**THÔNG TIN CHI TIẾT ĐỒ ÁN**

Thành viên nhóm:

* Nguyễn Phan Đức Anh – MSSV: 19120449
* Nguyễn Đình Trí – MSSV: 19120405

Chương trình client và server được viết trên Python 3.7.9

Nếu chạy thẳng từ mã nguồn thì cần cài đặt các thư viện sau:

* PySide2
* psutil
* pillow

Sau khi cài thư viện phụ, dùng Python interpreter chạy script *'server\_gui.py' và 'client\_gui.py'*

# Cấu trúc lệnh và tập lệnh

**Cấu trúc lệnh**

Cú pháp request: LÀ MỘT STRING

**“<MÃ REQUEST> <DỮ LIỆU THÊM>”**

- <MÃ REQUEST> viết IN HOA

- <DỮ LIỆU THÊM> có thể trống

Server trả về reply như sau: LÀ MỘT DÃY BYTES  
**b‘<MÃ TRẠNG THÁI> <Rawdata>’**

**Tập lệnh**

Lưu ý: những request có 2 cột (ví dụ: PROCESS) thì:

* Đầu tiên khi mở cửa sổ process ra thì gửi lệnh ở cột 1 (PROCESS). Nếu trả về ClientState.SUCCEEDED thì mới mở cửa sổ.
* Lúc này server chỉ chấp nhận các lệnh ở cột bên phải và lệnh EXIT
* Gọi lệnh FINISH khi đóng cửa sổ, chỉ đóng khi trả về ClientState.SUCCEEDED

|  |  |  |
| --- | --- | --- |
| ***Request từ client*** | | **<Dữ liệu thêm> *trả về từ Server (nếu có)*** |
| SHUTDOWN <S, L> *(SHUTDOWN S tắt máy, SHUTDOWN L log off máy)* | |  |
| SCREENSHOT | | Rawdata là một dãy bytes của hình có đinh dạng: b’<dài> <rộng> <các pixel>’  Hàm Image.frombytes(“RGBA”, (<dài>,<rộng>), <các pixel>) có thể được dùng để lấy hình. |
| LIVESTREAM(1) | START *(Bắt đầu livestream)* |  |
| STOP  *(Dừng livestream)* |  |
| FINISH |  |
| PROCESS,  APPLICATION | FETCH | Rawdata là dãy bytes chứa một danh sách các process có định dạng json. |
| START <Name> | Tương tự FETCH nhưng là các process sau khi đã mở mới |
| KILL <PID> | Tương tự FETCH nhưng là các process sau khi đã đóng |
| FINISH |  |
| REGISTRY | REGFILE <CONTENT> |  |
| GETVALUE <PATH> <VALUENAME> | Rawdata là giá trị tương ứng |
| SETVALUE <PATH> <VALUENAME> <TYPE(2)> <VALUE> |  |
| DELETEVALUE <PATH> <VALUENAME> |  |
| CREATEKEY <PATH> |  |
| DELETEKEY <PATH> |  |
| FINISH |  |
| KEYLOG | HOOK |  |
| UNHOOK |  |
| LOCK [LOCK\_TIMEOUT(3)] |  |
| UNLOCK(4) |  |
| FETCH | Rawdata là chuỗi bytes key đã log, chuỗi về string bình thường |
| CLEAR |  |
| FINISH |  |
| EXIT | |  |

(1) Lệnh LIVESTREAM có them xử lý đặc biệt từ hai phía server và client. Chi tiết xem ở phần *Thành phần mã nguồn - server > File livestream\_handler.py > class LivestreamHandler* và *Thành phần mã nguồn - client > File stream\_gui.py > class StreamGUI*.

(2) TYPE chỉ nhận các giá trị sau: “BINARY”, “DWORD”, “QWORD”, “STRING”, “MULTISTRING”, “EXPANDABLESTRING”

(3) LOCK\_TIMEOUT tính bằng giây, chỉ định thời gian mà bàn phím sẽ bị khoá. Nếu LOCK\_TIMEOUT trống, bàn phím sẽ được khoá đến khi lệnh UNLOCK được gửi.

