

XÂY DỰNG MÔ HÌNH TẬP TIN  
VÀ THIẾT KẾ KIẾN TRÚC TỔ CHỨC CHO VOLUME FILE

Thành viên	MSSV	Tỉ lệ tham gia (đơn vị %)
Nguyễn Hoàng Nhân	18127017	33.33
Kiều Công Hậu	18127259	33.33
Trần Thanh Tâm	18127268	33.33

STT	Công việc		Phụ trách
1	Code	Bàn bạc và thiết kế kiến trúc cho volume file.	Cả nhóm
2		Tạo mới một volume.	Kiều Công Hậu
3		Mở một volume (Đọc nội dung của volume file và lưu vào các cấu trúc dữ liệu trên RAM).	Kiều Công Hậu
4		Xóa một file/folder ra khỏi volume.	Kiều Công Hậu
5		Thiết kế giao diện.	Trần Thanh Tâm
6		Hiển thị danh sách các file/folder trong volume.	Trần Thanh Tâm
7		Cài đặt/Hủy mật khẩu truy xuất cho một file/folder trong volume.	Trần Thanh Tâm
8		Chép một file/folder trong volume ra ngoài (export).	Nguyễn Hoàng Nhân
9		Chép một file/folder từ bên ngoài vào volume (import).	Nguyễn Hoàng Nhân
10	Bảo cáo	Ghép nội dung và format.	Trần Thanh Tâm
11		Trình bày thuật toán cài đặt mật khẩu truy xuất cho một file/folder trong volume.	Trần Thanh Tâm
12		Trình bày thuật toán tạo mới một volume.	Kiều Công Hậu
13		Trình bày thuật toán mở một volume (đọc nội dung của volume file và lưu vào các cấu trúc dữ liệu trên RAM).	Kiều Công Hậu
14		Trình bày thuật toán xóa một file/folder trong volume.	Kiều Công Hậu
15		Trình bày thuật toán chép một file/folder từ ngoài vào volume (import).	Nguyễn Hoàng Nhân
16		Trình bày thuật toán chép một file/folder từ volume ra ngoài (export).	Nguyễn Hoàng Nhân
17		Trình bày phần giới thiệu chi tiết về kiến trúc NHT.	Cả nhóm

Tự đánh giá:

Nhóm đã hoàn thành tốt các mục tiêu đề ra ban đầu. Trong quá trình làm việc, mỗi thành viên đều chăm chỉ, nhiệt huyết, hoàn thành tốt nhiệm vụ của mỗi cá nhân và hỗ trợ lẫn nhau.

Lấy ý tưởng dựa trên ZIP và tinh giản một số thông tin, nhóm em lên ý tưởng và thiết kế kiểu dữ liệu với tên gọi là NHT. NHT gồm 3 vùng chính lần lượt như sau:

[DATA]

[ENTRY TABLE]

[VOLUME INFO]

## 1. Vùng [VOLUME INFO]

[VOLUME INFO] là vùng chứa những thông tin quan trọng của Volume, được đặt ở cuối Volume. Khi mở một Volume, chương trình sẽ đọc nội dung của vùng [VOLUME INFO] trước nhằm lấy các thông tin quan trọng của Volume.

Offset	Độ lớn (byte)	Nội dung
0	4	Chữ kí.
4	4	Kích thước của vùng [ENTRY TABLE] (đơn vị là byte).
8	4	Vị trí bắt đầu của vùng [ENTRY TABLE] – tính từ đầu Volume.

Chữ ký:

- Giá trị mặc định là 0x0254484E.
- Được dùng để làm dấu hiệu nhận biết Volume File có kiến trúc NHT.

Kích thước của vùng [ENTRY TABLE] và Vị trí bắt đầu của vùng [ENTRY TABLE]:

- Được dùng để xác định vị trí của vùng [ENTRY TABLE].
- Có được 2 thông tin này, ta tiếp tục tiến hành truy xuất và lấy thông tin từ vùng [ENTRY TABLE].

## 2. Vùng [ENTRY TABLE]

[ENTRY TABLE] là vùng được dùng để lưu danh sách các ENTRY liên tiếp nhau. Trong đó, mỗi ENTRY đóng vai trò quản lý những thông tin quan trọng của một File hoặc một Folder có trong Volume. Cấu trúc của một ENTRY như sau:

Offset	Độ lớn (byte)	Nội dung
0	2	Thời gian chỉnh sửa lần cuối.
2	2	Ngày chỉnh sửa lần cuối.
4	4	Kích thước dữ liệu file/folder thuộc vùng [DATA] của file.
8	2	Độ dài tên file/folder (n).
10	2	Độ dài password (m).
12	4	Vị trí của dữ liệu – tính từ đầu Volume.
16	n	Tên của file/folder.
16 + n	m	Password truy xuất của file/folder.

**\*Thời gian chỉnh sửa lần cuối:**

- Dùng 16 bit để lưu trữ thuộc tính này (Giờ, Phút, Giây).
- Giờ = <5 bit cao nhất>.
- Giây = <5 bit thấp nhất> \* 2.
- Phút = <6 bit còn lại ở giữa>.

**Nhận xét:**

- Giá trị cao nhất của Giờ là 23 nên cần tối thiểu 5 bit để biểu diễn.
- Giá trị cao nhất của Phút là 59 nên cần tối thiểu 6 bit để biểu diễn.
- Giá trị cao nhất của Giây là 59 nên cần tối thiểu 6 bit để biểu diễn.
- Tuy nhiên, sau khi biểu diễn Giờ và Phút thì chỉ còn vón vẹn 5 bit để biểu diễn Giây, nên ta buộc phải chia đôi Giây ra thì mới lưu được vào 5 bit này. Điều này dẫn tới sai số ở phần Giây khi ta đọc ngược lại giá trị từ 5 bit này rồi nhân cho 2. Nhưng sai số này là không đáng kể vì chỉ chênh lệch tối đa là 1s nên vẫn chấp nhận được. Đồng thời điều này lại còn giúp tiết kiệm bộ nhớ, vì nếu muốn biểu diễn chính xác thuộc tính này có thể sẽ cần tận 32 bit (gấp đôi).

**\*Ngày chỉnh sửa lần cuối:**

- Dùng 16 bit để lưu trữ thuộc tính này (Năm, Tháng, Ngày).
- Năm = <7 bit cao nhất> + 1980.
- Ngày = <5 bit thấp nhất>.
- Tháng = <4 bit còn lại ở giữa>.

**Nhận xét:**

- Đối với Năm, 7 bit chỉ biểu diễn được các giá trị nguyên từ 0 đến 127, nên hệ thống này chỉ có thể xài được từ năm 1980 đến năm 2107 (hiện tại là năm 2020 nên vẫn còn rất xa, có thể đến lúc này kiến trúc này đã lỗi thời và có nhiều kiến trúc tiên tiến khác thay thế nên vẫn hợp lý, lại còn giúp tiết kiệm bộ nhớ).

**\*Kích thước dữ liệu của File:**

- Dùng 32 bit để biểu diễn thuộc tính này, đồng nghĩa với việc chỉ lưu được những File có kích thước khoảng 4GB (hợp lý vì kích thước của 1 File thường không bao giờ lớn đến như vậy).
- Điều đặc biệt là: nếu ENTRY này quản lý Folder thì trường này có giá trị bằng 0. Vì 1 Folder thì chứa rất nhiều Folder con và các File, nên Kích thước của Folder này sẽ bằng tổng Kích thước dữ liệu của các Folder con và File nằm bên trong nó, điều này dẫn tới Kích thước Folder sẽ vô cùng lớn, vượt 4GB là điều bình thường. Bên cạnh đó, việc lưu trữ Kích thước dữ liệu của Folder cũng không mang lại quá nhiều lợi ích đáng kể.

**\*Vị trí của dữ liệu (Offset của vùng dữ liệu của một File):**

- Thông số này kết hợp với Kích thước dữ liệu của File sẽ giúp ta truy xuất đến nội dung của File trong vùng [DATA].

**\*Độ dài tên File:**

- Dùng để lưu lại độ dài của tên file/folder, phục vụ cho việc truy xuất nội dung của trường dữ liệu “Tên của file/folder” tại offset 16.

**\*Độ dài của mật khẩu:**

- Dùng để lưu lại độ dài của mật khẩu, phục vụ việc truy xuất nội dung của trường dữ liệu “Mật khẩu truy xuất của file/folder”.

