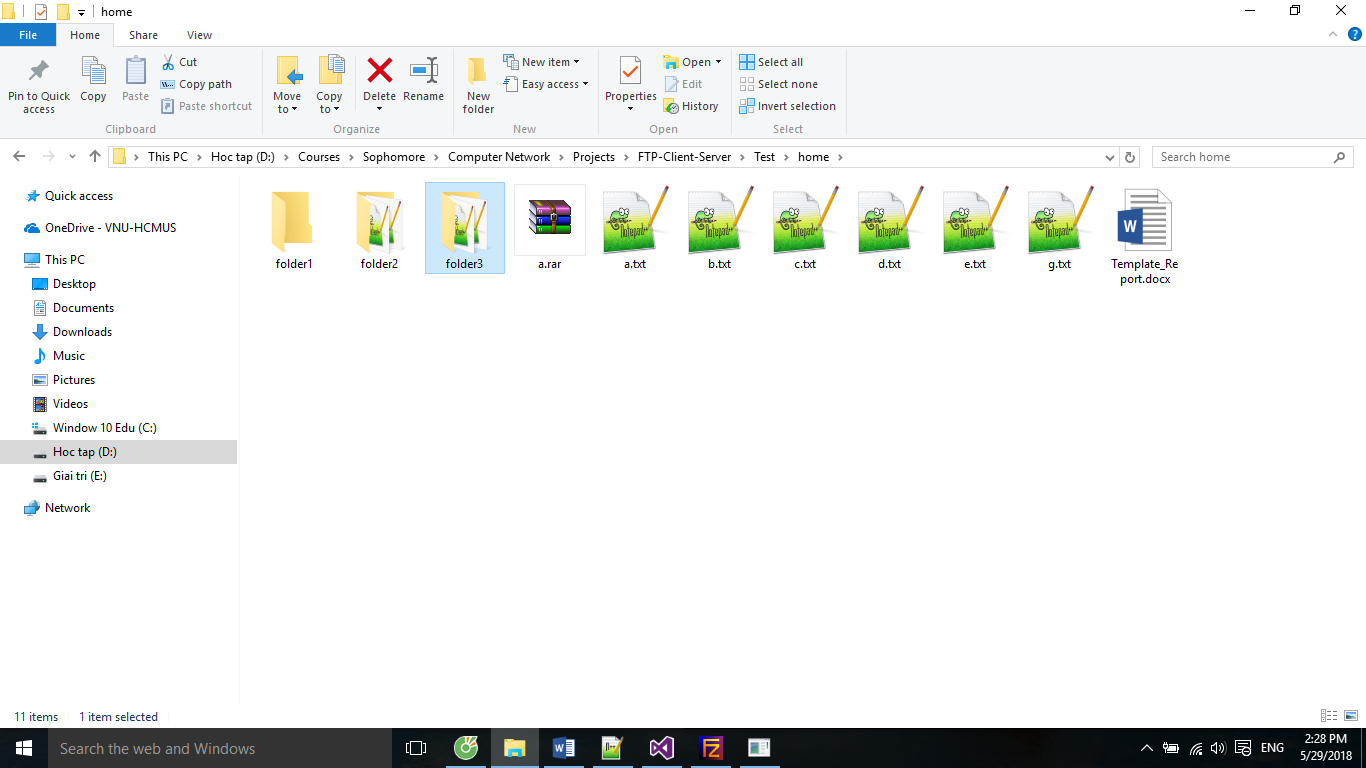
1. CHẠY THỬ VÀ KẾT QUẢ

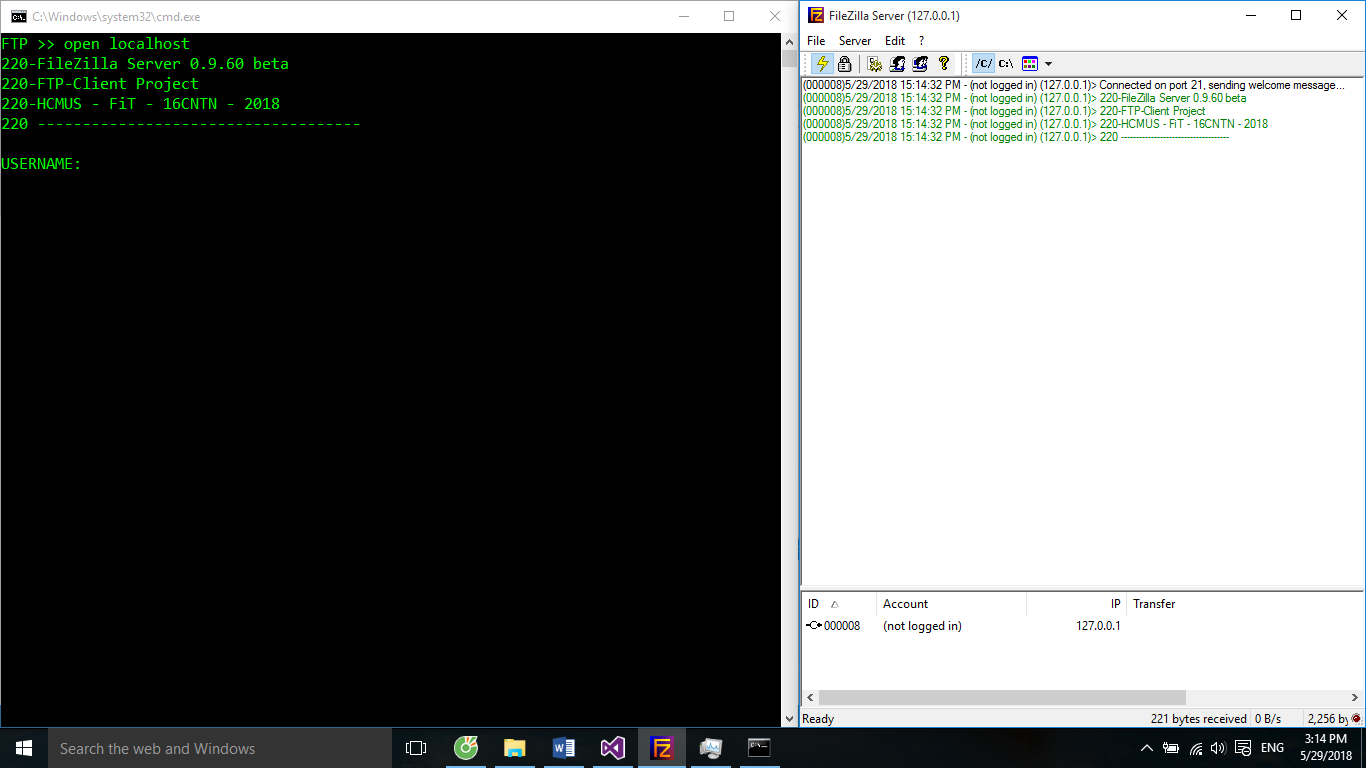
* Dùng phần mềm File **Zilla Server**.
* Cấu hình user trên File Zilla Server:

+ Tên đăng nhập: cfx

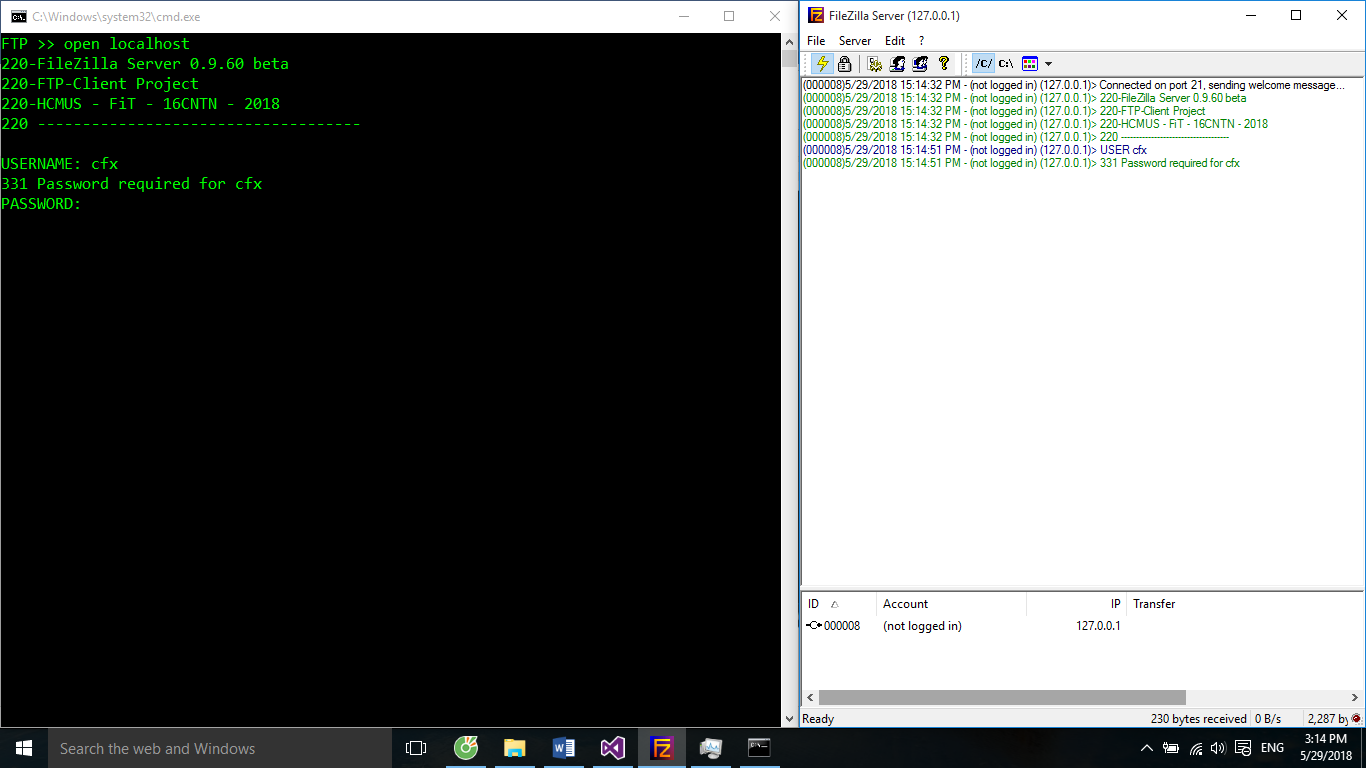
+ Mật khẩu: 1234

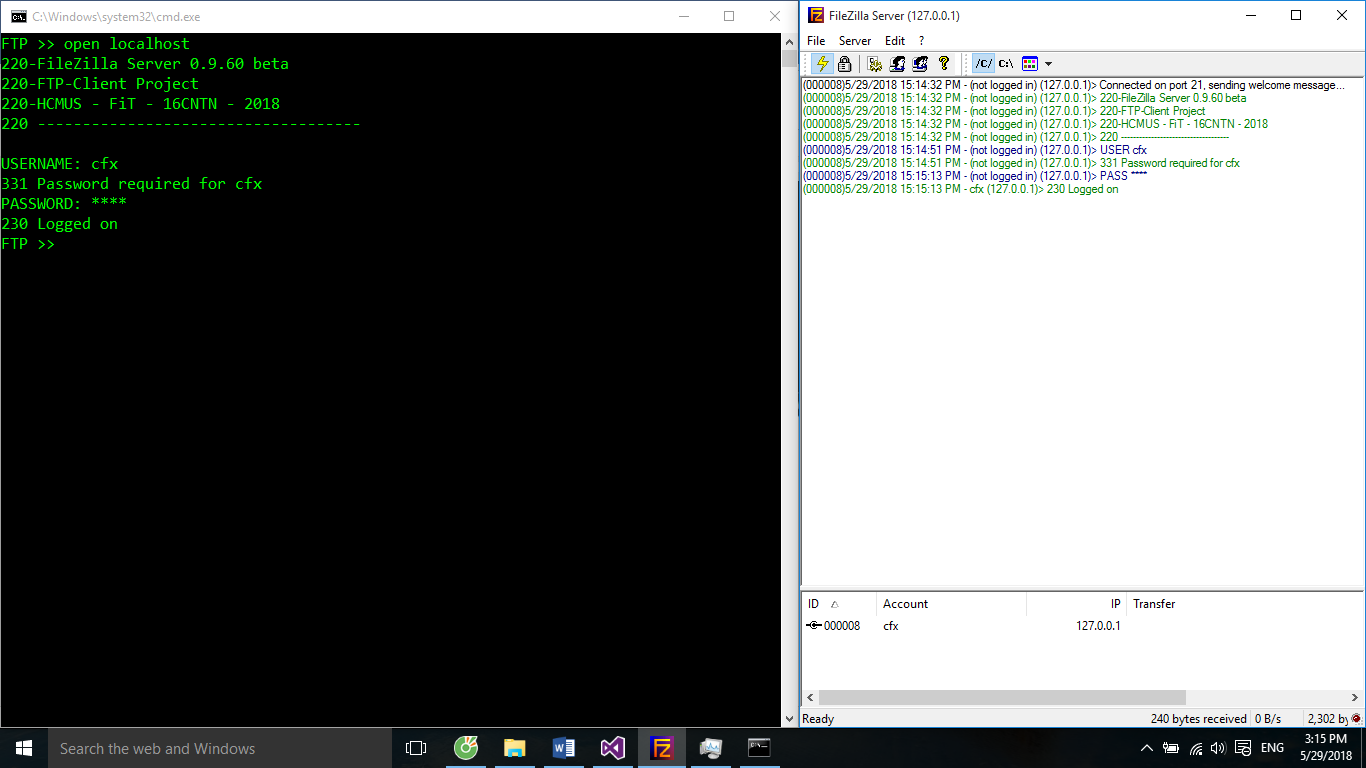
+ Quyền truy cập: Full access

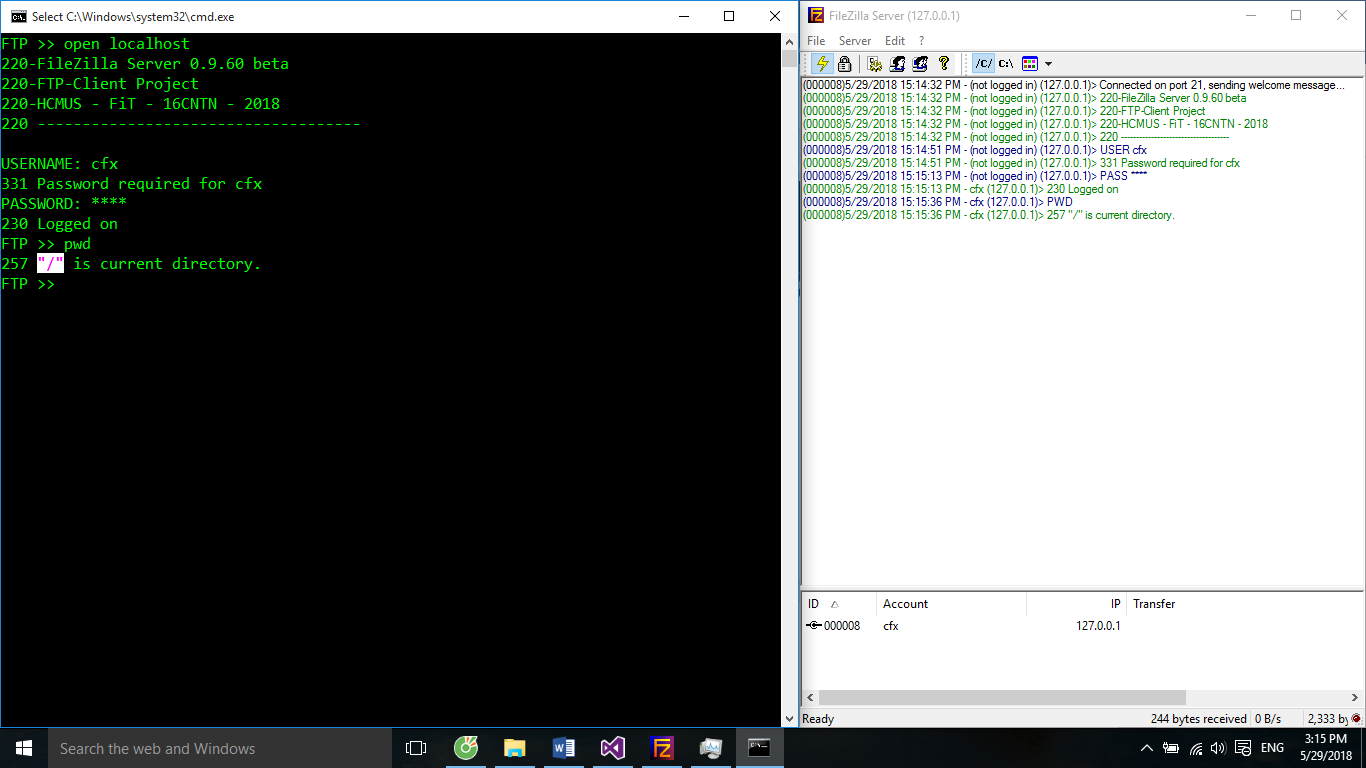
* Nội dung thư mục localhost:
  1. Thử chế độ active (mặc đinh):
* Connect:



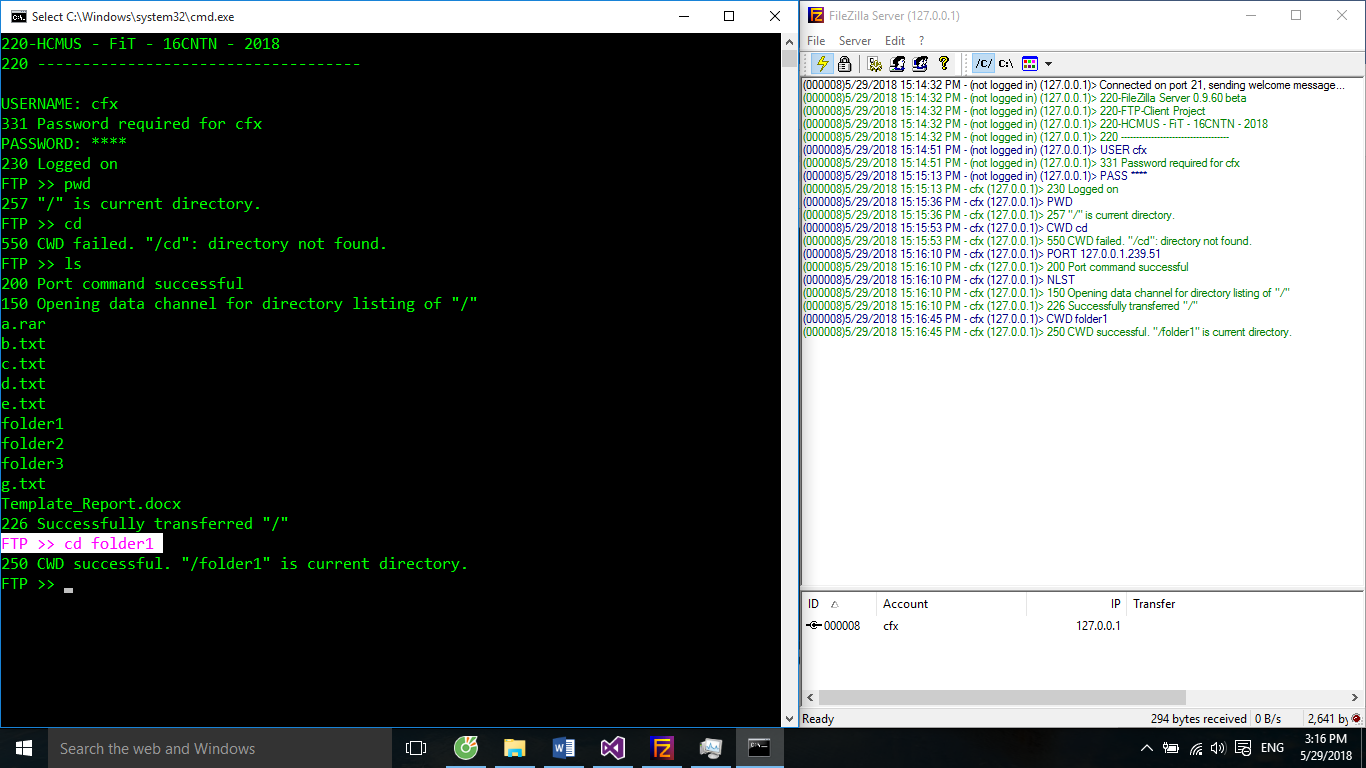
* Login user:



* Password:  
  
