BÁO CÁO ĐỒ ÁN LẬP TRÌNH SOCKET
Mạng máy tính
Lớp: 19TN

1. **Thành viên:**
    - Nguyễn Minh Uyên - 19120154
    - Nguyễn Lê Bảo Thi - 19120376
2. **Đánh giá đồ án**
    - Mức độ hoàn thành đạt 95%
    - Phần chưa làm tốt: còn 1 số lỗi xử lý các phím đặc biệt ở hàm printKeys trong tính năng Key Stroke; phần hiển thị nội dung file .reg ở tính năng cập nhật Registry bằng file .reg bị lỗi font chữ (trong video demo em sợ thầy không đọc được nội dung file .reg đó nên đã mở file đó ở ngoài chương trình để copy lại nội dung dán lên giao diện).
    - Video demo được tách thành 2 phần: phần 1 demo mở server, client để kết nối và thực hiện các tính năng; phần 2 là sau khi tính năng shutdown thực hiện tắt máy, em mở lại để demo nốt phần đóng kết nối. Do có lỗi trong quá trình cài máy ảo nên em xin phép quay video demo trên 2 máy thật rồi ghép lại để thầy tiện theo dõi.
3. **Kịch bản giao tiếp của chương trình**

    Khi mở server, server tạo một socket với port mặc định là 65432 để lắng nghe bằng giao thức TCP. Client khởi động, tạo một socket để gửi tín hiệu kết nối cho server. Khi nhận được tín hiệu kết nối từ client, socket server chấp nhận kết nối và tạo một socket mới để thực hiện giao tiếp với client. Chương trình tiếp tục bằng việc client gửi yêu cầu thực hiện một tính năng cho server, server nhận yêu cầu và tiếp tục giao tiếp với client để thực hiện xong tính năng đó. Server và client đều chạy trên một thread, tức là tại một thời điểm chỉ gửi/nhận hoặc thực hiện thao tác của tính năng (trừ khi thực hiện thao tác hook của tính năng Keystroke). Khi kết thúc một tính năng, client gửi thông điệp "Quit" cho server để server sẵn sàng nhận yêu cầu thực hiện tính năng khác. Để kết thúc chương trình, client cũng gửi thông điệp "Quit" cho server để ngắt kết nối, đóng các socket và chương trình.
4. **Môi trường lập trình và các framework hỗ trợ để thực thi ứng dụng**

- Môi trường lập trình:
  - Ngôn ngữ lập trình: Python 3.
  - Hệ điều hành: Windows 10.
- Các thư viện hỗ trợ để thực thi ứng dụng:
  - socket: hỗ trợ object socket với các tính năng cơ bản
  - pynput: hỗ trợ ghi nhận thao tác bàn phím
  - tkinter: hỗ trợ thiết kế giao diện
  - pillow (PIL): hỗ trợ xử lý ảnh
  - pyautogui: hỗ trợ chụp ảnh màn hình
  - winreg: hỗ trợ thao tác với registry
  - os: hỗ trợ sử dụng các lệnh của hệ điều hành, dùng để thực hiện các tính năng Process và Application
  - io (BytesIO): quản lý các hoạt động đầu vào và đầu ra liên quan đến tệp.
  - functools (partial): Hỗ trợ các hàm bậc cao hoạt động trên các chức năng khác

5. Hướng dẫn sử dụng các tính năng của chương trình

5.1. Client

5.1.1. Thiết lập kết nối tới server

Bước 1: Nhập địa chỉ IP của host server. Port mặc định là: 65432

Bước 2: Nhấn nút "Connect to server" để kết nối tới server.

Lưu ý: Sau khi nhấn nút trên client sẽ gửi yêu cầu muốn kết nối tới server và sẽ nhận được thông báo là "Kết nối tới server thành công" nếu server đã được mở và tồn tại. Còn không thì sẽ có thông báo là "Chưa kết nối tới server".

5.1.2. Process Running

Ý nghĩa: Một **tiến trình** (process) là một thể hiện của một chương trình đang được thực thi. Mỗi quy trình đang chạy trong Windows được gán một số thập phân duy nhất được gọi là **ID quy trình** hoặc PID. Ở chức năng ta có thể xem các process, diệt process, bật process của server và xóa các process đang hiển thị trên cửa sổ của client.