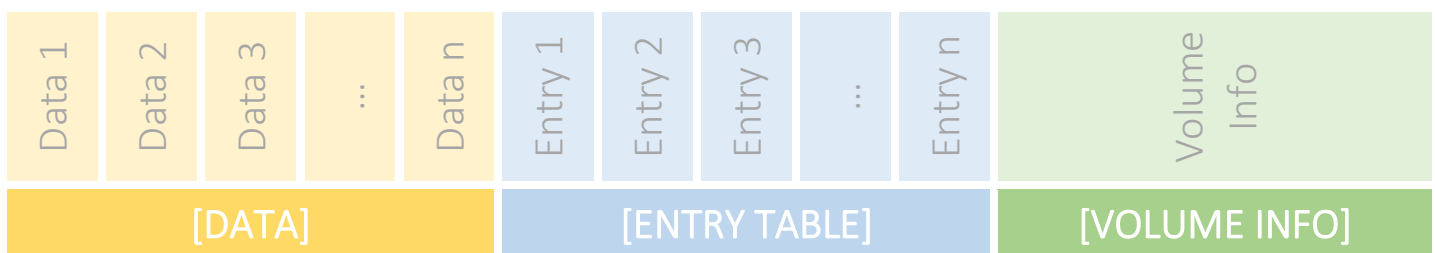
**\*Tên của file/folder:**

- Độ dài của tên là giá trị  $n$ , mà  $n$  được lưu trữ bằng 16 bit tại offset 8 nên độ dài tối đa của file/folder sẽ là 65535.
- Cách lưu tên trong kiến trúc này khá đặc biệt vì nó lưu luôn cả đường dẫn của nó.
- Ví dụ:
  - File 'abc.txt' được lưu trong Folder 'My Folder' thì tên của nó sẽ là: 'My Folder/abc.txt'.
  - Folder 'HCMUS' được lưu trong Folder 'Learning' thì tên của nó sẽ là: 'Learning/HCMUS/'.
- Như vậy, đối với folder thì tên của nó luôn kết thúc với dấu '/', còn file thì không. Cách lưu này ta giúp ta biết folder nào là cha của file/folder nào, giúp cho việc đọc Volume File và lưu vào cấu trúc dữ liệu Cây Thư Mục trên RAM dễ dàng.

**\*Mật khẩu truy xuất file / folder:**

- Độ dài của mật khẩu là giá trị  $m$ , mà  $m$  được lưu trữ bằng 16 bit tại offset 10 nên độ dài tối đa mật khẩu sẽ là 65535.
- Hiện tại, kiến trúc này đang sử dụng kỹ thuật SHA-256 để băm mật khẩu và lưu vào trường dữ liệu này nên  $m$  luôn bằng một giá trị không đổi 256.
- Sau này, nếu có đổi kỹ thuật bảo mật thì  $m$  có thể sẽ là một giá trị không cụ thể nào đó.

**\*Mối quan hệ giữa hai vùng [ENTRY TABLE] và [DATA]:**



Nhờ vùng [ENTRY TABLE] đóng vai trò quản lý, ta có thể dễ dàng truy xuất đến nội dung của các file/folder được lưu trữ trong vùng [DATA] của Volume File này thông qua các thông số quan trọng của mỗi Entry vừa được nêu trên. Cụ thể, Entry 1 sẽ quản lý Data 1, Entry 2 quản lý Data 2, Entry  $i$  quản lý Data  $i$  ( $i = 0, 1, 2, \dots, n$ ).

**Lưu ý:**

- Các Entry được lưu trong vùng [ENTRY TABLE] này luôn được sắp xếp theo một thứ tự có quy luật như sau: Entry quản lý một file/folder con thì luôn đứng sau Entry quản lý Folder cha của nó (đứng sau ở đây không phải là liền kề sau) vì đặc thù của cách lưu "Tên của file/folder" trong Volume File.

### 3. Vùng [DATA]

Là vùng chỉ chứa dữ liệu của các file trong volume theo thứ tự file được import vào trước thì sẽ đứng trước, file/folder vào sau sẽ đứng sau.

Dữ liệu giữa các file sẽ liên tiếp nhau mà không có bất kỳ một kí tự nào để đánh dấu vị trí bắt đầu hay kết thúc của dữ liệu file.

## 1. Tổng quan

Đầu vào Tên đường dẫn chứa volume.

Đầu ra Volume file chưa có dữ liệu tập tin hay thư mục nào.

## 2. Thuật toán

**Bước 1** Nhập đường dẫn nơi chứa volume file.

**Bước 2** Kiểm tra đường dẫn này có hợp lệ hay không. (Không hợp lệ: tên volume bị trùng, đường dẫn không tồn tại hoặc đường dẫn sai cú pháp). Nếu hợp lệ, ta tiến tới Bước 3, ngược lại ta tiến tới Bước 4.

**Bước 3** Chép vào volume file trống nội dung của vùng VOLUME INFO.

- Định danh = 0x0254484E.
- Kích thước vùng ENTRY TABLE = 0.
- Offset bắt đầu của vùng ENTRY TABLE = 0.

**Bước 4** Kết thúc chức năng và in ra thông báo phù hợp.

## 3. Nhận xét

Sau khi tạo mới một volume, volume file này chỉ chiếm 12 byte (kích thước của vùng VOLUME INFO).

Mọi volume file theo định dạng này có phần mở rộng là “.nht”.

## 1. Tổng quan:

Đầu vào Đường dẫn của volume mà ta mong muốn mở.

Đầu ra Mở được volume và cho phép thực hiện các thao tác chức năng trên volume này (xem danh sách các file/folder trong volume, chép một file/folder vào volume (import), chép một file/folder ra khỏi volume (export), xóa một file/folder, đặt/hủy mật khẩu cho một file/foder trong volume,...)

## 2. Ý tưởng:

Để thuận tiện cho việc mở một volume file đã được tạo hoặc được mở trước đó, thay vì nhập toàn bộ đường dẫn dài của volume file, phần mềm có hỗ trợ hệ thống lưu trữ Cache để lưu lại những volume file mà ta đã từng tạo hoặc mở trước đó. Điều này giúp cho việc mở volume nhanh hơn rất nhiều và giúp ta quản lý được những volume file nào hiện đang có trên máy tính của chúng ta.

### 3. Thuật toán:

- Bước 1** Nhập đường dẫn của volume file mà ta muốn mở.
- Bước 2** Kiểm tra xem đường dẫn này có hợp lệ hay không. Nếu không thì kết thúc chức năng mở volume, nếu hợp lệ thì tới Bước 3. Lưu ý: Nếu ta mở một volume trong Cache mà Volume này đã bị xóa trước đây thì Cache sẽ được cập nhật lại (xóa đường dẫn này ra khỏi Cache).
- Bước 3** Kiểm tra xem đây có phải là Volume File của định dạng NHT hay không. Để kiểm tra được, ta sẽ đọc nội dung vùng VOLUME INFO bằng cách mở Volume File này lên và dời con trỏ đọc file về cuối Volume File, sau đó dời ngược về lại 12 byte (kích thước của vùng VOLUME INFO). Sau khi đọc nội dung cho vùng Volume Info, ta sẽ kiểm tra Định danh có chính xác hay không. Nếu không thì kết thúc chức năng mở volume, nếu hợp lệ thì tới Bước 4.
- Bước 4** Kiểm tra xem đường dẫn này có tồn tại trong Cache hay chưa. Nếu đã có rồi thì thôi, nếu chưa có thì thêm đường dẫn của volume file này vào Cache.
- Bước 5** Đọc nội dung của vùng VOLUME INFO để lấy các thông tin cơ bản của Volume.
- Bước 6** Nhờ những thông tin của VOLUME INFO, ta sẽ đọc được tiếp nội dung của toàn bộ các Entry trong vùng ENTRY TABLE. Ta sẽ đọc tuần tự từng Entry một, đọc tới đâu ta sẽ thêm Entry này vào RAM tới đó bằng cách tìm folder cha của nó đã được lưu trên RAM trước đó. Sau khi đọc hết tất cả các Entry trong ENTRY TABLE và lưu vào RAM, ta có thể thực hiện các chức năng thao tác trên Volume này được rồi.