* Pwd



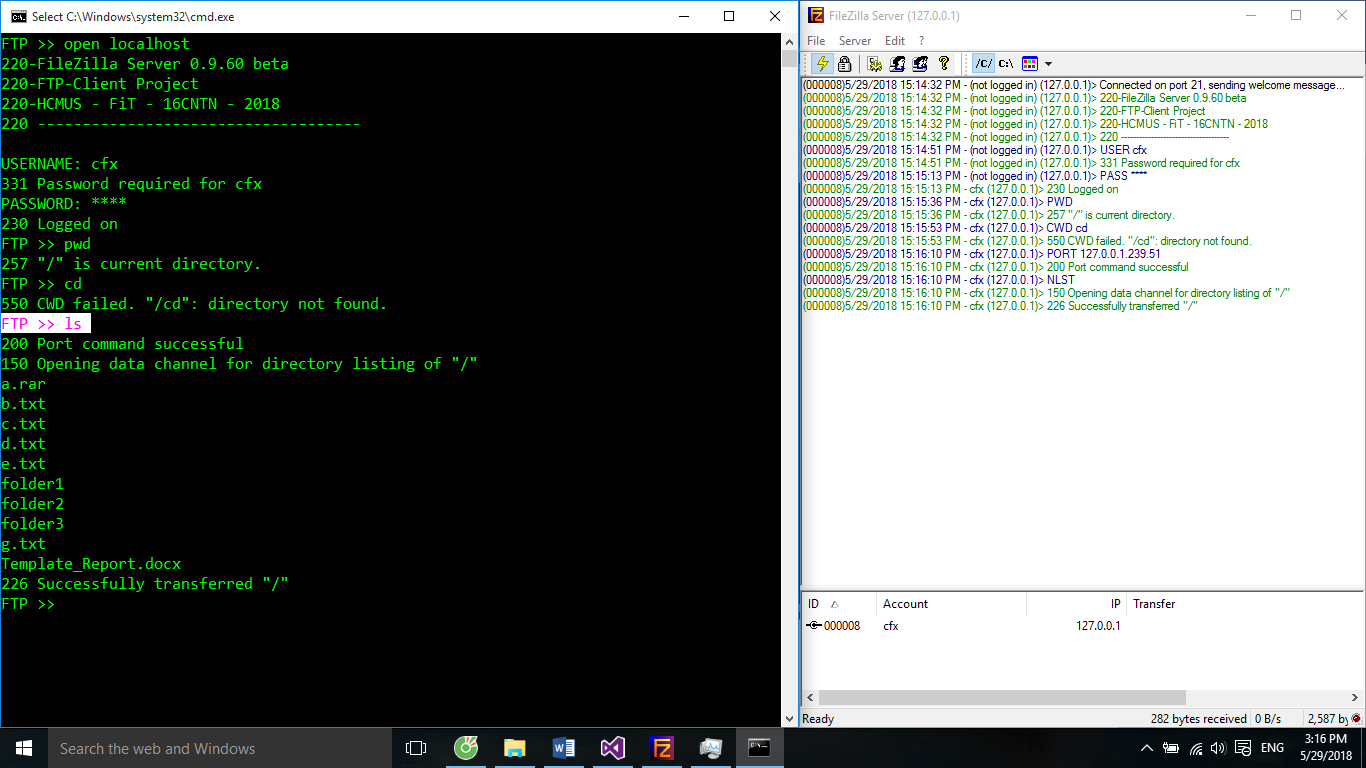
* Cd:



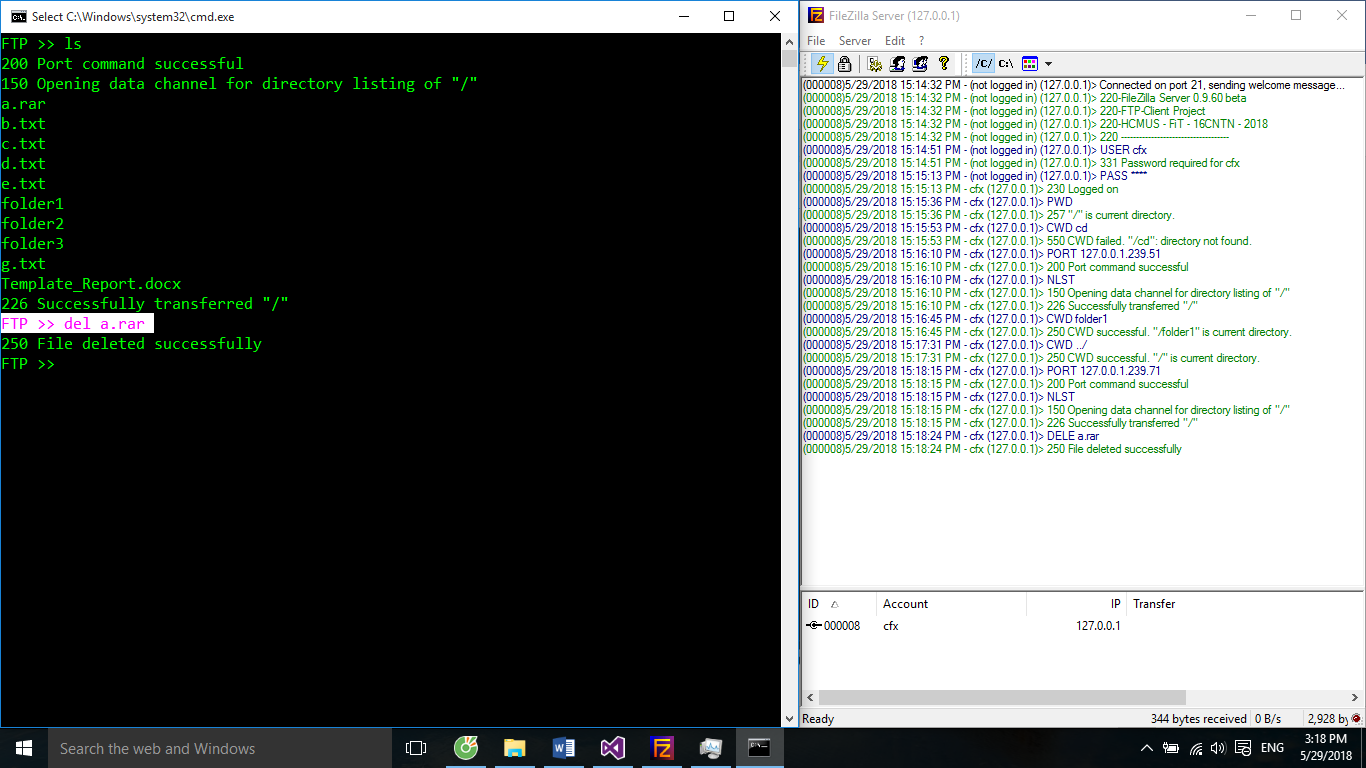
* Lcd

//lỗi

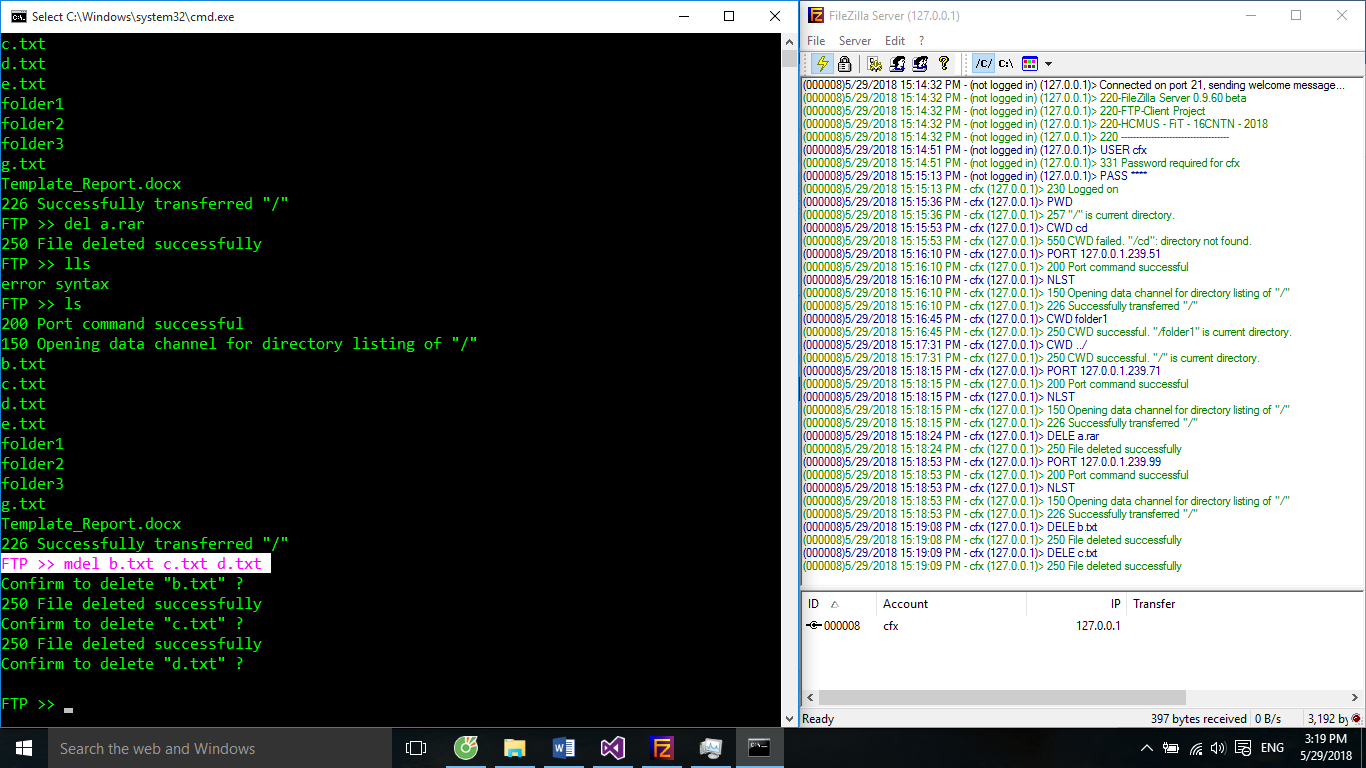