(4) UNLOCK vẫn có khả năng thực hiện không thành công, ngay cả khi được gửi sau LOCK trong những trường hợp sau:

1. Khi TIME\_OUT của LOCK đã hết hiệu lực.
2. Khi máy chủ server, bằng một cách nào đó, khôi phục quyền điều khiển của mình trước khi nhận lệnh từ client.

# Thành phần mã nguồn – Phần Client

## File client.py

### class ClientState

Class enum, chỉ các trạng thái yêu cầu từ server

Có các trạng thái:

* SUCCEEDED: yêu cầu được server xử lý thành công.
* FAILED: yêu cầu không được xử lý hoặc thất bại.
* INVALID: yêu cầu không hợp lệ.
* BADMESSAGE: Câu trả lời của server không đúng quy chuẩn.
* NOCONNECTION: chưa có socket giao tiếp nào được mở.
* BADCONNECTION: có vấn đề về socket, dẫn đến message không được gửi thành công hoặc gửi bị thiếu.

### class ClientProgram

Class quy định chương trình client

#### ClientProgram.Run()

Hàm Run chạy một vòng lặp dùng console để gửi và nhận lệnh.  
Chỉ sử dụng để debug.

#### ClientProgram.Connect(host=HOST, port=PORT)

Kết nối đến server tại host, thông qua port.  
thông số mặc định là HOST = 127.0.0.1 và PORT = 6666

#### ClientProgram.Disconnect()

Ngắt kết nối đến server

#### ClientProgram.MakeRequest(request)

Gửi đến server một yêu cầu, đồng thời chờ và nhận câu trả lời từ server.  
Trả về một ClientState và câu trả lời (dạng bytes) từ server.

Mọi giao tiếp đến server nên được gửi và nhận thông qua hàm này

#### ClientProgram.SendMessage(string, binaryData=None)

Gửi một thông điệp đến server.  
Thông điệp bao gồm string: một chuỗi kí tự (sẽ được chuyển thành và binaryData: bất kì dữ liệu nào dưới dạng bytes. Thông điệp được gửi đến server sẽ có dạng ‘string’ + binaryData, cách nhau bởi một byte (kí tự trắng trong UTF-8) duy nhất.

Trả về một giá trị bool. True khi thông điệp được gửi thành công, hoặc False khi thông điệp không được gửi hoặc gửi không đầy đủ.

#### ClientProgram.ReceiveMessage()

Tiếp nhận một thông điệp từ server.

Trả về thông điệp dưới dạng bytes hoặc None (nếu vì bất kì lý do nào mà không nhận được thông điệp hoặc nhận được không đầy đủ)

#### ClientProgram.ProcessReply(self, reply)

Xử lý thông điệp nhận được từ server

Trả về một ClientState và data nhận được từ server

## File client\_gui.py

### class ClientGUI

Class quy định đối tượng trình giao tiếp đồ họa với người dùng

#### ClientGUI.Connect(host)

Kết nối đến server tại host, thông qua port 6666 (quy định sẵn giữa server và client)

#### ClientGUI.Disconnect()

Ngắt kết nối đến server.

#### ClientGUI.OnShutdownButton()

Quy định cách xử lý riêng cho yêu cầu “Tắt máy”

Hàm được gọi khi nhấn nút “Tắt máy”

#### ClientGUI.OnExitButton()

Hàm được gọi khi nhấn nút “Thoát”

#### ClientGUI.ShowWindow()

Hiện cửa sổ chính của chương trình

## File Request\_gui.py

### class Request

Class quy định interface các đối tượng cửa sổ con các thao tác phức tạp của chương trình.

#### Request.OnStartGUI()

Hàm này được gọi khi ấn vào nút tương ứng trên cửa sổ chính.

#### Request.OnExitGUI()

Hàm này được gọi khi ấn vào nút X để tắt cửa sổ con.