Lưu ý:

- Trên RAM, các Entry được lưu theo cấu trúc dữ liệu cây thư mục để phục vụ việc truy xuất các file/folder dễ dàng theo đúng mối quan hệ cha – con.
- Bên cạnh đó, còn có một cấu trúc dữ liệu danh sách liên kết (vector) có các phần tử là con trỏ trỏ đến vùng dữ liệu của mỗi Entry trên cây thư mục nhằm lưu trữ đúng vị trí thứ tự của các Entry trong bảng ENTRY TABLE, mục đích để phục vụ thao tác dời và ghi lại dữ liệu của ENTRY TABLE xuống Volume File.

## CHỨC NĂNG IMPORT FILE/FOLDER VÀO VOLUME **Phần IV**

### 1. Tổng quát

**Đầu vào** Đường dẫn đến 1 file/folder cần được import; 1 volume có sẵn.

**Đầu ra** Volume đã được import file/folder đó.

### 2. Ý tưởng

Để import 1 file, ta sẽ chép dữ liệu của file vào volume và lưu ở cuối vùng [DATA]; sau đó tạo 1 entry ở vùng [ENTRY TABLE] để quản lý thông tin của file như tên, kích thước, ngày giờ chỉnh sửa gần nhất,...; cuối cùng ta cập nhật thông tin của volume ở vùng [VOLUME INFO].

Còn để import 1 folder, ta sẽ không có thao tác chép dữ liệu, vì folder không chứa dữ liệu. Ta chỉ thực hiện thao tác tạo entry cho folder và cập nhật vùng [VOLUME INFO].

Tuy nhiên, do folder có thể chứa các file và folder con bên trong, nên ta cũng phải import luôn những file và folder con này vào volume. Vì vậy quá trình import 1 folder thực chất bao gồm việc import chính folder đó và kèm theo việc import tất cả những file và folder con chứa bên trong folder đó.

Vì mô hình folder có cấu trúc cây, nên ta có thể dùng một thuật toán duyệt cây để duyệt và import từng file và folder con của folder được chỉ định. Ở đây, em sẽ sử dụng thuật toán “level order traversal” (duyet theo level) vì thuật toán này không sử dụng đệ quy, và vì thế tiết kiệm bộ nhớ hơn so với các thuật toán sử dụng đệ quy – vốn sử dụng khá nhiều bộ nhớ khi được gọi đệ quy nhiều lần.

### 3. Thuật toán

**Bước 1** Nếu ta import 1 file, đến bước 2; còn nếu ta import 1 folder, đến bước 3.

**Bước 2** Trích xuất thông tin của file và tạo 1 entry mới ở cuối vùng [ENTRY TABLE] của volume để lưu và quản lí các thông tin đó; sau đó ta chép dữ liệu của file và lưu ở cuối vùng [DATA] của volume. Rồi ta nhảy đến bước 4.

**Bước 3** Gọi T là cây thư mục có root là folder có đường dẫn do ta nhập vào (hay nói cách khác, T là cây thư mục của folder ta muốn import). Ta sử dụng thuật toán “level order traversal” để duyệt qua tất cả các file và folder trong cây T. Mỗi lần truy cập một file/folder, ta sẽ lấy thông tin của file/folder đó và lưu vào 1 entry trong vùng [ENTRY TABLE], đồng thời nếu nó là file chứa dữ liệu (tức không phải folder) thì ta sẽ chép dữ liệu của nó và lưu ở cuối vùng [DATA] của volume. Và tất nhiên nếu ta có ghi dữ liệu nào đó (dữ liệu này là nói chung cho entry của file/folder và dữ liệu file) vào vị trí ở giữa volume, ta sẽ phải dời tất cả dữ liệu nằm ở phía sau nó về phía sau.

**Bước 3.1** Tạo 1 cấu trúc queue, gọi là Q. Queue này sẽ được dùng để lưu đường dẫn đến các file và folder trong cây T.

**Bước 3.2** Push đường dẫn của root folder trong cây T vào đuôi queue Q.

**Bước 3.3** Kiểm tra xem queue Q có rỗng hay không. Nếu rỗng, nhảy đến bước 4. Nếu không rỗng, tiếp tục đến bước 3.4.

**Bước 3.4** Trích xuất đường dẫn nằm ở đầu queue Q (ta gọi file/folder có đường dẫn này là cur\_file), sau đó xóa (pop) đường dẫn đó khỏi queue.

**Bước 3.5** Dùng đường dẫn này để truy cập cur\_file. Ta sẽ lấy thông tin của cur\_file và lưu vào 1 entry mới ở cuối vùng [ENTRY TABLE] của volume. Sau đó ta dời toàn bộ vùng [VOLUME INFO] về phía sau một đoạn bằng với kích thước entry mới được tạo.

**Bước 3.6** Nếu cur\_file là 1 file bình thường (không phải folder), đến bước 3.7; còn nếu cur\_file là 1 folder, đến bước 3.8.

**Bước 3.7** Chép dữ liệu của file và lưu ở cuối vùng [DATA] của volume. Ta cũng sẽ dời toàn bộ vùng [ENTRY TABLE] và [VOLUME INFO] về phía sau một đoạn bằng với kích thước khối dữ liệu ta mới chép vào. Sau đó ta quay lại bước 3.3.

**Bước 3.8** Lấy đường dẫn của tất cả các file và folder con chứa trong cur\_file và push vào đuôi queue Q. Sau đó quay lại bước 3.3.

**Bước 4** Cập nhật lại thông tin về kích thước và offset của vùng [ENTRY TABLE] lưu trong vùng [VOLUME INFO].

#### 4. Nhận xét

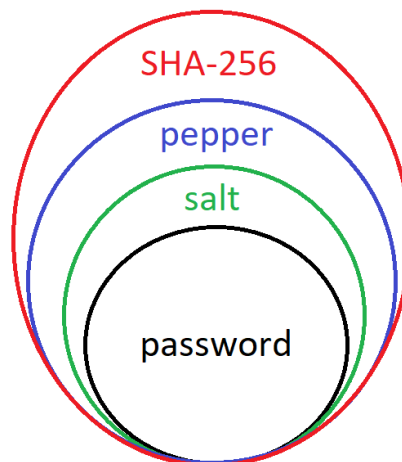
Ở thuật toán này, chỗ phức tạp nhất chính là khi import 1 folder, ta cũng phải duyệt toàn bộ cây thư mục của folder đó để import tất cả file và folder con của folder đó. Và em đã lựa chọn thuật toán “level order traversal” để duyệt cây thư mục này, vì như em đã nói ở trên, đây là một thuật toán không sử dụng đệ quy. Nếu sử dụng các thuật toán đệ quy để duyệt cây, việc thực hiện sẽ chiếm khá nhiều bộ nhớ khi cây thư mục có nhiều level. Trong khi nếu sử dụng thuật toán không đệ quy như “level order traversal”, ta sẽ tránh gặp phải vấn đề này, mặc dù ta cũng phải đánh đổi là việc đọc hiểu code cũng sẽ khó khăn hơn, nhất là đối với những ai chưa biết đến thuật toán này.

Trên đây chỉ là thuật toán lí thuyết mà em nghĩ ra ban đầu. Trong thực tế, để tránh việc dời dữ liệu trong volume nhiều lần và tránh việc di chuyển tới lui đến nhiều vị trí cách xa nhau trong volume để ghi dữ liệu, khi em viết code, em đã thay đổi thứ tự một số bước trong thuật toán để cho phù hợp hơn, mà không làm thay đổi bản chất của thuật toán. Cụ thể, để tránh việc di chuyển tới lui nhiều lần trong volume thì trong code của em, đối với trường hợp import 1 folder, em sẽ tạo 1 lần tất cả những entry mới cho tất cả các file và folder của cây thư mục trước, rồi mới chép dữ liệu của tất cả các file vào volume sau. Còn để tránh việc dời dữ liệu trong volume nhiều lần, thay vì em dời vùng [ENTRY TABLE] và [VOLUME INFO] sau mỗi lần chép dữ liệu của 1 file mới vào vùng [DATA], thì bắt đầu từ vị trí đuôi của vùng [DATA] ban đầu, em sẽ ghi luôn 1 lần nguyên 1 khối dữ liệu bao gồm: dữ liệu của những file cần được import (tức là dữ liệu này sẽ nằm ở cuối vùng [DATA] mới), vùng [ENTRY TABLE] (có chứa entry của những file/ folder cần được import), và vùng [VOLUME INFO] (đã cập nhật thông tin bên trong). Trong đó, vùng [ENTRY TABLE] và [VOLUME INFO] ban đầu đã được lưu sẵn trong RAM khi đọc volume lên, và sẽ được cập nhật lại trước khi ghi toàn bộ xuống lại volume.

