**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO ĐỒ ÁN THỰC HÀNH 1**

**LẬP TRÌNH SOCKET**

**KHOA: CÔNG NGHỆ THÔNG TIN**

Thành phố Hồ Chí Minh - 2022

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIsÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO ĐỒ ÁN THỰC HÀNH 1**

**LẬP TRÌNH SOCKET**

**| Giáo viên hướng dẫn |**

**ThS. Nguyễn Thanh Quân**

**| Thành viên nhóm |**

Trần Gia Bảo – 22127034

Phan Văn Tài - 22127372

Thành phố Hồ Chí Minh – 2022

MỤC LỤC

[Email-Client 5](#_Toc152959615)

[I. Thông tin nhóm: 5](#_Toc152959616)

[II. Giới thiệu: 5](#_Toc152959617)

[III. Đánh giá mức độ hoàn thành: 5](#_Toc152959618)

[IV. Kịch bản giao tiếp của chương trình: 6](#_Toc152959619)

[1. Giao thức trao đổi giữa Client và Server, Cấu Trúc Thông Điệp: 6](#_Toc152959620)

[**1.1** **Giao thức SMTP (Sending Email):** 6](#_Toc152959621)

[**1.2** **Giao thức POP3 (Receiving Email):** 7](#_Toc152959622)

[2. Kiểu Dữ Liệu Của Thông Điệp: 7](#_Toc152959623)

[V. Môi trường lập trình và các framework hỗ trợ để thực thi ứng dụng. 8](#_Toc152959624)

[VI. Các tính năng của chương trình: 8](#_Toc152959625)

[1. Gửi Email: 8](#_Toc152959626)

[**1.1** **Gửi Email với TO, CC :** 8](#_Toc152959627)

[**1.2** **Gửi Email với BCC:** 9](#_Toc152959628)

[**1.3** **Gửi attached file:** 9](#_Toc152959629)

[**1.4** **Đọc file json:** 11](#_Toc152959630)

[2 Nhận Email: 11](#_Toc152959631)

[**2.1 Tải Email từ mailbox của Server:** 11](#_Toc152959632)

[**2.2** **Lọc Mail:** 11](#_Toc152959633)

[**2.3** **Quản lý trạng thái Email:** 12](#_Toc152959634)

[**2.4** **Xử lý tự động tải Email** 12](#_Toc152959635)

[3 Tổ chức file config: 12](#_Toc152959636)

[VII. Bảng phân công công việc: 13](#_Toc152959640)

[VIII. Nguồn tài liệu tham khảo: 14](#_Toc152959641)

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành đến Khoa Công nghệ thông tin, Trường Đại học Khoa học Tự Nhiên Thành phố Hồ Chí Minh đã tạo điều kiện thuận lợi cho chúng em học tập và hoàn thành dự án này. Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc đến ThS. Nguyễn Thanh Quân. Thầy đã truyền đạt đầy đủ nội dung liên quan đến đồ án giúp chúng em làm bài

Chúng em đã cố gắng vận dụng những kiến thức đã học được để hoàn thành dự án. Nhưng do kiến thức hạn chế và không có nhiều kinh nghiệm thực tiễn nên khó tránh khỏi những thiếu sót trong quá trình làm bài. Rất kính mong sự góp ý của Thầy để bài dự án của chúng em được hoàn thiện hơn.

Một lần nữa, chúng em xin trân trọng cảm ơn sự quan tâm giúp đỡ của các thầy đã giúp đỡ chúng em trong quá trình thực hiện dự án này.

Xin trân trọng cảm ơn!

Trường Đại học Khoa học Tự nhiên TPHCM Khoa: Công nghệ Thông tin

**Email-Client**

Email Client được làm bởi ngôn ngữ Python | Mạng Máy Tính – HCMUS

1. **Thông tin nhóm:**

|  |  |
| --- | --- |
| Tên thành viên | Mã số sinh viên |
| Phan Văn Tài | 22127372 |
| Trần Gia Bảo | 22127034 |

1. **Giới thiệu:**

Chương trình mô phỏng Email – Client dùng để kết nối đến Mail Server để gửi mail và lấy mail trong hộp thư, trong đó:

* **Người dùng** (máy khách) là một tài khoản email (có thể giả) có chức năng gửi mail và nhận mail. Các thông tin được hiển thị trên màn hình console.
* **Server**: Sử dụng server được cung cấp trong đồ án.