#### Request.MakeBaseRequest()

Gửi một yêu cầu đến server, thông báo bắt đầu làm việc với một tập lệnh phức tạp.

#### Request.MakeFinishRequest()

Gửi một yêu cầu đến server, thông báo kết thúc phiên làm việc với tập lệnh phức tạp đã gửi khi cửa sổ hiện lên.

#### Request.ShowWindow()

Hiện cửa sổ con. Mỗi class con cần overload hàm này

## File ProcessRunning\_gui.py

### class ProcessRunning -> Request

Class quy định đối tượng cửa sổ con thực hiện chức năng liên quan đến các process.

#### ProcessRunning.RequestKill(id\_to\_kill)

Hủy process có id là id\_to\_kill ở server.

Hiện thông báo đến client, cho biết việc hủy có thành công không.

#### ProcessRunning.RequestStart(name\_to\_start)

Tiến hành chạy một process có tên name\_to\_start bên server.

Hiện thông báo đến client, cho biết việc chạy có thành công không.

#### ProcessRunning.OnViewButton()

Callback, được gọi khi ấn nút ‘Xem’

Xuất danh sách process lên trên cửa sổ.

#### ProcessRunning.OnEraseButton()

Callback, được gọi khi ấn nút ‘Xóa’

Xóa danh sách process (nếu đã hiện)

#### ProcessRunning.OnStartButton()

Callback, được gọi khi ấn nút ‘Start’

Hiện cửa sổ cho phép người dùng nhập tên của process muốn mở.

#### ProcessRunning.OnKillButton()

Callback, được gọi khi ấn nút ‘Kill’

Hiện cửa sổ cho phép người dùng nhập id của process muốn hủy.

### class ApplicationRunning -> ProcessRunning

Class quy định đối tượng cửa sổ con thực hiện chức năng liên quan đến các process mà được tính là application.

## File Keystroke\_gui.py

### class Keystroke -> Request

Class quy định đối tượng cửa sổ con thực hiện chức năng liên quan đến các process.

#### Keystroke.OnHookButton()

Callback, được gọi khi ấn nút ‘Hook’

Tiến hành cài hook ở phía server, hiện thông báo đến người dùng về trạng thái của Hook.

#### Keystroke.OnUnhookButton()

Callback, được gọi khi ấn nút ‘Unhook’

Tiến hành gỡ hook ở phía server, hiện thông báo đến người dùng về trạng thái của Hook.

#### Keystroke.OnPrintButton()

Callback, được gọi khi ấn nút ‘In phím’

In những phím đã được ghi lại ở server lên cửa sổ

#### Keystroke.OnClearButton()

Callback, được gọi khi ấn nút ‘Xóa’

Xóa cửa sổ in phím bên client cũng như các phím đã lưu ở server

#### Keystroke.PutTextWithNewLine(text)

Hàm phụ trợ

In một dòng chữ lên cửa sổ.

#### Keystroke.ClearText()

Hàm phụ trợ. Xóa cửa sổ in chữ.

## File Registry\_gui.py

### class Registry -> Request

Class quy định đối tượng cửa sổ con thực hiện chức năng liên quan đến các registry.

#### Registry.Browser()

Callback, được gọi khi nhấn nút “Browse…”

Hiện cửa sổ cho phép chọn và mở file .reg.

Đọc nội dung và in lên khung nội dung của cửa sổ “Nội dung”.

#### Registry.OnFileRegContentSend()

Callback, được gọi khi ấn nút “Gửi nội dung”

Gửi nội dung file reg trong khung “Nội dung” đến server, yêu cầu server chạy file .reg này.

Thông báo đến người dùng trạng thái xử lý của yêu cầu này

#### Registry.RequestGetValue(path, valuename)

Yêu cầu server trả về giá trị của Value tên valuename tại subkey có đường dẫn path.

Hiển thị giá trị lấy được hoặc “Lỗi”.