Em cũng có cài đặt chức năng phát hiện file/folder trùng tên để không cho người dùng import.

## CHỨC NĂNG ĐẶT MẬT KHẨU **Phần V**

Để đảm bảo tính bảo mật cho dữ liệu, tính năng đặt mật khẩu sẽ được mã hoá thành chuỗi 256 bit ngẫu nhiên dựa trên sơ đồ sau:





Password khi được cài đặt sẽ được bảo vệ với 3 lớp được gọi là “salt”, “pepper” và cuối cùng là “SHA-256”.

## 1. SHA-256:

SHA-256 là một trong những thuật toán hash an toàn nhất thời điểm hiện nay. Với password đầu vào có độ dài bất kì đều được hash và cho ra một chuỗi 256 bit có xác suất trùng nhau cực kì thấp và không có khả năng nào có thể suy ngược lại từ chuỗi bit đó ra password được nhập vào.

Từ chuỗi password được nhập vào, SHA-256 sẽ thêm cho đủ 256 bit và sau đó trộn 256 bit này với bộ dữ liệu mã hoá được cài đặt mặt định trong thuật toán.

Chuỗi hash = SHA-256(mật khẩu, bộ dữ liệu mã hoá)

Với tính chất của mã một chiều, cho dù password, chuỗi được hash, thuật toán SHA-256 được công khai thì không thể có cách nào suy ra được bộ dữ liệu mã hoá.

Với mỗi thuật toán SHA-256 được cài đặt sẽ có một bộ dữ liệu mã hoá riêng biệt do người lập trình cài đặt. Do vậy dù có suy ra được toàn bộ dữ liệu của một hệ thống bảo mật sử dụng SHA-256 thì cũng không thể áp dụng bộ dữ liệu đó lên hệ thống SHA-256 khác.

Như vậy cách duy nhất để hacker tấn công một hệ thống an ninh sử dụng SHA-256 là Brute-Force để là thử mọi trường hợp của SHA-256 tức là chỉ có 1 trường hợp đúng trong  $2^{256} = (2^{32})^8 \sim (4 \text{ tỷ})^8$  trường hợp.

Vậy Brute-Force  $2^{256}$  khó đến cỡ nào? Một chanel trên Youtube 3Blue1Brown đã thực hiện một chứng minh trong video “How secure is 256 bit security?” ([https://www.youtube.com/watch?v=S9JGmA5\\_unY&t=203s](https://www.youtube.com/watch?v=S9JGmA5_unY&t=203s)). Để kiểm tra toàn bộ  $2^{256} = (2^{32})^8 \sim (4 \text{ tỷ})^8$  trường hợp cần mất đến 37 tuổi của chính vũ trụ, mà trong đó tập hợp tất cả các nguồn lực từ tất cả thiên hà trong chính vũ trụ này (có khoảng 2 nghìn tỷ), tức là việc này không khả thi.

## 2. Salt & Pepper:

Vậy tại SHA-256 có độ hiệu quả kinh khủng như vậy thì tại sao cần đến Salt và Pepper?

Với mỗi input vào SHA-256 sẽ cho ra đúng 1 chuỗi hash không đổi. Do vậy điều đáng quan ngại nhất là khi nhiều người dùng sử dụng chung một password, như vậy chỉ cần hacker biết được mã hash đó được mã hoá từ mật khẩu nào thì chắc chắn những file hay folder có chung mật khẩu như vậy cũng sẽ bị hack. Để ngăn chặn tình trạng trên, cùng một mật khẩu được cài đặt phải được hash ra những chuỗi hash khác nhau.

### a. Salt

Với một chuỗi password có độ dài n, thuật toán sẽ ngẫu nhiên chọn 3 vị trí (có thể trùng nhau) trong những kí tự của password và đem chúng đi XOR với một bộ mã hoá gồm 3 kí tự (mỗi kí tự 1 byte).

Ví dụ chuỗi “password” khi được addSalt() ngẫu nhiên 10 lần sẽ cho ra kết quả như sau.

password



```
p/s;wo&d
p)ssmord
pa'swotd
>}ssword
p{s;word
pasuwo&d
pas=?or0
p5ss9o:d
pass9;:d
pass9o:0
```

## b. Pepper

Pepper sẽ thêm vào sau chuỗi password một kí tự được random trong 256 kí tự (trong bảng mã ASCII).

Ví dụ chuỗi “password” khi được addPepper() ngẫu nhiên 10 lần sẽ cho kết quả như sau.

password



```
password)
password#
password!
passwordä
passwordß
passwordl
passwordlf
password«
passwordR
passwordÉ
```

## 5. Kết hợp Salt & Pepper & SHA-256:

Ví dụ chuỗi “password” qua các lớp bảo mật như sơ đồ bảo mật trên thì sẽ cho ra kết quả như sau:

password



```
p/s;wo&d
p)ssmord
pa'swotd
>}ssword
p{s;word
pasuwo&d
pas=?or0
p5ss9o:d
pass9;:d
pass9o:0
```

addSalt()



```
p/s;wo&dG
p)ssmord|
pa'swotd|
>}ssword@
p{s;wordM
pasuwo&dL
pas=?or0C
p5ss9o:d7
pass9;:di
pass9o:0a
```

addPepper()



```
407507e7ca419ebdc75cf4a6dc04f48657ba2c2247d1d4cc7c3ab904f18b9ce0
67af2cb242095b511ef341ffc4503d8dbd441b1568e2d6b705733258eb87c3b3
3224980a6299d7d2ed3a0535938060fe859f803ec88129f7290ac77417f174db
a5c73d29d9b5b092ce349885b9827b1bf028e08cdf97676a6e2f75324e8ebfe
aedcfb6e4e1202ba4fe5f86118c2350576df1ba4d308b2f87106e98c38faba41
e234c18bbfcf12e5ab808a259e6a33c6efecf083e2c81da497b0fed66a1f8ef7
9e6dfd4fffc8a8e43c3397810cab0c0db4888baead0346b6056e13e1e23915ff0
68f42055889e99fb60680b8a441f3203ae55110c3eb325e365510dd03141da58
a95274dffff3aa4c8bb2c878fb5ab0b5d8ddaad3388c6c61d30ed43bac91a0c72
9d2905b1fce8f1658c68cd00e2b40971629500d653f2ddf15b945ece0386ea1c
```

SHA256()

### 3. Xác thực mật khẩu

Cùng một chuỗi mật khẩu được cài đặt cho ra khoảng  $256 * [\text{độ dài mật khẩu}]^3$  tổ hợp và được hash thành chuỗi 256 bit tương ứng. Tỷ lệ  $\frac{1}{256 * [\text{độ dài mật khẩu}]^3}$ .

Trung bình một mật khẩu được cho là an toàn có độ dài từ 10 – 16 kí tự như vậy sẽ có 256000 – 1048576 (~1 triệu) trường hợp cần xác thực, với hữu hạn trường hợp như vậy khả năng xác thực hoàn toàn có thể.

Khi người dùng nhập mật khẩu, chương trình sẽ dùng mật khẩu ấy thực hiện  $256 * [\text{độ dài mật khẩu}]^3$  lần mã hoá thành chuỗi hash và đem so với chuỗi hash đã được lưu, cho đến khi hai chuỗi trùng khớp thì mật khẩu đúng. Hoặc nếu thử hết tất cả trường hợp mà không tạo ra chuỗi hash nào trùng khớp thì mật khẩu sai.

### 4. Bàn luận

Tại sao không addSalt() thêm vào nhiều kí tự hơn mà chỉ dừng lại ở 3 ký tự? – Tại sao không addPepper() thêm vào nhiều kí tự hơn mà chỉ thêm 1 kí tự? – Điều đó có ảnh hưởng gì đến yếu tố bảo mật không?