1. **Đánh giá mức độ hoàn thành:**

Chúng em đã cố gắng hoàn thành đầy đủ các yêu cầu của đồ án đúng kì hạn

Những mục đã làm được:

* Gửi được Email ở chế độ TO, CC, BCC riêng lẻ và có thể gửi cả 1 mail ở cả 3 chế độ cùng 1 lần chạy.
* Gửi được các loại attached file khác nhau trong cùng 1 mail và có giới hạn dung lượng gửi file của mỗi mail dưới 3MB.
* Tải được các email ( gồm nội dung và kèm file, lưu được xuống máy cục bộ của client).
* Quản lý được trạng thái của email ( đã đọc / chưa đọc)
* Xử lý lọc email dựa trên các thông tin về người gửi, subject, nội dung. Xử lý đưa từng mail tải xuống vào các folder phân loại, thư mục cụ thể.
* Xử lý tự động tải email từ Mailbox theo thời gian cấu hình trong file config
* Tổ chức file config ở dạng \*.json
* Viết báo cáo tổng kết đồ án.

1. **Kịch bản giao tiếp của chương trình:**
2. **Giao thức trao đổi giữa Client và Server, Cấu Trúc Thông Điệp:**
   1. **Giao thức SMTP (Sending Email):**

* **Client kết nối với Server SMTP:**

Sử dụng ‘**socket’** để thiết lập kết nối TCP với máy chủ SMTP trên cổng giả đã được tạo ra bằng ứng dụng test-mail-server. (VD: 2225)

Server phản hồi với mã trạng thái **“220”** được xem là thành công.

* **Đăng nhập và Xác thực:**

Gửi **‘EHLO’** để bắt đầu phiên giao tiếp.

Gửi **‘MAIL FROM’** với địa chri email người gửi đọc được từ file config

Gửi **‘RCPT TO’** với địa chỉ email người nhận

Gửi **‘DATA’** để bắt đầu quá trình gửi dữ liệu

Gửi nội dung email bao gồm địa chỉ người nhận, người gửi; chủ đề; và phần thân (nội dung email).

Gửi **‘QUIT’** để đóng kết nối đến Server.

* 1. **Giao thức POP3 (Receiving Email):**
* **Client kết nối với Server POP3:**

Sử dụng ‘**socket**’ để thiết lập kết nối TCP với máy chủ POP3 trên cổng được tạo ra (VD: 3335).

Server phản hồi với mã trạng thái **“OK”.**

* **Đăng nhập và Xác thực:**

Gửi **‘USER’** với tên đăng nhập

Gửi **‘PASS’** với mật khẩu

Gửi **‘STAT’** để kiểm tra số lượng email trong hộp thư

* **Lấy danh sách Email:**

Gửi **‘LIST’** để lấy danh sách email và kích thước của chúng.

Gửi **‘UIDL’** để lấy danh sách UID của các email.

* **Lấy Nội dung Email:**

Sử dụng **‘RETR’** để lấy nội dung email theo UID

Lặp lại quá trình này cho tất cả các email chưa được đọc đang có trong mailbox

* **Đánh dấu và Xóa Email:**

Sử dụng **‘DELE’** để đánh dấu các email đã đọc và xóa đi

Gửi **‘QUIT’** để đóng kết nối.

1. **Kiểu Dữ Liệu Của Thông Điệp:**

**Dữ liệu Email (SMTP):** Dạng chuỗi (string), bao gồm địa chỉ người nhận, người gửi, chủ đề, và nội dung. Lúc gửi đi bằng hàm **‘send()**’ của thư viện ‘**socket’** phải chuyển sang dạng binary với format **‘utf8’**.