#### Registry.RequestSetValue(path, valuename, type, value)

Yêu cầu server gắn (hoặc tạo nếu chưa tồn tại) giá trị value với kiểu type cho Value tên valuename tại subkey có đường dẫn path.

Hiển thị trạng thái thực thi của yêu cầu (thành công/lỗi)

#### Registry.RequestDeleteValue(path, valuename)

Yêu cầu server xóa Value tên valuename tại subkey có đường dẫn path.

Hiển thị trạng thái thực thi của yêu cầu (thành công/lỗi).

#### Registry.RequestCreateKey(path)

Yêu cầu server tạo subkey với đường dẫn path.

Hiển thị trạng thái thực thi của yêu cầu (thành công/lỗi).

#### Registry.RequestDeleteKey(path)

Yêu cầu server xóa subkey với đường dẫn path.

Hiển thị trạng thái thực thi của yêu cầu (thành công/lỗi).

#### Registry.onFunction()

Hàm phụ trợ, giúp cửa sổ con hiển thị đúng các nút tùy vào thao tác muốn thực hiện.

#### Registry.onSending()

Callback, được gọi khi nhấn nút “Gửi”.

Gửi một yêu cầu đến server đúng theo dữ liệu được người dùng nhập.

#### Registry.onClearing()

Callback, được gọi khi nhấn nút “Xóa”

Xóa khung chữ hiển thị trạng thái.

#### Registry.createNameValueBox()

Hàm phụ trợ hiện khung nhập tên Value

#### Registry.createValueBox()

Hàm phụ trợ hiện khung nhập giá trị Value

#### Registry.createDatatypeBox()

Hàm phụ trợ hiện khung nhập kiểu dữ liệu của Value

## File ScreenCapture\_gui.py

### class Screenshot -> Request

Class quy định đối tượng cửa sổ con thực hiện chức năng liên quan đến các registry.

Khác với các lớp GUI trên, Screenshot không thông báo cho server, vì chụp màn hình chỉ có một yêu cầu duy nhất, do đó OnStartGUI() và OnExitGUI() được overload một cách thích hợp.

#### Screenshot.BytesToImage(rawdata)

Chuyển một dãy bytes thành một đối tượng Image (thuộc module PIL), sẵn sàng để hiển thị hoặc lưu.

Dãy bytes cần có cấu trúc đúng như đã quy định trước giữa client và server

#### Screenshot.CaptureScreen()

Yêu cầu server chụp màn hình của mình và gửi về cho client.

#### Screenshot.SavePicture()

Lưu một hình đã chụp vào đĩa. Địa điểm lưu và tên file được người dùng chọn bằng một cửa sổ Browser.

# Thành phần mã nguồn – Phần Server

## File server.py

### class ServerProgram

Class quy định đối tượng chương trình phía server, kết nối với một client, tiếp nhận và xử lý yêu cầu của client đó.

ServerProgram sử dụng các “handler” để xử lý các request của mình. Mỗi handler nằm trên một file riêng biệt.

#### ServerProgram.OpenServer(host=HOST, port=PORT, backlog=BACKLOG)

Mở socket Server tại host, port với số backlog, chờ và chấp nhận một kết nối duy nhất đến.

Thông số mặc định: HOST = 0.0.0.0, PORT = 6666, BACKLOG = 5

#### ServerProgram.CloseServer()

Đóng socket kết nối giữa client và server.

#### ServerProgram.Run()

Khởi động vòng lặp tiếp nhận yêu cầu – Xử lý yêu cầu – Trả lời.

Vòng lặp vô tận cho đến khi client muốn kết thúc hoặc kết nối bị gián đoạn.

#### ServerProgram.ReceiveMessage()

Lắng nghe và nhận một thông điệp/yêu cầu từ client.

Trả về thông điệp dưới dạng chuỗi kí tự hoặc None nếu có lỗi xảy ra

#### ServerProgram.SendMessage()

Gửi một thông điệp (câu trả lời) cho client.