Điều addSalt() thêm nhiều hơn 3 ký tự và addPepper() nhiều hơn 1 kí tự là hoàn toàn có thể thực hiện được. Tuy nhiên trên khuôn khổ của một chương trình thử nghiệm nên chỉ dừng lại ở mức này. Thực tế thêm càng nhiều Salt & Pepper thì hacker sẽ khó lần ra, đồng nghĩa với việc khi xác thực mật khẩu cũng mất nhiều thời gian hơn. Như vậy việc dừng lại ở 3 ký tự cho addSalt() và 1 ký tự cho addPepper() là hợp lí.

## CHỨC NĂNG EXPORT FILE/FOLDER TỪ VOLUME

## Phần VI

### 1. Tổng quát

Đầu vào: volume chứa file/folder cần được export; đường dẫn đến vị trí export file/folder ra.

Đầu ra: file/folder đã được export từ volume.

### 2. Ý tưởng

Để export 1 file, ta tạo 1 file trống ở vị trí người dùng muốn export file ra, và chép dữ liệu của file từ trong volume ra file trống đó.

Còn để export 1 folder, nếu suy nghĩ theo cách đơn giản nhất (và có thể là ngây thơ nhất) thì ta chỉ cần tạo 1 folder trống có tên là tên của folder được export, và đặt ở vị trí người dùng chỉ định là xong.

Tuy nhiên, cũng giống như chức năng import ở trên, trong thực tế khi ta export 1 folder, ta cũng phải export luôn cả những file và folder con chứa trong folder đó. Và folder đó trong volume cũng có cấu trúc dạng cây thư mục. Và vì thế, thuật toán “level order traversal” một lần nữa sẽ được sử dụng để duyệt cây thư mục đó, và export tất cả những file và folder trong cây đó ra.

### 3. Thuật toán

**Bước 1** Nếu ta export 1 file, đến bước 2; còn nếu ta export 1 folder, đến bước 3.

**Bước 2** Tạo 1 file trống ở đường dẫn vị trí người dùng muốn export file ra, rồi chép dữ liệu của file từ trong volume ra file trống đó. Kết thúc quá trình export.

**Bước 3** Gọi T là cây thư mục có root là folder người dùng muốn export. Ta sử dụng thuật toán “level order traversal” để duyệt qua tất cả các file và folder trong cây T. Mỗi lần truy cập một file/folder, nếu nó là file thì ta thực hiện giống như bước 2, còn nếu nó là folder thì ta chỉ cần tạo 1 folder ở vị trí tương ứng.

**Bước 3.1** Tạo 1 cấu trúc queue, gọi là Q. Queue này sẽ được dùng để lưu các entry của các file và folder trong cây T, vì các entry này lưu các thông tin cần thiết cho việc export các file và folder, như: tên, đường dẫn trong volume, kích thước, offset đến vị trí lưu dữ liệu.

**Bước 3.2** Push entry của root folder trong cây T vào đuôi queue Q.

**Bước 3.3** Kiểm tra xem queue Q có rỗng hay không. Nếu rỗng, kết thúc quá trình export. Nếu không rỗng, tiếp tục đến bước 3.4.

**Bước 3.4** Trích xuất entry nằm ở đầu queue Q (ta gọi file/folder có entry này là `cur_file`), sau đó xóa (pop) entry đó khỏi queue.

**Bước 3.5** Nếu `cur_file` là 1 file bình thường (không phải folder), đến bước 3.6; còn nếu `cur_file` là 1 folder, đến bước 3.7.

**Bước 3.6** Dùng thông tin lưu trong entry này để truy cập vị trí lưu `cur_file` trong volume. Ta sẽ tạo 1 file trống ở vị trí tương ứng với vị trí của `cur_file` trong cây T. Sau đó ta chép dữ liệu của file từ trong volume ra file trống đó. Rồi ta quay lại bước 3.3.

**Bước 3.7** Tạo 1 folder ở vị trí tương ứng với vị trí của `cur_file` trong cây T.

**Bước 3.8** Lấy đường dẫn của tất cả các file và folder con chứa trong `cur_file` và push vào đuôi queue Q. Sau đó quay lại bước 3.3.

### 4. Nhận xét

Nhìn chung thuật toán của chức năng Export có nhiều điểm tương đồng với thuật toán của chức năng Import. Điều đó cũng không có gì lạ nếu xét về việc 2 chức năng import và export giống như 2 quá trình thuận nghịch.

Em cũng có cài đặt chức năng phát hiện file/folder trùng tên để không cho người dùng export.

## 1. Tổng quan

Đầu vào File / Folder mong muốn xóa.

Đầu ra Xóa thành công File / Folder đó ra khỏi Volume File.

## 2. Ý tưởng

Thuật toán xóa 1 file trong volume rất đơn giản. Sau đây, em chỉ nói về thuật toán xóa 1 folder trong volume vì thuật toán này tổng quát, bao trùm cả thuật toán xóa 1 file trong volume.

Để xóa một folder, ta sử dụng thuật toán đệ quy để xóa các file / folder con của nó trước rồi mới đến các folder cha sau.

Chỉ xóa những file / folder không có mật khẩu. Những file / folder có mật khẩu sẽ được giữ lại. Nếu muốn xóa những file / folder chứa mật khẩu, ta phải nhập đúng mật khẩu trước rồi mới được phép thực hiện chức năng xóa.

## 3. Thuật toán xóa 1 folder trong volume

**Bước 1** Tìm tất cả các file và folder con của folder đang mà ta muốn xóa. (Nếu cái chúng ta cần xóa là file thì bỏ qua bước này và nhảy tới Bước 2. Gọi đệ quy để xóa tất cả các file và folder con mà ta vừa tìm được.

**Bước 2** Ta sẽ đệ quy cho đến khi gặp một file không có mật khẩu hoặc một folder rỗng không có mật khẩu thì mới được xóa, lúc này ta đến Bước 3. Bỏ qua tất cả các file/folder có mật khẩu và các folder không rỗng.

**Bước 3** Xóa nội dung của file/folder này trên volume file và xóa entry đang quản lý nó ra khỏi ENTRY TABLE. Cụ thể, ta sẽ dời dữ liệu đè lên vùng dữ liệu của file/folder này.

**Bước 3.1** Dời nội dung vùng DATA phía sau nội dung của file/folder mà ta cần xóa và đè lên chính nội dung của file/folder mà ta cần xóa. (Thật ra folder không có dữ liệu trên vùng DATA nhưng đây là thuật toán tổng quát nên vẫn hợp lý).

**Bước 3.2** Sau khi dời vùng DATA, ta sẽ cập nhật lại nội dung của ENTRY TABLE được lưu trên RAM. Cụ thể là xóa Entry quản lý file này ra khỏi ENTRY TABLE và cập nhật lại tất cả các giá trị "Vị trí đầu vùng dữ liệu" của các entry đứng sau entry mà ta đã xóa bằng cách lấy giá trị cũ trừ đi kích thước vùng dữ liệu của entry mà ta vừa xóa.

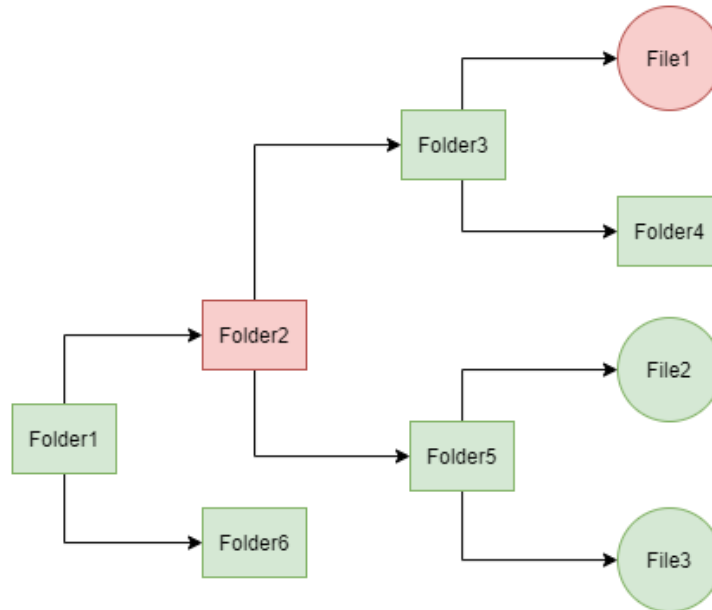
**Bước 3.3** Nối tiếp đuôi vùng DATA mà ta vừa mới dời, ta sẽ viết xuống Volume File toàn bộ nội dung của ENTRY TABLE mà ta vừa cập nhật.