**Dữ liệu phản hồi từ Server(POP3):** Dạng binary, phải **‘decode()’** sang dạng text bình thường.

**Các danh sách Email** được tải về từ Server bằng giao thức POP3, chúng em lưu từng Mail vào file .txt ở dạng text thông thường.

1. **Môi trường lập trình và các framework hỗ trợ để thực thi ứng dụng.**
2. **Môi trường lập trình:**

Chúng em quyết định chọn ngôn ngữ Python cho đồ án này và thực hiện code trên IDE Visual Studio Code.

1. **Framework:**

Chúng em có tham khảo và code theo form của mẫu được gắn link trong file Project (<https://gaia.cs.umass.edu/kurose_ross/programming/Python_code_only/SMTP_programming_lab_only.pdf>) .

1. **Các tính năng của chương trình:**
2. **Gửi Email:**
   1. **Gửi Email với TO, CC :**

Chức năng nhận mail được code trong file ‘POP3EmailClient.py’. Tuân theo mẫu màn hình console của file Project. Khi người dùng chọn nhận

Chức năng gửi mail được code trong file ‘SMTPEmailClient.py’. Tuân theo mẫu màn hình console của file Project. Đầu tiên chương trình cho người dùng nhập vào lần lượt địa chỉ email muốn gửi đi ở 3 định dạng ‘to’, ‘cc’, ‘bcc’. Ở dạng đơn lẻ, ‘To’ và ‘Cc’ có cách làm tương tự nhau nên chúng tôi xin đề cập về cách gửi ‘To’.

Sau khi lần lượt nhập các định dạng gửi, nhập ‘**Subject’** và **‘Data’** đầy đủ. Chúng tôi tạo một clientSocket kết nối với ‘Mail Server’. Sau đó lần lượt gửi các lệnh **‘EHLO’**, **‘MAIL FROM’**. Đặc biệt với trường hợp người dùng gửi cùng lúc nhiều người. Chúng tôi dùng vòng for() để lấy từng địa chỉ nhập trên và gửi đi với lệnh ‘**RCPT TO’** để bao hết cả trường hợp gửi 1 người và gửi nhiều người. Tiếp đến, dùng tiếp vòng for() ứng với mỗi địa chỉ trong ‘list\_TO’, sẽ gửi đi phần thân mail gồm có ‘To’, ‘From’, ‘Subject’. Chúng tôi sử dụng lệnh **‘clientSocket.send(b'Content-Type:multipart/mixed; boundary="boundary"\r\n\r\n')’** với mục đích xác định gửi email với nhiều phần khác nhau (gửi nội dung, gửi nhiều file đính kèm trong 1 mail):

* + **"Content-Type:multipart/mixed"** chỉ định kiểu nội dung của yêu cầu là "multipart/mixed", tức là yêu cầu có nhiều phần khác nhau được kết hợp trong nội dung.
  + **boundary="boundary"**chỉ định ranh giới (boundary) được sử dụng để phân tách các phần khác nhau trong yêu cầu multipart.
* Với vấn đề gửi nhiều file đính kèm, giữa nội dung email, các file đính kèm sẽ được phân cách bằng dòng lệnh “--boundary”. Và khi gửi hết thông tin thì kết thúc bằng “--boundary--“. Việc này giúp chúng tôi dễ dàng trong việc nhận Mail từ Server về và lọc ra lấy từng nội dung trong phần nhận Mail bằng POP3.
  1. **Gửi Email với BCC:**

Phần gửi mail ở dạng BCC khác với To và Cc, nên chúng tôi sẽ chia ra để nói rõ hơn. Vì gửi Bcc là những người nhận liên quan không biết danh sách những người được gửi. Vì thế khi chỉ gửi riêng ở dạng Bcc, chúng tôi vẫn dùng các lệnh gửi như bình thường trong giao thức. Tuy nhiên ở phần dữ liệu, thay vì gửi ‘To: ‘ + danh sách người được gửi, thì chúng tôi sẽ gửi lệnh ‘To: undisclosed-recipients: ;’ đúng như khi chúng tôi test gửi bằng thunderbird riêng.