* ls active



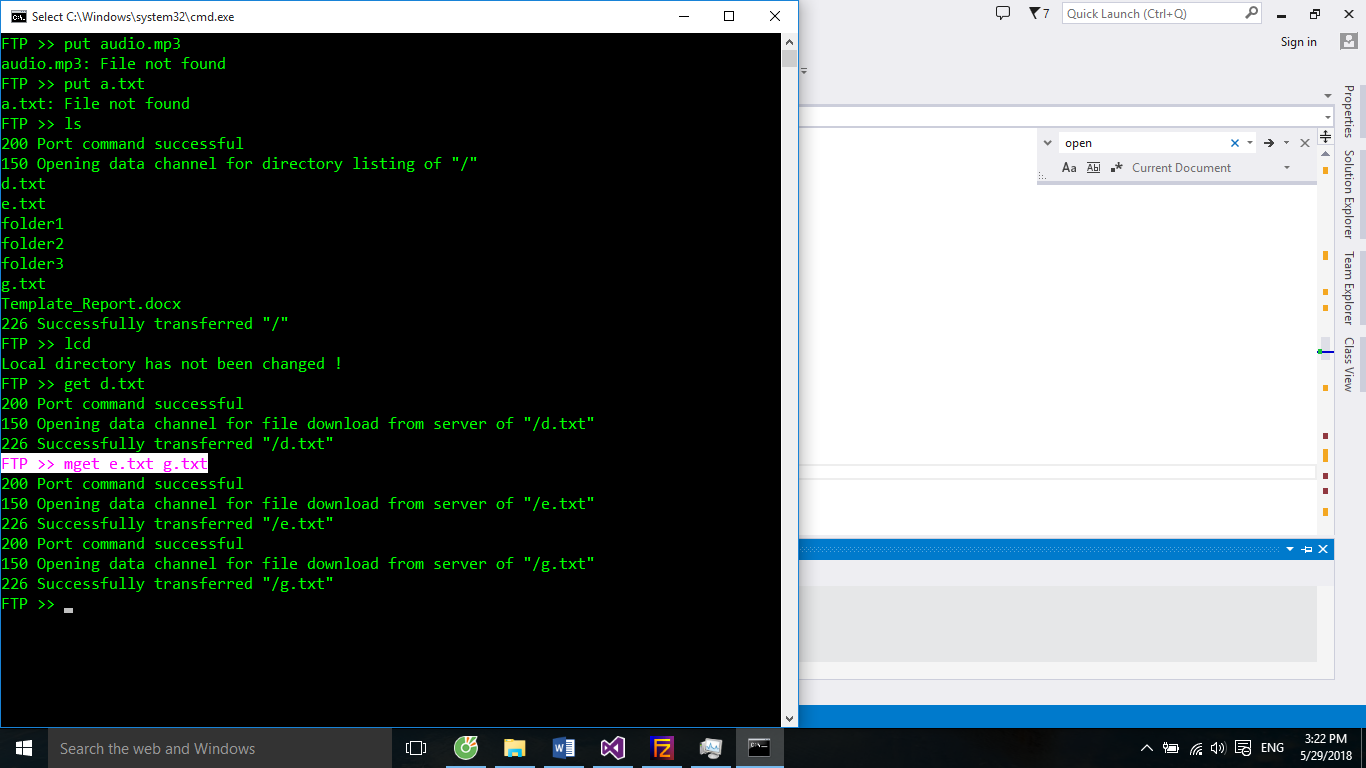
* Del active