**Bước 3.4** Cập nhật lại dữ liệu của vùng VOLUME INFO. Cụ thể là giá trị mới của 'Kích thước của vùng ENTRY TABLE' sẽ bằng giá trị cũ của nó trừ đi kích thước của entry mà ta vừa xóa, giá trị mới của "Vị trí bắt đầu vùng ENTRY TABLE" sẽ bằng giá trị cũ của nó trừ đi kích thước vùng dữ liệu của entry mà ta vừa xóa.

**Bước 3.5** Lúc này, kích thước của Volume File giảm đi một giá trị bằng chính kích thước vùng dữ liệu của entry mà ta đã xóa cộng với kích thước của entry mà ta đã xóa. Nên ta sẽ yêu cầu hệ điều hành cập nhật lại kích thước của Volume File này, nếu không cập nhật kích thước Volume File thì các giá trị phía sau cùng vẫn sẽ còn lại sau các công đoạn dời dữ liệu ở các bước trên, điều này dẫn tới làm hỏng cấu trúc Volume File.

**Bước 4** Folder cha nào quản lý file/folder này sẽ xóa nó ra khỏi danh sách các file/folder con.

#### 4. Minh họa thuật toán



Trong hình minh họa trên, các ô hình chữ nhật tượng trưng cho Folder, các ô hình tròn tượng trưng cho File. Màu đỏ tượng trưng cho File/Folder đó có mật khẩu (như trong hình, chỉ có Folder2 và File1 có mật khẩu), màu xanh lá cây tượng trưng cho File/Folder không có mật khẩu. Hình mũi tên tượng trưng cho quan hệ cha con giữa các File và Folder với nhau (như hình trên, Folder1 chứa Folder2 và Folder6, Folder2 chứa Folder3 và Folder5, ...).

Giả sử ta muốn xóa Folder2. Vì Folder2 này có mật khẩu nên ta buộc phải nhập chính xác mật khẩu thì mới được thực hiện thao tác xóa. Sau khi nhập chính xác, ta sẽ tiến hành xóa Folder2 qua các bước sau (Áp dụng thuật toán được nêu trên).

Xét Folder2:

- Thực hiện Bước 1: Folder2 chứa 2 folder con là Folder3 và Folder5. Ta thực hiện đệ quy để xóa Folder3 và Folder5.

Xét Folder3:

- Thực hiện Bước 1: Folder3 chứa File1 và Folder4. Ta thực hiện đệ quy để xóa File1 và Folder4.

Xét File1:

- Thực hiện Bước 1: bỏ qua bước này vì File1 là file.
- Thực hiện Bước 2: vì File1 có mật khẩu nên không được xóa File1.

Xét Folder4:

- Thực hiện Bước 1: Folder4 không có file / folder con nào cả.
- Thực hiện Bước 2: Folder4 rỗng và không chứa mật khẩu.
- Thực hiện Bước 3: xóa dữ liệu của Folder4 trên Volume File và trên RAM.
- Thực hiện Bước 4: Folder3 xóa Folder4 ra khỏi danh sách các file / folder con.

Xét ngược lại Folder3 (Vì đã xét hết các file/folder con của Folder3):

- Thực hiện tiếp Bước 2: ta thấy Folder3 vẫn còn chứa File1 nên không được phép xóa Folder3 này.

Xét Folder5:

- Thực hiện Bước 1: Folder5 chứa 2 file con là File2 và File3. Ta thực hiện đệ quy để xóa File2 và File3.

Xét File2:

- Thực hiện Bước 1: bỏ qua bước này vì File2 là file.
- Thực hiện Bước 2: File2 là file không có mật khẩu.
- Thực hiện Bước 3: xóa dữ liệu của File2 trên Volume File và trên RAM.
- Thực hiện Bước 4: Folder5 xóa File2 ra khỏi danh sách các file / folder con.

Xét File3:

- Thực hiện Bước 1: bỏ qua bước này vì File3 là file.
- Thực hiện Bước 2: File3 là file không có mật khẩu.
- Thực hiện Bước 3: xóa dữ liệu của File3 trên Volume File và trên RAM.
- Thực hiện Bước 4: Folder5 xóa File3 ra khỏi danh sách các file / folder con.

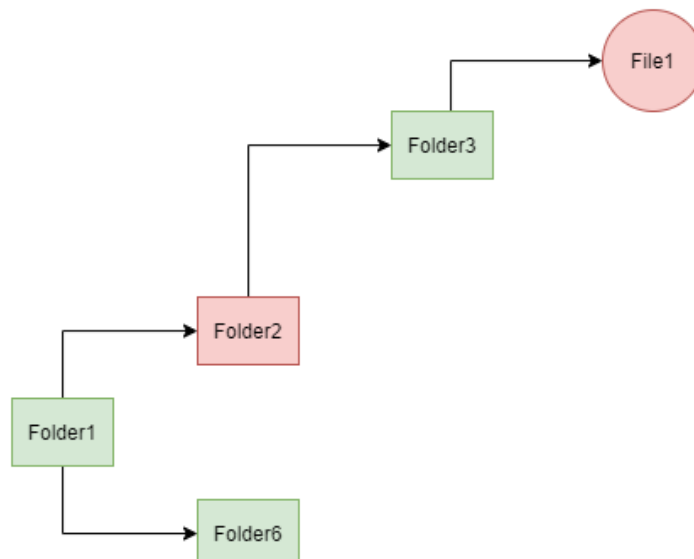
Xét ngược lại Folder5 (Vì đã xét hết các file / folder con của Folder5):

- Thực hiện tiếp Bước 2: ta thấy Folder5 lúc này đã là folder rỗng (đến đây thì tất nhiên Folder5 không có mật khẩu).
- Thực hiện Bước 3: xóa dữ liệu của Folder5 trên Volume File và trên RAM.
- Thực hiện Bước 4: Folder2 xóa Folder5 ra khỏi danh sách các file / folder con.

Xét ngược lại Folder2 (Vì ta đã xét hết các file / folder con của Folder2):

- Thực hiện tiếp Bước 2: Ta thấy Folder2 vẫn còn Folder3 trong danh sách các file / folder con nên không thể xóa Folder2 được.

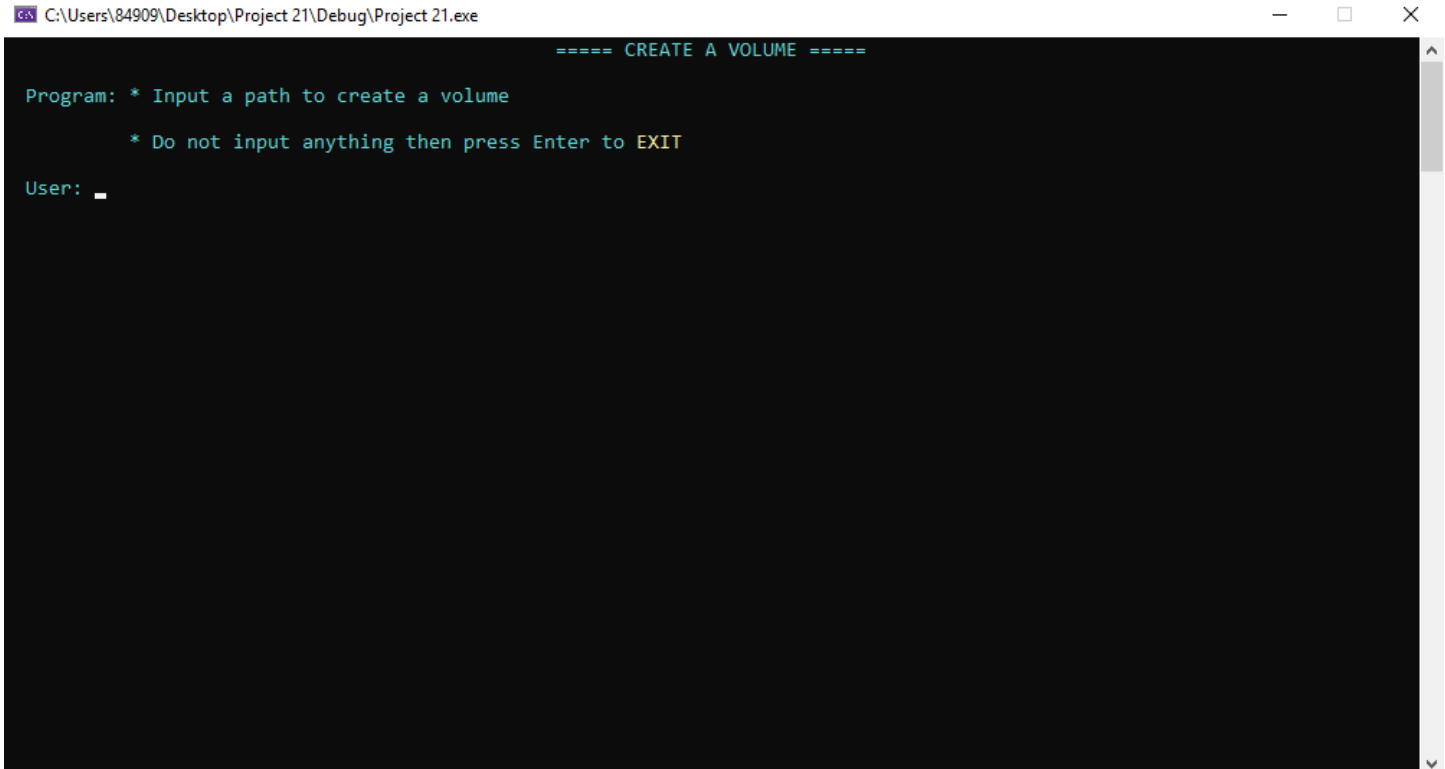
Kết thúc thao tác xóa Folder2. Kết quả là:







## 2. Create a folder



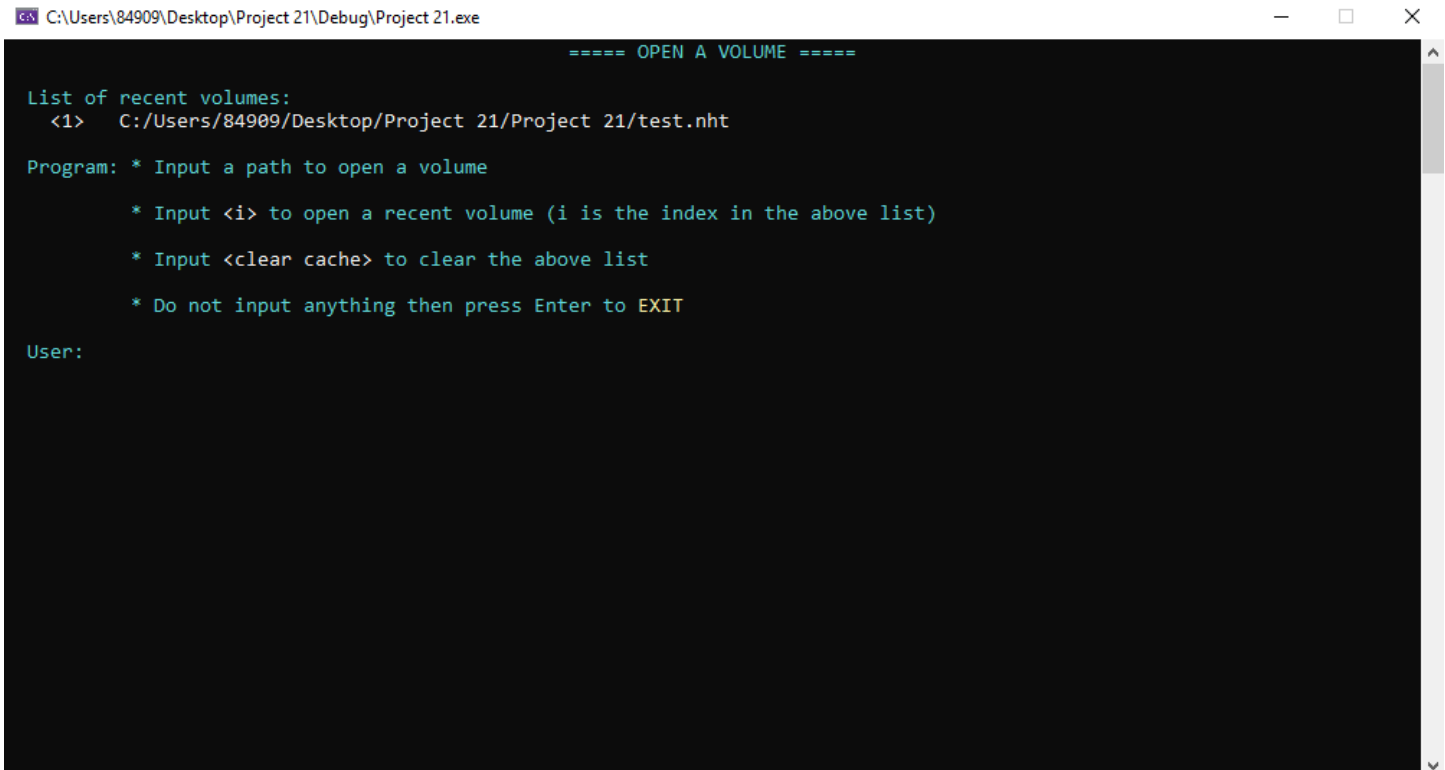
```
C:\Users\84909\Desktop\Project 21\Debug\Project 21.exe

==== CREATE A VOLUME =====

Program: * Input a path to create a volume
        * Do not input anything then press Enter to EXIT

User: _
```

## 3. Open a volume



```
C:\Users\84909\Desktop\Project 21\Debug\Project 21.exe

==== OPEN A VOLUME =====

List of recent volumes:
<1> C:/Users/84909/Desktop/Project 21/Project 21/test.nht

Program: * Input a path to open a volume
        * Input <i> to open a recent volume (i is the index in the above list)
        * Input <clear cache> to clear the above list
        * Do not input anything then press Enter to EXIT

User:
```

## 4. Instruction

C:\Users\84909\Desktop\Project 21\Debug\Project 21.exe

==== .NHT INSTRUCTION ====

\* Use ARROW 'UP' and ARROW 'DOWN' to move upward and downward between lines.

\* When you open an existing volume successfully these are functional key you should know:

[ENTER]: To enter a folder.

[P]: To set / reset password for a folder or file.  
If a folder/file has password, it requires password to do any manipulation on it.

[I]: To import a file/folder from a given path.

[E]: To export a file/folder from a given path.

[DEL] or [D]: To delete a file or a folder  
If you delete a folder which has a locked file/folder inside,  
It won't be deleted totally and still contain the locked files/folders inside.

[F1]: Show this instruction.

Press any key to continue . . .