Khi Bcc được gửi chung với To hoặc Cc hoặc cả 3 cùng được gửi 1 lúc, thì ở phần hiển thị danh sách người nhận sẽ không hiện ‘To: undisclosed-recipients: ;’ nữa mà sẽ hiện danh sách ‘To’ hoặc ‘Cc’ đã được gửi.

* 1. **Gửi attached file:**

Sau khi gửi nội dung email, tiếp đến là gửi các file đính kèm (\*.txt, \*.docx, \*.pdf, ….). Biến attachtment\_path vừa dùng để lưu các đường dẫn file vừa dùng để kiểm tra xem người dùng có yêu cầu gửi file đính kèm không. Việc gửi file được gói gọn trong hàm send\_File. Lệnh **os.path.exists(attachment\_path)** để kiểm tra đường dẫn file có tồn tại không. **os.path.getsize(attachment\_path**) để lấy dung lượng file dưới dạng Byte rồi lưu vào biến size. Biến size lưu trữ toàn bộ dung lượng các file gửi nếu vượt quá giới hạn MAX\_SIZE thì không gửi.

Chúng tôi dùng lệnh **os.path.basename** để lưu tên file vào biến attachment\_name. Lệnh **with open(attachment\_path, 'rb') as attachment\_file** để mở file đính kèm ở chế độ đọc nhị phân ('rb' - read binary). Sử dụng with để đảm bảo rằng file sẽ được đóng tự động sau khi hoàn thành công việc. Dữ liệu file được lưu vào biến attachment\_data sau đó mã hoá dữ liệu bằng base64 rồi lưu vào biến encoded\_attachment.

Để xác định loại file (.txt, .pdf, .jpg, .zip,...), chúng tôi kiểm tra 3 kí tự cuối của attachment\_name. **'--boundary\r\n'** đểxác định đầu file, Để gửi các loại file khác nhau phải dùng các Content-Type tương ứng:

**'Content-Type:application/octet-stream; name="{attachment\_name}"\r\n' :** dùng chofile txt

**'Content-Type:application/pdf; name="{attachment\_name}"\r\n'** : dùng cho file pdf

**'Content-Type:application/msword; name="{attachment\_name}"\r\n'** : dùng cho file doc

**'Content-Type:image/jpeg; name="{attachment\_name}"\r\n'** : dùng cho file jpg

**'Content-Type:application/zip; name="{attachment\_name}"\r\n'** : dùng cho file zip

**'Content-Disposition:attachment; filename="{attachment\_name}"\r\n'**: để thông báo rằng phần nội dung được đính kèm và không phải là một phần của nội dung chính của email. Nó định rõ mục đích của phần đính kèm là để người nhận có thể tải về và lưu trữ file.

**'Content-Transfer-Encoding: base64\r\n\r\n'**: cho biết rằng dữ liệu đính kèm đã được mã hóa bằng phương pháp mã hóa Base64. Trong đồ án này, base64 được dùng với 2 mục đích:

* 1. Dùng để gửi mail có chứa file đính kèm: base64 mở file ra và đọc hết dữ liệu trong file đính kèm ở dạng byte và đổi sang string trước khi được đưa vào gói gửi lên đường truyền 🡪 gửi mail.
  2. Dùng để tải file đính kèm trong mail về máy cục bộ của client: Vì đã đọc file bằng base64, nên lúc tải file về chúng tôi cũng dùng base64 cuối cùng để bỏ hết dữ liệu file và chuyển đổi lại về dạng byte bằng base64. Ngoài ra các thao tác đọc và viết file khác chỉ ở dạng ‘read’ và ‘write’ string bình thường, không liên quan đến việc mã hoá string thành byte và ngược lại.

Vì nội dung dữ liệu file quá lớn, lệnh **‘send()’** của thư viện **‘socket’** không thể gửi hết 1 lần trên 1 dòng. Vì thế chúng tôi đã dùng thủ thuật chia nhỏ dữ liệu truyền thành các phần nhỏ. Biến chunk\_size đặt giá trị cho độ dài tối đa của mỗi dòng. Trong giao thức SMTP, nó thường được chọn để không vượt quá **76 ký tự** để đảm bảo tính tương thích với các máy chủ email và tránh trục trặc trong quá trình truyền tải.