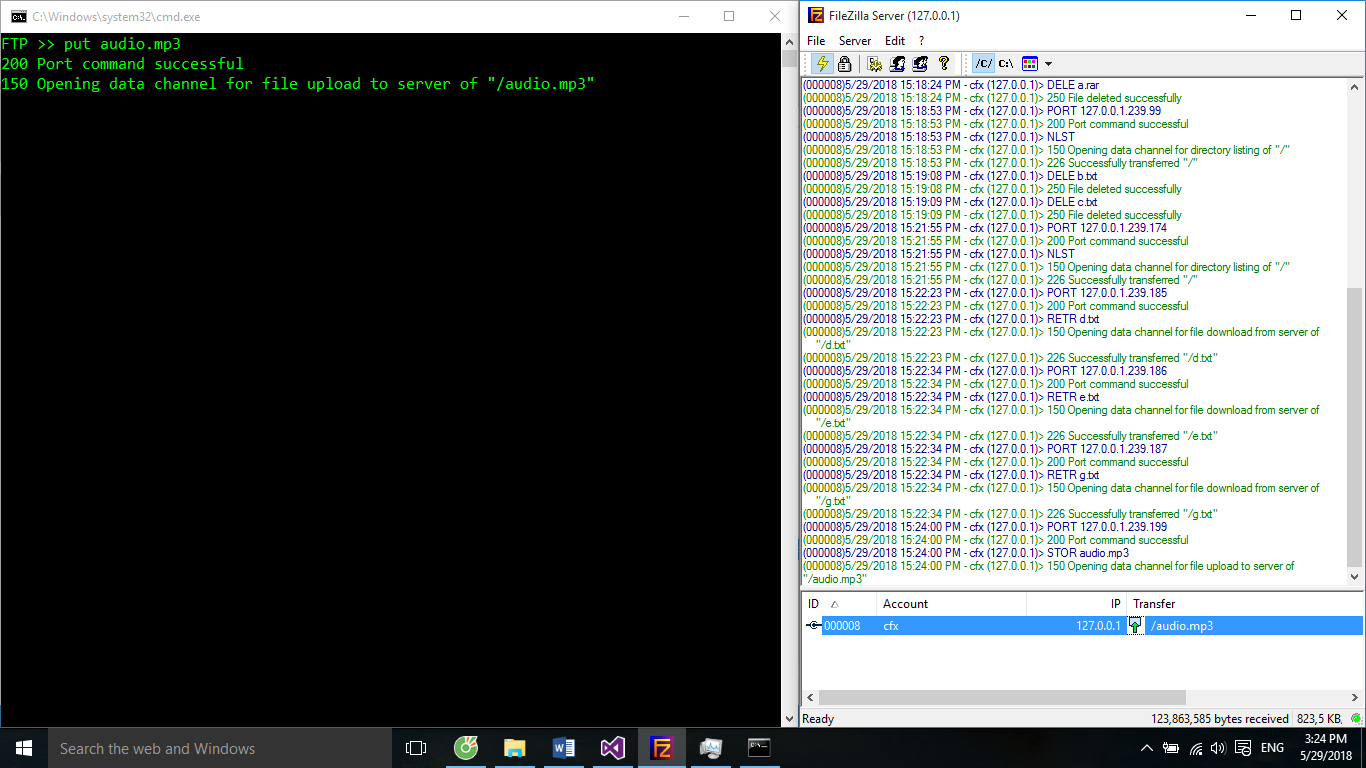
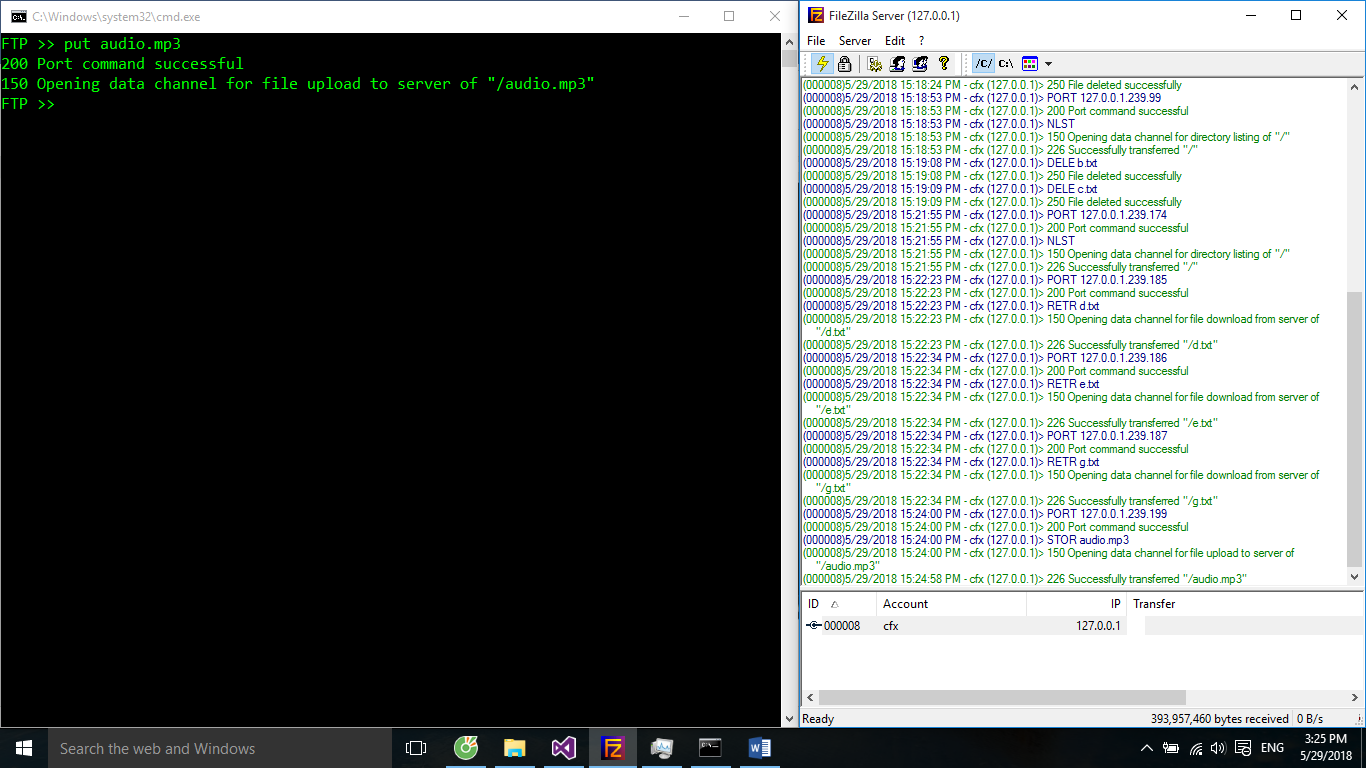