Trả về True nếu gửi thành công hoặc False nếu có lỗi xảy ra

#### ServerProgram.HandleRequest(requestString)

Nhận yêu cầu requestString, sử dụng một handler hợp lý để xử lý request này, gói trạng thái xử lý và dữ liệu cần thiết thành một thông điệp và gửi cho client.

Trả về hằng số CONTINUE\_PROGRAM nếu server nên tiếp tục lắng nghe hoặc QUIT\_PROGRAM nếu client có gửi yêu cầu ngắt kết nối.

#### ServerProgram.SplitRequest(request)

Hàm phụ trợ. Hỗ trợ tách một thông điệp yêu cầu của client thành hai phần:

* + request: là lệnh.
  + extraData: các thông số.

Trả về hai phần này, hoặc thay thế bất kì bằng None nếu thông điệp thiếu phần đó.

## File server\_gui.py

### openServer()

Hàm wrapper để mở server tại (0.0.0.0, 6666) và vào vòng lặp ServerProgram.Run() khi có kết nối.

Được gọi khi nhấn nút “Mở Server” ở cửa sổ GUI.

### MakeWindow()

Hiển thị cửa sổ server với nút “Mở Server”

## File handler\_state.py

### class HandlerState

Một enum, định nghĩa những trạng thái mà một handler có thể trả về khi gọi handler.Execute(). Gồm có:

* + SUCCEEDED: xử lý thành công.
  + FAILED: yêu cầu hợp lệ nhưng xử lý thất bại.
  + INVALID: yêu cầu không hợp lệ

## File process\_handler.py

### class ProcessHandler

Handler xử lý những yêu cầu liên quan đến Process

#### ProcessHandler.Execute(reqCode, data)

Phân tích reqCode, data và xử lý yêu cầu.

Trả một HandlerState tùy theo trạng thái xử lý yêu cầu và extraData là dữ liệu cần gửi về client (có thể None nếu không có dữ liệu cần trả về).

Execute bắt mọi lỗi có thể xảy ra và trả về HandlerState.FAILED nếu điều này xảy ra.

#### ProcessHandler.FetchAndUpdate()

Lấy danh sách các process của máy và lưu tạm thời lại.

#### ProcessHandler.FetchWithPIDs(pids)

Hàm bổ trợ.

Với mỗi process\_id trong danh sách pid, trả về danh sách các process tương ứng.

#### ProcessHandler.KillProcess(pid)

Tắt process có id là pid đang chạy trong máy, nếu có.

Nếu không, báo lỗi (lỗi này sẽ được xử lý bởi Execute() nếu hàm được Execute gọi)

#### ProcessHandler.StartProcess(name)

Chạy process ứng với tên name.

## File application\_handler.py

### WindowEnumCallback(hwindow, param)

Hàm phụ trợ.

Sử dụng để lấy các process được nhận định là “Application” (tên cửa sổ chính khác rỗng)

### class ApplicationHandler -> ProcessHandler

Handler xử lý những yêu cầu liên quan đến Process. Kế thừa từ ProcessHandler

#### ApplicationHandler.FetchAppPIDSet()

Lấy và trả về một danh sách các process id đươc nhận định là Application.

## File vkcode.py

vkcode chứa các hằng số liên quan đến Virtual Key Code (vkCode) của Win32 API.

Dùng để hỗ trợ cho KeylogHandler.

### NormalChar(vkCode)

Hàm kiểm tra vkCode có tương ứng với phím kí tự thông thường (số, chữ cái, kí tự đặc biệt,…)

### VK\_CODE

Danh sách 255 vkCOde (dựa trên bàn phím US thông dụng) và “tên” phím tương ứng mà KeylogHandler sẽ ghi lại.