Chúng tôi dùng vòng lặp for chạy qua từng phần của **encoded\_attachment**, bắt đầu từ 0 và mỗi lần tăng chunk\_size cho đến khi hết dữ liệu. **clientSocket.send(encoded\_attachment[i:i+chunk\_size].encode(FORMAT) + b'\r\n')** : Lệnh này để gửi một phần nhỏ của dữ liệu đính kèm, có độ dài là chunk\_size.

* 1. **Đọc file json:**

Ngoài ra, chúng tôi còn thiết kế hàm **readJson().** Hàm này lấy các thông tin từ file json như tên người dùng, mật khẩu, địa chỉ mail server, port smtp, port pop3, thời gian autoload và các từ khoá để lọc mail vào thư mục

1. **Nhận Email:**

**2.1 Tải Email từ mailbox của Server:**

Chức năng nhận mail được code trong file ‘POP3EmailClient.py’. Tuân theo mẫu màn hình console của file Project. Khi người dùng chọn nhận mail, chúng tôi tạo một clientSocket kết nối với ‘Mail Server’. Tiếp đến gửi các lệnh ‘USER’, ‘PASS’ để xác thực người dùng. Sau đó gửi lệnh ‘STAT’ để lấy số byte có trong mail, lệnh ‘LIST’ để lấy danh sách mail, ‘UIDL’ để xác định email duy nhất (nhận biết các email mới). Chúng tôi dùng lệnh **response.count('.msg')** để lấy số lượng mail nhận được. Sau đó chạy for() gửi lệnh ‘RETR’ để lấy email theo thứ tự. Vì không thể lấy hết dữ liệu 1 lần trên đường truyền nên chúng tôi dùng vòng lặp while để lấy từng phần dữ liệu rồi lưu vào biến response. Sau đó lọc ra người gửi, tiêu đề dựa vào từ khoá ‘From:’, ‘Subject:’. Sau đó chuyển mail vào các thư mục tương ứng thông qua các từ khoá của file json cung cấp.

Chúng tôi dùng ‘boundary’ để phân chia các phần trong việc gửi nhiều file đính kèm trong 1 mail, vì thế khi nhận mail cũng dùng nó để nhận biết mail có bao nhiêu file đính kèm và lọc dữ liệu từng file khác nhau.

* 1. **Lọc Mail:**

Sau khi tải mail và đưa vào các thư mục tương ứng. Chúng tôi cung cấp giao diện cho người dùng lựa chọn. Có 5 loại thư mục chứa mail: Inbox, Project, Important, Work, Spam. Sau khi người dùng chọn thư mục muốn đọc, chúng tôi lần lượt lưu mail vào biến data và trích lọc ra người gửi, tiêu đề rồi lưu vào danh sách list\_data, list\_from, list\_sub.

Giao thức gửi của chúng tôi và giao thức gửi của thunderbird khác nhau. Vì thế nên khúc lọc mail của 2 loại cũng khác nhau.

Với thunderbird có sử dụng **‘MIME-Version’** còn chúng tôi không dùng nên cách sử lí để lưu list\_subject 2 loại sẽ có đôi chút khác nhau.

Sau đó dùng các list\_subject, list\_from, response để tìm các đặc điểm phù hợp với các đặc điểm có trong file config 🡪 chọn ra loại thư mục phù hợp lưu vào.