Các chức năng:

- Kill: Để diệt process

  Khi chọn chức năng này sẽ hiển thị một cửa sổ. Client nhập ID của process muốn diệt. Sau khi nhập xong ID

process nhấn nút Kill để gửi yêu cầu cho server diệt process đó. Có hộp thoại xuất hiện để thông báo diệt thành công hay thất bại.

- Xem: Để hiển thị danh sách các process
  Khi chọn chức năng này thì client sẽ gửi yêu cầu đến server, server gửi lại cho client danh sách các process đang thực thi và hiển thị danh sách các process lên cửa sổ của client.
- Xóa: Để xóa hiển thị các process trên cửa sổ của client.
  Khi chọn chức năng này client sẽ tiến hành xóa hiển thị danh sách các process trên cửa sổ.
- Start: Để bật process trên server
  Khi chọn chức năng này sẽ hiển thị một cửa sổ yêu cầu nhập tên của process muốn bật (ví dụ notepad). Sau khi nhập xong tên process nhấn nút Start để gửi yêu cầu cho server bật process đó. Có hộp thoại xuất hiện để thông báo bật thành công hay thất bại.

5.1.3. App Running

Ý nghĩa: Một ứng dụng (application) là một thể hiện của một ứng dụng đang được thực thi. Mỗi ứng dụng đang chạy trong Windows được gán một số thập phân duy nhất được gọi là ID application. Ở chức năng ta có thể xem, diệt, bật application của server và xóa các application đang hiển thị ở client.

Các chức năng:

- Kill: Để diệt application
  Khi chọn chức năng này sẽ hiển thị một cửa sổ yêu cầu nhập ID của application muốn diệt. Sau khi nhập xong ID application nhấn nút Kill để gửi yêu cầu cho server diệt application đó. Có hộp thoại xuất hiện để thông báo diệt thành công hay thất bại.
- Xem: Để hiển thị danh sách các application hiện tại trên server
  Khi chọn chức năng này thì client sẽ gửi yêu cầu đến server, server gửi lại cho client danh sách các application để hiển thị lên cửa sổ.

- Xóa: Để xóa hiển thị các application trên cửa sổ của client
  Khi chọn chức năng này client sẽ tiến hành xóa hiển thị danh sách các application trên cửa sổ.
- Start: Để bật application trên server
  Khi chọn chức năng này sẽ hiển thị một cửa sổ yêu cầu nhập tên của application muốn bật (ví dụ notepad). Sau khi nhập xong nhấn nút Start để gửi yêu cầu cho server bật application đó. Có hộp thoại xuất hiện để thông báo bật thành công hay thất bại.

5.1.4. Keystroke

Ý nghĩa: Là trình theo dõi thao tác bàn phím của server.

Các chức năng:

- Hook: Bật ghi nhận lại thao tác bàn phím của server.
- Unhook: Tắt ghi nhận lại thao tác bàn phím của server.
- In phím: Hiển thị những thao tác bàn phím đã ghi nhận được lên cửa sổ của client.
- Xóa: Xóa hiển thị các phím đã in trên cửa sổ của client.

5.1.5. Registry

Ý nghĩa: thực hiện các thao tác cập nhật registry của server.

Các chức năng:

- Sửa registry thông qua file .reg: chọn đường dẫn tới file .reg chứa nội dung cần sửa trên máy client, nội dung này sẽ hiển thị lên cửa sổ. Bấm nút "Gửi nội dung" để gửi đến server. Server nhận nội dung này, ghi vào một file .reg tự tạo rồi import vào Registry editor để cập nhật.
- Sửa registry trực tiếp: chọn một trong 5 chức năng: get/set/delete value, create/delete key. Với các chức năng get value và delete value, nhập đường dẫn và tên value cần xem hoặc xóa. Với set value, nhập đường dẫn, tên value, value và kiểu dữ liệu cần set. Với create và delete key, nhập đường dẫn tới key cần tạo/xóa. Sau đó bấm "Gửi" để gửi yêu cầu cho server, server nhận và cập nhật registry theo yêu cầu và gửi

lại thông báo thao tác thành công hay thất bại cho client hiển thị lên cửa sổ. Nút "Xóa" dùng để xóa các thông báo đã hiển thị trên cửa sổ của client.