### VK\_SHIFT

Danh sách các vkCode tương ứng với phím SHIFT

### VK\_CAPS\_LOCK

vkCode ứng với phím CAPS LOCK

## File keystroke\_handler.py

### class KBDLLHOOKSTRUCT

C struct dùng trong LowLevelKeyboardProc() để lấy ra vkCode cho phím được nhấn

### LowLevelKeyboardProc(nCode, wParam, lParam)

Hàm dùng để ghi lại các phím được nhấn

### InstallHookAndLoop(lpParameter)

“Gắn” một hook có callback là LowLevelKeyboardProc() (ở trên) vào hệ điều hành, đồng thời tạo một vòng lặp message. Khi vòng lặp này kết thúc là báo hiệu cho thời gian vận hành đã hết và hook sẽ được gỡ.

lpParameter là thông số ‘giả’. Có thông số này vì hàm CreateThread của Wn32 API yêu cầu một thông số.

### StartThread(function)

Wrapper cho hàm CreateThread của Win32 API. Tạo một thread chạy function trên đó với thông số NULL.

Trả về một HANDLE cho thread và DWORD là id của thread

### class Logger

Logger dùng như một class static. Logger lưu lại dữ liệu vào một file trên đĩa cứng và có thể trích dẫn nội dung bất kì lúc nào.

#### Logger.Init(filepath)

Khởi động logger với dữ liệu ghi được lưu vào file tại đường dẫn filepath

#### Logger.Del()

Tắt logger, xóa dữ liệu đã lưu

#### Logger.Log(vkCode, isCapitalized)

Ghi lại kí tự tương ứng với vkCode.

Nếu vkCode tương ứng với một phím kí tự, isCapitalized quyết định nếu phím ghi lại là chữ thường hay chữ hoa (hoặc kí tự dưới hay trên đối với phím ký tự đặc biệt và phím số)

Nếu vkCode không phải kí tự, phím ghi lại sẽ có thêm cặp dấu <> để quy ước đó là phím đặc biệt có tên dài.

#### Logger.DumpBuffer()

Logger sử dụng một buffer tạm lưu trữ MAX\_BUFFER\_LENGTH dữ liệu trước khi đưa vào file lưu trên đĩa.

DumpBuffer lưu dữ liệu tạm này vào đĩa và xóa buffer

#### Logger.Read()

Đọc và trả về dữ liệu được ghi lại từ cả trên đĩa và trong buffer.

### class KeystrokeHandler

Handler xử lý những yêu cầu liên quan đến việc ghi lại phím gõ ở máy tính chạy server.

#### KeystrokeHandler.Execute(reqCode, data)

Phân tích reqCode, data và xử lý yêu cầu.

Trả một HandlerState tùy theo trạng thái xử lý yêu cầu và extraData là dữ liệu cần gửi về client (có thể None nếu không có dữ liệu cần trả về).

Execute bắt mọi lỗi có thể xảy ra và trả về HandlerState.FAILED nếu điều này xảy ra.

#### KeystrokeHandler.Hook()

Cài hook để bắt đầu quá trình ghi ký tự.

#### KeystrokeHandler.Unhook()

Gỡ bỏ hook đã cài.

## File registry\_handler.py

### class RegistryHandler

Handler xử lý những yêu cầu liên quan đến đọc, sửa Registry của máy server

#### RegistryHandler.Execute(reqCode, data)

Phân tích reqCode, data và xử lý yêu cầu.

Trả một HandlerState tùy theo trạng thái xử lý yêu cầu và extraData là dữ liệu cần gửi về client (có thể None nếu không có dữ liệu cần trả về).

Execute bắt mọi lỗi có thể xảy ra và trả về HandlerState.FAILED nếu điều này xảy ra.

Đối với yêu cầu chạy file .reg. Một file tempreg.reg được lưu tại thư mục vận hành để chạy.

#### RegistryHandler.UseRegFile(filepath)

Chạy file .reg tại filepath trên máy một cách im lặng

#### RegistryHandler.GetValue(keyPath, valueName))

Lấy giá trị của Value tên valueName của subkey tại đường dẫn keyPath

#### RegistryHandler.SetValue(keyPath, valueName, typeValue, value)

Gán (nếu đã tồn tại) hoặc tạo mới một Value tên valueName với value thuộc kiểu typeValue cho subkey tại đường dẫn keypath.