* 1. **Quản lý trạng thái Email:**

Với việc đánh dấu trạng thái ‘đã đọc’ và chưa đọc’ của email trong hộp thư folder. Chúng tôi đề xuất giải pháp khi tải Mail về từ mailbox, chúng tôi sẽ chèn thêm ‘chuadoc’ vào cuối của mail vào lúc tải về lọc mail. Đến khi mở mail ra đọc xong thì chúng tôi sẽ mở file ra, viết lại dữ liệu vào file và replace lại **‘chuadoc’** thành **‘dadoc’**. Và dùng hàm ‘check\_Readed()’ để kiểm tra xem 7 kí tự cuối cùng có phải ‘chuadoc’ không, nếu phải thì return 1 🡪 chưa đọc, ngược lại là 0 🡪 đã đọc

* 1. **Xử lý tự động tải Email**

Chúng tôi giải quyết vấn đề này ở **‘fetch\_emails()’** và **‘auto\_fetch\_emails()’** trong file **‘main.py’**. Dùng biến stop\_fetching = False và cho hàm **‘auto\_fetch\_emails()**’ chạy liên tục cho đến khi biến stop\_fetching = **True**.

Vì trình biên dịch đọc và thực thi code theo thứ tự từ trên xuống nên không thể giải quyết lập trình song song bình thường. Vì thế chúng tôi sử dụng thêm thư viện **‘threading’** của python để lập trình đa luồng. Chúng tôi sẽ cho hàm **‘auto\_fetch\_emails()’** chạy liên tục và song song với hàm **‘While True:’** trong suốt quá trình chạy chương trình chính.

1. **Tổ chức file config:**

Ban đầu chúng tôi sử dụng file config dạng \*.txt, sau khi hoàn thành xong bắt đầu tìm hiểu và chuyển sang file config dạng \*.json. Nhưng cách đọc và tìm ra các thành phần như Username, Password, MailServer, … hầu như tương đồng nhau.

Chúng tôi mở file config, đọc file và bỏ hết dữ liệu vào một biến content dạng string. Sau đó tạo các biến vị trí và lưu vị trí đầu, vị trí cuối của các thành phần trên và cuối cùng lưu vào biến cần dùng với chuỗi con của content từ vị trí bắt đầu đến vị trí kết thúc.

Đặc biệt ở file .json, kí tự “ khi tìm bằng hàm find() không được mà phải ở dạng \”. Điểm này khác biệt so với file dạng .txt khi đọc lên.

1. **Bảng phân công công việc:**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Yêu cầu** | **Người thực hiện** | **Mức độ hoàn thành (Do người còn lại đánh giá)** |
| 1 | Gửi Email với To và CC | Bảo | Hoàn thành tốt, đúng thời hạn |
| 2 | Gửi Email với BCC | Bảo | Hoàn thành tốt, đúng thời hạn |
| 3 | Gửi attached file | Bảo | Hoàn thành tốt, đúng thời hạn |
| 4 | Tải emal về mail client từ Mailbox của Server | Tài | Hoàn thành tốt, đúng thời hạn |
| 5 | Quản lý trạng thái email | Tài | Hoàn thành tốt, đúng thời hạn |
| 6 | Xử lý lọc email | Tài | Hoàn thành tốt, đúng thời hạn |
| 7 | Xử lý tự động tải email từ Mailbox theo thời gian cấu hình của config file | Tài | Hoàn thành tốt, đúng thời hạn |
| 8 | Tổ chức file config | Bảo | Hoàn thành tốt, đúng thời hạn |
| 9 | Báo cáo | Bảo + Tài | Hoàn thành tốt, đúng thời hạn |

1. **Nguồn tài liệu tham khảo:**
2. Module : <https://topdev.vn/blog/module-trong-python/>
3. Thao tác với folder trong python : <https://gochocit.com/ngon-ngu-lap-trinh/cac-thao-tac-tren-thu-muc-directory-voi-python#3-tao-create-mot-thu-muc-moi-trong-python>
4. Lập trình đa luồng python: <https://howkteam.vn/course/python-nang-cao/da-luong-multithreading-trong-python-4423>
5. Các lệnh POP3: <https://123docz.net/trich-doan/2453594-cac-lenh-cua-pop3.htm>
6. Thao tác với folder python: <https://www.geeksforgeeks.org/python-os-path-exists-method/>
7. Chat-GPT : <https://openai.com/blog/chatgpt>, <https://poe.com/ChatGPT>

ﺺﺺﺺﺺﺺﺺﺺHẾT ﺺﺺﺺﺺﺺﺺﺺ