5.1.6. Print Screen

Ý nghĩa: Dùng để chụp màn hình server và lưu ảnh chụp màn hình vào client.

Các chức năng:

- Take: Dùng để gửi yêu cầu chụp màn hình đến server, server khi nhận được yêu cầu thực hiện chụp màn hình và gửi ảnh chụp lại cho client hiển thị lên cửa sổ.
- Save: Dùng để lưu hình ảnh đã được chụp từ server đang được hiển thị trên màn hình.

5.1.7. Shut Down

Ý nghĩa: Dùng để gửi yêu cầu tắt máy server. Server khi nhận yêu cầu sẽ tự động tắt máy sau 1 phút.

5.1.8. Quit

Ý nghĩa: Dùng để gửi yêu cầu ngắt kết nối đến server và đóng cửa sổ client.

5.2. Server

Bấm nút "Mở" để khởi động server, server sẽ chờ đợi tín hiệu kết nối từ client. Khi client đóng, server cũng sẽ đóng theo.

6. Giải thích các hàm quan trọng trong chương trình

- **Client**
  - Class ClientGUI
    - __init__(self, master): Tạo GUI chính của client gồm có ô để nhập IP của server để kết nối các phím gồm phím Connect to Server, Process Running, App Running, Keystroke, Registry, Print Screen, Shut down, Quit khi nhấn vào mỗi nút sẽ mở hàm tương ứng với các nút đó ở trong class này.
    - Closing(self): Gửi yêu cầu "QUIT" để ngắt kết nối với server và tiến hành xóa cửa sổ
    - ProcessRunning(self): Kiểm tra xem đã kết nối với server chưa nếu như chưa kết nối thì sẽ xuất hiện hộp thoại "Chưa kết nối đên server" còn ngược lại sẽ gửi yêu cầu "PROCESS" đến server và tạo class ProcessGUI để thực hiện các yêu cầu trên đó.

        - AppRunning(self): Kiểm tra xem đã kết nối với server chưa nếu như chưa kết nối thì sẽ xuất hiện hộp thoại "Chưa kết nối đên server" còn ngược lại sẽ gửi yêu cầu "APPLICATION" đến server và tạo class AppGUI để thực hiện các yêu cầu trên đó.
        - Keystroke(self): Kiểm tra xem đã kết nối với server chưa nếu như chưa kết nối thì sẽ xuất hiện hộp thoại "Chưa kết nối đên server" còn ngược lại sẽ gửi yêu cầu "KEYLOG" đến server và tạo class KeystrokeGUI để thực hiện các yêu cầu trên đó.
        - Registry(self): Kiểm tra xem đã kết nối với server chưa nếu như chưa kết nối thì sẽ xuất hiện hộp thoại "Chưa kết nối đên server" còn ngược lại sẽ gửi yêu cầu "REGISTRY" đến server và tạo class RegistryGUI để thực hiện các yêu cầu trên đó.
        - PrintScreen(self): Kiểm tra xem đã kết nối với server chưa nếu như chưa kết nối thì sẽ xuất hiện hộp thoại "Chưa kết nối đên server" còn ngược lại sẽ gửi yêu cầu "TAKEPIC" đến server và tạo class PicGUI để thực hiện các yêu cầu trên đó.
        - ShutDown(self): Kiểm tra xem đã kết nối với server chưa nếu như chưa kết nối thì sẽ xuất hiện hộp thoại "Chưa kết nối đên server" còn ngược lại sẽ gửi yêu cầu "SHUTDOWN" đến server và đóng kết nối.
        - Quit(self): Kiểm tra xem đã kết nối với server chưa nếu như chưa kết nối thì sẽ đóng cửa sổ còn ngược lại sẽ gửi yêu cầu "QUIT" đến server và ngắt kết nối đồng thời đóng cửa sổ
    - connectServer(IPVar): Tham số truyền vào là stringVar(). Lấy IP được nhập vào bằng IPVar.get(). Sử dụng hàm .connect(HOST, PORT) tiến hành connect tới server có địa chỉ là IP mới nhập vào và port mặc định là 65432. Nếu kết nối thành công sẽ hiện hộp thoại "Kết nối đến server thành công" và nếu có lỗi sẽ là "Chưa kết nối đến server"
    - receive: Sử dụng hàm .recv(BUFF_SIZE) để nhận dữ liệu một lần tối đa BUFF_SIZE = 4069 bytes và dùng vòng lặp

cho tới khi nhận đủ. Kết quả trả về là data đã nhận phía trên đã được giải mã và xóa đi các kí tự trắng thừa đầu và cuối