#### RegistryHandler.DeleteValue(keyPath, valueName)

Xóa Value tên valueName của subkey tại đường dẫn keyPath

#### RegistryHandler.CreateKey(keyPath)

Thêm subkey ứng với đường dẫn keyPath

#### RegistryHandler.DeleteKey(keyPath)

Xóa subkey ứng với đường dẫn keyPath

#### RegistryHandler.GetKeyHandle(keyPath, access)

Hàm phụ trợ.

Trả về đối tượng key ứng với subkey của registry tại keypath, được mở với mức truy cập access.

#### RegistryHandler.GetSeperatedBaseAndSubKey(string)

Hàm phụ trợ, tách và trả về đối tượng key base và một chuỗi kí tự sub.

* + base ứng với một trong các khóa HKEY\_CLASSES\_ROOT, HKEY\_CURRENT\_USER, HKEY\_LOCAL\_MACHINE,… (do yêu cầu của API trong winreg của Python)
  + sub ứng với đường dẫn đến sau khi truy cập vào base.

## File screenshot\_handler.py

### class ScreenshotHandler

Handler xử lý những yêu cầu liên quan đến việc chụp màn hình

#### ScreenshotHandler.Execute(reqCode, data)

Hàm Execute của ScreenshotHandler luôn trả về dữ liệu trả lời cho lệnh chụp màn hình, không kể reqCode, data.

#### ScreenshotHandler.TakeScreenshotAsBytes()

Chụp màn hình và trả về width, height, byte\_data.

* width, height là chiều dài và rộng của hình chụp được.
* byte\_data là một dãy bytes được chuyển đổi từ đối tượng hình chụp được.

## File livestream\_handler.py

### class LivestreamHandler

Handler xử lý những yêu cầu liên quan đến việc phát trực tiếp màn hình.

#### LivestreamHandler.Execute(reqCode, data)

Phân tích reqCode, data và xử lý yêu cầu.

Nếu yêu cầu là “START”, hàm tạo ra một luồng xử lý con chạy hàm Livestream.

Nếu yêu cầu là “STOP”, kết thúc luồng xử lý trên.

Mọi sự cố xảy ra trên luồng xử lý gửi hình đều yêu cầu lệnh “STOP” rồi “START” một lần nữa.

#### Livestream(hostSocket)

Hàm chịu trách nhiệm gửi liên tục những gói tin chứa hình chụp màn hình cho client qua một port mới (mà client sẽ phải kết nối đến trước khi bắt đầu gửi gói tin qua việc connect đến hostSocket). Luồng xử lý này *cố gắng* đạt được tốc độ gửi 24 hình/giây.

### SendMessage(sock, string, binaryData)

Gửi một thông điệp string + binaryData qua socket sock.

## File shutdown\_handler.py

### class ShutdownHandler

Handler xử lý những yêu cầu liên quan đến việc tắt máy tính

#### ShutdownHandler.Execute(reqCode, data)

Hàm Execute của ScreenshotHandler nhận data là “S” hoặc “L” (ứng với thông điệp “SHUTDOWN S” và “SHUTDOWN L”).

Nếu data là “S” thì kết nối sẽ được ngắt, chương trình server tắt và máy tính sẽ tắt trong 3 giây.

Nếu data là “L” thì kết nối sẽ được ngắt, chương trình server tắt và người dung máy server sẽ bị log out trong 3 giây.

## File info\_handler.py

### class ShutdownHandler

Handler xử lý những yêu cầu liên quan đến lấy các thông tin từ server.

#### ShutdownHandler.Execute(reqCode, data)

Phân tích reqCode, data và xử lý yêu cầu.

Nếu data là “MACADDRESS”: trả về kết quả là địa chỉ MAC của máy server dưới định dạng: “XX:XX:XX:XX:XX:XX”.