* Mdel active (bỏ qua file d.txt nếu nhấn N)



* Get & mget



* Put: uploading file “audio.mp3” 322MB
*   
  - Put done: (lỗi không bắt được thông báo 226 sau khi đã put thành công)
* 
* Mput: <lỗi>

1. ĐÁNH GIÁ

|  |  |
| --- | --- |
| **Tên hàm** | **Mức độ hoàn thành** |
| bool connect(); | 100% |
| bool login(); | 100% |
| void cmd\_user(); | 100% |
| void cmd\_pass(); | 100% |
| void cmd\_ls(); | 100% |
| void cmd\_pwd(); | 100% |
| void cmd\_cd(); | 100% |
| void cmd\_lcd(); | 100% |
| void cmd\_get(); | 100% |
| void cmd\_mget(); | 100% |
| void cmd\_put(); | Còn lỗi |
| void cmd\_mput(); | 100% |
| void cmd\_del(); | 100% |
| void cmd\_mdel(); | 100% |
| void cmd\_mkdir(); | 100% |
| void cmd\_rmdir(); | 100% |
| void cmd\_pasv(); | 100% |
| void cmd\_quit(); | 100% |
| void cmd\_help(); | 100% |
| Chuyển đổi 2 chế độ | -- |

1. BẮT GÓI TIN VÀ MÔ TẢ QUÁ TRÌNH GỬI NHÂN

* Thiết lập IP máy thật là 192.168.17.1 đặt client ở máy thật và máy ảo là 192.168.17.135 đặt server.

Ảnh có chứa ảnh chụp màn hình

Mô tả được tạo với mức tin cậy rất cao

* Tiến hành yêu cầu **get test.txt** ( yêu cầu download file test.txt từ server về client):

Ở chế độ **active** và **passive** thì client đều gởi port lên server, 2 bên thỏa thuận xong server thông báo **port command successful** về thì client tiếp tục gởi lệnh **RETR test.txt** (tệp cần download là file test.txt) lên server chấp nhận và thông báo về cho client mở data chanel để download file về và gởi về dung lượng , nội dung gói tin cho client. Hoàn tất quá trình gởi dữ liệu server gởi 1 thông báo về cho client rằng đã gởi thành công.