- receive1: Sử dụng hàm .recv(BUFF_SIZE) để nhận dữ liệu một lần tối đa BUFF_SIZE = 4096 bytes và dùng vòng lặp cho tới khi nhận đủ. Kết quả trả về là data đã nhận là những bytes đã được nhận (dùng để nhận ảnh trong tính năng screenshot).
- Class ProcessGUI (object):
  - __init__(self, master) để tạo GUI của process gồm các nút kill, xem, xóa, start và bảng nơi hiển thị danh sách các process. Khi đóng cửa sổ GUI process nó sẽ gọi hàm closeProcess.
  - closeProcess(self) dùng để đóng GUI của process và gửi yêu cầu "QUIT" đến server để dừng thao tác process running
  - killProcClick(self), killProcGUI(self) và killProc(self): Khi nhấn nút "Kill" trên processGUI hàm killProc được gọi và sẽ gửi yêu cầu "KILL" cho server và mở GUI của kill để nhập ID của process muốn kill(killProcGUI). Khi nhấn nút "Kill" trong GUI của kill thì hàm killProcClick được gọi và tiến hành lấy ID mới nhập và tiến hành gửi ID đó cho server. Nhận thông báo trả về là '1' thì sẽ xuất hiện hộp thoại đã "diệt process thành công" còn nếu không sẽ trả về là "diệt process không thành công"
  - startProcClick (self),startProcGUI (self) và startProc (self): Khi nhấn nút "Start" trên processGUI hàm startProc được gọi và sẽ gửi yêu cầu "START" cho server và mở GUI của start để nhập ID của process muốn start (startProcGUI). Khi nhấn nút "Start" trong GUI của start thì hàm startProcClick được gọi và tiến hành lấy ID mới nhập và tiến hành gửi ID đó cho server. Nhận thông báo trả về là '1' thì sẽ xuất hiện hộp thoại đã "bật process thành công" còn nếu không sẽ trả về là "bật process không thành công"
  - xoaProc (self): Dùng vòng lặp để xóa lần lượt các trường trong bảng hiển thị các process

        - xemProc (self): Gọi hàm xoaProc(self) để xóa danh sách các process đang có trong bảng và gửi yêu cầu "XEM" đến server rồi nhận dữ liệu từ server. Chuyển dữ liệu vừa nhận bằng hàm receive thành danh sách để chèn vào các trường trong bảng.
    - Class AppGUI (object):
        - __init__(self, master) để tạo GUI của app gồm kill, xem, xóa, start và bảng nơi hiển thị danh sách các app. Khi đóng cửa sổ GUI app nó sẽ gọi hàm closeApp.
        - closeApp (self) dùng để đóng GUI của app và gửi yêu cầu "QUIT" đến server để dừng thao tác app running
        - killAppClick(self), killAppGUI(self) và killApp(self): Khi nhấn nút "Kill" trên appGUI hàm killApp được gọi và sẽ gửi yêu cầu "KILL" cho server và mở GUI của kill để nhập ID của app muốn kill(killAppGUI). Khi nhấn nút "Kill" trong GUI của kill thì hàm killAppClick được gọi và tiến hành lấy ID mới nhập và tiến hành gửi ID đó cho server. Nhận thông báo trả về là '1' thì sẽ xuất hiện hộp thoại đã "diệt app thành công" còn nếu không sẽ trả về là "diệt app không thành công"
        - startAppClick (self),startAppGUI (self) và startApp (self): Khi nhấn nút "Start" trên appGUI hàm startApp được gọi và sẽ gửi yêu cầu "START" cho server và mở GUI của start để nhập ID của app muốn start (startAppGUI). Khi nhấn nút "Start" trong GUI của start thì hàm startAppClick được gọi và tiến hành lấy ID mới nhập và tiến hành gửi ID đó cho server. Nhận thông báo trả về là '1' thì sẽ xuất hiện hộp thoại đã "bật app thành công" còn nếu không sẽ trả về là "bật app không thành công"
        - xoaProc (self): Dùng vòng lặp để xóa lần lượt các trường trong bảng hiển thị các app
        - xemApp (self): Gọi hàm xoaApp (self) để xóa danh sách các app đang có trong bảng và gửi yêu cầu "XEM" đến server rồi nhận dữ liệu từ server. Chuyển

dữ liệu vừa nhận bằng hàm receive thành danh sách để chèn vào các trường trong bảng.

- Class KeystrokeGUI
  - __init__(self, master): Tạo GUI của keystroke gồm các nút hook, unhook, xem, xóa và bảng nơi hiển thị text được ghi nhận. Khi đóng cửa sổ GUI keystroke nó sẽ gọi hàm closeKeystroke.
  - closeKeystroke(self): dùng để đóng GUI của keystroke và gửi yêu cầu "QUIT" đến server để dừng thao tác keystroke
  - xoa(self): Xóa các hiển thị trên nơi hiển thị phím được ghi nhận
  - inPhim(self): Gửi yêu cầu "PRINT" tới server và nhận dữ liệu để hiển thị trên nơi hiển thị phím được ghi nhận
  - unHook(self): Gửi yêu cầu "UNHOOK" lên server và nhận về tín hiệu là được hay không.
  - hook(self): Gửi yêu cầu "HOOK" lên server và nhận về tín hiệu là được hay không.
- Class RegistryGUI
  - __init__(self, master): Tạo GUI cho Registry gồm nút Browser, gởi nội dung, gởi, xóa và các ô để hiển thị link được browser, nội dung của file .reg được chọn, chọn chức năng, chọn kiểu dữ liệu, đường dẫn, name value, value và hiển thị thông tin đã thực hiện lên. Khi đóng cửa sổ GUI registry nó sẽ gọi hàm closeRegistry.
  - closeRegistry(self): dùng để đóng GUI của Registry và gửi yêu cầu "QUIT" đến server để dừng thao tác registry.
  - Option(self, event): Hàm này dùng để xử lý GUI khi chọn chức năng thích ứng xóa và hiển thị các hiển thị cần thiết.
  - Browser(self): Để mở hộp thoại open và chọn file .reg và hiển thị đường link của file được chọn lên nơi hiện đường dẫn và hiển thị nội dung file đó vào nơi hiển thị nội dung

        - goiND(self): Gửi yêu cầu “REG” lên server và tiến hành gửi dữ liệu được đọc từ file .reg mới được browser bằng hàm receive cho server và nhận tín hiệu trả về rồi hiển thị hộp thoại thông báo sửa thành công hay chưa.
        - goi(self), ghiT2(self, s), fixmsg(self, s): Lấy dữ liệu từ các trường chọn nội dung, name value, value, chọn kiểu dữ liệu với các trường không có dữ liệu sẽ dùng hàm fixmsg(self,s) để đưa nó về chuỗi rỗng và tiến hành gửi cho server. Khi gửi xong nhận tín hiệu từ server và ghi lên T2 bằng hàm ghiT2(self, s)
        - xoa(self): Xóa các hiển thị trên nơi hiển thị thông tin đã thực hiện
    - Class PicGUI
        - __init__(self, master): Tạo GUI cho Print Screen gồm các nút take và nút save cùng với hình ảnh được chụp lại màn hình server được hiển thị bởi các thao tác như trong take.
        - closePic(self): dùng để đóng GUI của Pic và gửi yêu cầu “QUIT” đến server để dừng thao tác print screen.
        - save(self): Lấy đường dẫn từ việc chọn trong hộp thoại save as và tiến hành lưu ảnh dưới định dạng .jpg, .png, .*. Khi đã lưu xong tiến hành đóng file lại.
        - take(self): Sẽ gửi yêu cầu “TAKE” đến server và nhận data bằng hàm receive1() (nhận bytes) mở data đó bằng Image.open(BytesIO()) để giải mã data đó thành hình ảnh rồi tiến hành chỉnh kích thước và hiển thị ảnh chụp màn hình của server lên màn hình.
- **Server**
    - sendData(sock, msg): Gửi msg bằng hàm sendall. Data được gửi là các byte được mã hóa “utf8”.
    - recvall: Sử dụng hàm .recv(BUFF_SIZE) để nhận dữ liệu một lần tối đa BUFF_SIZE = 4096 bytes và dùng vòng lặp cho tới khi nhận đủ. Kết quả trả về là data đã nhận phía trên đã được giải mã và xóa đi các kí tự trắng thừa đầu và cuối. Nếu gặp lỗi socket.error thì thoát hàm.

- take: Dùng pyautogui.screenshot() để chụp màn hình server và tiến hành mã hóa thành các bytes bằng BytesIO(), getvalue() và gửi data cuối cùng tới client bằng hàm sendall gửi bytes.
- takepic: Gọi hàm take để bytes hình ảnh cho server và tiếp tục nhận thông điệp nếu là "TAKE" thì sẽ tiếp tục gửi tiếp cho đến khi nhận được thông điệp khác.
- Shutdown: dùng lệnh "shutdown /s" của hệ điều hành, nếu xảy ra lỗi thì gửi tín hiệu "0" cho client để báo thực hiện không thành công.
- BaseRegistryKey: tách base registry key từ đường dẫn bằng cách lấy đoạn đầu tiên trước dấu '\' của đường dẫn và trả về base key đó dưới dạng winreg.HKEY_*Constant.
- GetValue: lấy giá trị của registry value. Dùng hàm baseRegistryKey để trích ra base registry key, phần còn lại của đường dẫn là subkey. Mở key bằng hàm winreg.OpenKey với tham số là base key và subkey. Sau đó dùng hàm winreg.QueryValueEx với tham số là base key và tên value, kết quả trả về là 1 tuple gồm giá trị của registry item và kiểu giá trị. Trích ra phần giá trị để trả về, nếu có lỗi trả về "0".
- SetValue: gán giá trị cho registry item. Dùng hàm baseRegistryKey để trích ra base registry key, phần còn lại của đường dẫn là subkey. Mở key bằng hàm winreg.OpenKey với tham số là base key, subkey, reverse=0 và winreg.KEY_SET_VALUE. Đưa kiểu giá trị về đúng định dạng winreg rồi gọi hàm winreg.SetValueEx, nếu thành công trả về "1", có lỗi trả về "0".
- DeleteValue: xóa giá trị của registry item. Trích base registry key và mở key tương tự các hàm trên. Sau đó gọi hàm winreg.DeleteValue với tham số là base key và tên item cần xóa giá trị. Nếu thành công trả về "1", có lỗi trả về "0".
- DeleteKey: xóa registry key. Trích base key và subkey từ link tới key cần xóa như các hàm trên. Sau đó gọi hàm winreg.DeleteKey với tham số là base key và subkey để xóa. Nếu thành công trả về "1", có lỗi trả về "0".

- CreateKey: tạo key theo đường dẫn. Tương tự Delete key nhưng gọi hàm winreg.CreateKey.
- Registry: xử lý toàn bộ các chức năng của module Registry trong chương trình chính, bao gồm cập nhật registry từ file .reg (nếu tín hiệu là "REG") và cập nhật registry trực tiếp (nếu tín hiệu là "SEND"). Dùng vòng lặp while True để liên tục nhận tín hiệu thực hiện các yêu cầu cho tới khi nhận được tín hiệu "QUIT" thì hàm dừng lại. Ở chức năng cập nhật từ file .reg, cho server mở sẵn 1 file trống là "fileReg.reg" với mod write để ghi lại nội dung file .reg chứa cập nhật nhận được từ client rồi dùng lệnh "reg import fileReg.reg" của hệ điều hành để cập nhật. Ở chức năng cập nhật trực tiếp, nhận thông tin cập nhật từ client bao gồm kiểu cập nhật (option), đường dẫn (link), tên item (valueName) nếu có, giá trị của item (value) nếu có và kiểu giá trị (typeValue) nếu có. Các thông tin trên được client gửi với format tách nhau bởi "\n" nên khi nhận cần tách data bởi "\n" để nhận được giá trị từng thông tin và dùng các hàm createKey, deleteKey, getValue, setValue, deleteValue đã mô tả ở trên để thực hiện từng chức năng. Nếu thực hiện thành công gửi tín hiệu "1" cho client, có lỗi gửi tín hiệu "0".
- PrintKeys: xử lý các kí tự bàn phím đặc biệt như space, tab, backspace, enter,… và gửi toàn bộ kí tự bàn phím ghi nhận được từ tính năng keylog cho client.
- Keylog: thực hiện các chức năng ghi nhận thao tác bàn phím hoặc ngừng ghi nhận sử dụng object Listener của thư viện pynput. Dùng vòng lặp while True để liên tục nhận tín hiệu thực hiện các tính năng như ghi nhận (HOOK), ngừng ghi nhận (UNHOOK), gửi những ghi nhận cho client (PRINT), thoát (QUIT). Dùng biến bool isHook để lưu trạng thái bật tắt của tính năng ghi nhận, mảng keys để lưu những phím đã ghi nhận, object listener để ghi nhận thao tác bàn phím. Khi nhận tín hiệu HOOK, nếu isHook chưa bật thì bật thành True và gọi listener.start để bắt đầu ghi nhận, gửi tín hiệu "1", nếu isHook đã bật trước đó thì không cần làm gì, gửi tín hiệu "0". Tương tự khi nhận tín hiệu UNHOOK, nếu biến isHook

đang bật thì tắt thành False và stop listener. Khi nhận tín hiệu PRINT thì gọi hàm printKeys để xử lý và gửi data ghi nhận được cho client, sau đó clear mảng keys. Khi nhận lệnh QUIT để thoát thì stop listener nếu isHook còn bật.

- Process: Dùng vòng lặp while True để nhận tín hiệu chức năng cần thực hiện cho tới khi thoát (tín hiệu "QUIT"). Ở option "XEM", gọi lệnh "wmic process get Name, processID, threadCount" của hệ điều hành để nhận về các process đang thực thi gồm tên, ID và số thread của process dưới dạng string và gửi cho client. Ở option "KILL", dùng vòng lặp while True để thực hiện đến khi gặp tín hiệu thoát, nhận ID process cần diệt từ client và tìm trong danh sách ID các process đang thực thi, nếu không tồn tại ID gửi "0" cho client, nếu tồn tại thì xóa bằng lệnh "wmic where processID = ID delete" rồi gửi "1", nếu có lỗi gửi "0". Ở option "START", cũng dùng vòng lặp while True để thực hiện đến khi thoát, nhận tên process cần khởi động, gắn thêm đuôi .exe và dùng lệnh "wmic process call create" để khởi động process, nếu thành công gửi "1", có lỗi gửi "0".
- Application: tương tự Process, chỉ khác ở lệnh xem application là "powershell "gps | where {$_.MainWindowTitle } | select name, id, {$_.Threads.Count}".
- buttonServer_click(): Khởi tạo socket với ipv4 và TCP bind tới (HOST, PORT) trong đó HOST là địa chỉ của máy đặt server còn port mặc định là 65432. Mở lắng nghe kết nối tới server .listen(). Khi có yêu cầu kết nối chấp nhận và tiến hành nhận yêu cầu và xử lý các yêu cầu như "KEYLOG", "REGISTRY", "SHUTDOWN", "TAKEPIC", "PROCESS", "APPLICATION", "QUIT" còn các yêu cầu khác thì tiến hành break. Đóng kết nối khi yêu cầu là "QUIT" hoặc khi socket có lỗi.
- run_server(): Tạo GUI của server gồm nút Mở server và khi nhấn vào nút đó sẽ mở tới hàm buttonServer_click().

7. Bảng phân công công việc

STT	Công việc	Người thực hiện

1	Client + GUI client	Thi
2	Server + GUI server	Uyên

8. Tài liệu tham khảo

- Giáo trình mạng máy tính, slide mạng máy tính
- Stack Overflow
- Real Python
- Tkinter tutorial
- ….