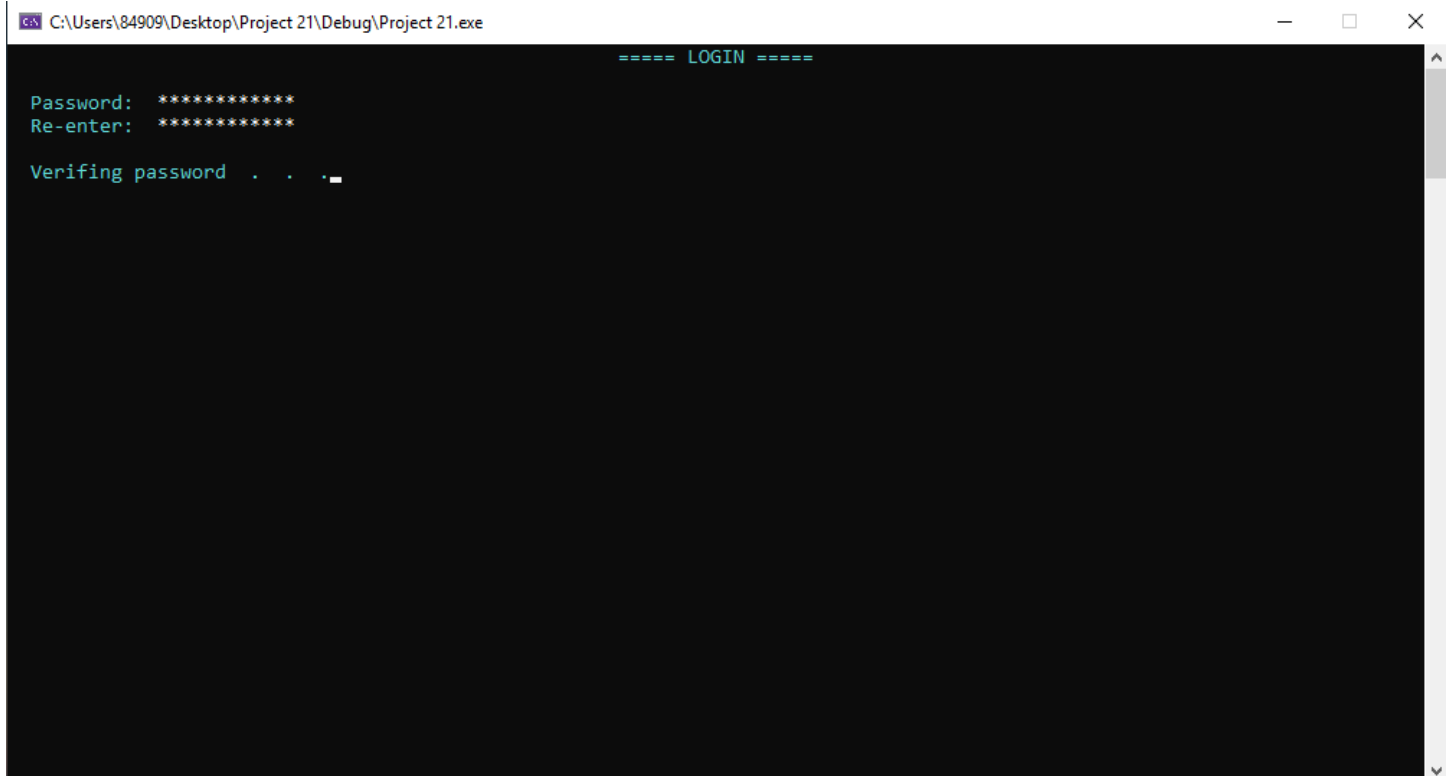
## 5. Show volume files/folders

C:\Users\84909\Desktop\Project 21\Debug\Project 21.exe

Path file-system/

Name	Size(bytes)	Type	Modified	Password
..		Folder		
.git	0	File	16/05/2020 14:12:48	[OFF]
Cache.cpp	3,203	File	15/05/2020 07:46:20	[OFF]
Cache.h	546	File	15/05/2020 07:30:46	[ON]
Console.cpp	3,456	File	15/05/2020 09:39:22	[OFF]
Console.h	1,226	File	15/05/2020 08:52:56	[OFF]
Entry.cpp	8,800	File	16/05/2020 14:01:52	[OFF]
Entry.h	2,198	File	15/05/2020 14:29:40	[OFF]
EntryTable.cpp	1,665	File	15/05/2020 07:30:46	[OFF]
EntryTable.h	426	File	15/05/2020 07:30:46	[OFF]
File.cpp	592	File	15/05/2020 07:30:46	[OFF]
File.h	491	File	15/05/2020 07:30:46	[OFF]
FileData.cpp	1,044	File	15/05/2020 07:30:46	[OFF]
FileData.h	388	File	15/05/2020 07:30:46	[OFF]
Folder.cpp	1,860	File	15/05/2020 07:30:46	[OFF]
Folder.h	721	File	15/05/2020 07:30:46	[OFF]
GUI.cpp	5,487	File	16/05/2020 08:51:26	[OFF]
GUI.h	771	File	16/05/2020 07:23:52	[OFF]
main.cpp	167	File	15/05/2020 07:30:46	[OFF]
Program.cpp	8,482	File	16/05/2020 14:01:52	[OFF]
Program.h	358	File	16/05/2020 14:01:52	[OFF]
README.md	53	File	12/05/2020 07:55:30	[OFF]
SHA256.cpp	13,759	File	15/05/2020 07:30:46	[OFF]
SHA256.h	1,993	File	15/05/2020 07:30:46	[OFF]
test	312	File	15/05/2020 07:30:46	[OFF]
Volume.cpp	26,646	File	16/05/2020 14:01:52	[OFF]
Volume.h	1,263	File	16/05/2020 14:01:52	[OFF]
VolumeInfo.cpp	2,056	File	16/05/2020 14:01:52	[OFF]
VolumeInfo.h	1,034	File	16/05/2020 14:01:52	[OFF]

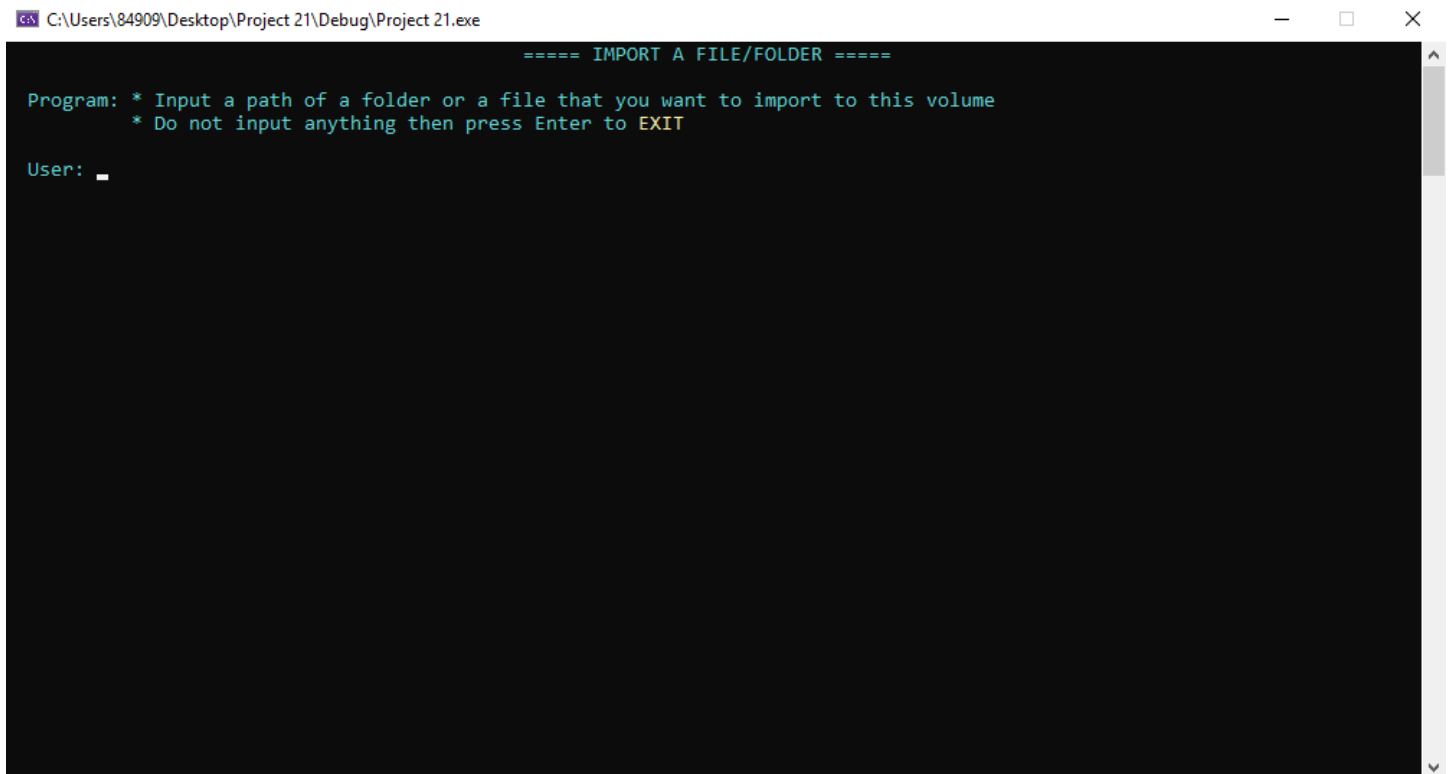
## 6. Check password



A screenshot of a Windows command prompt window titled "C:\Users\84909\Desktop\Project 21\Debug\Project 21.exe". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The command prompt displays the following text:

```
===== LOGIN =====  
  
Password: *****  
Re-enter: *****  
  
Verifing password . . .
```

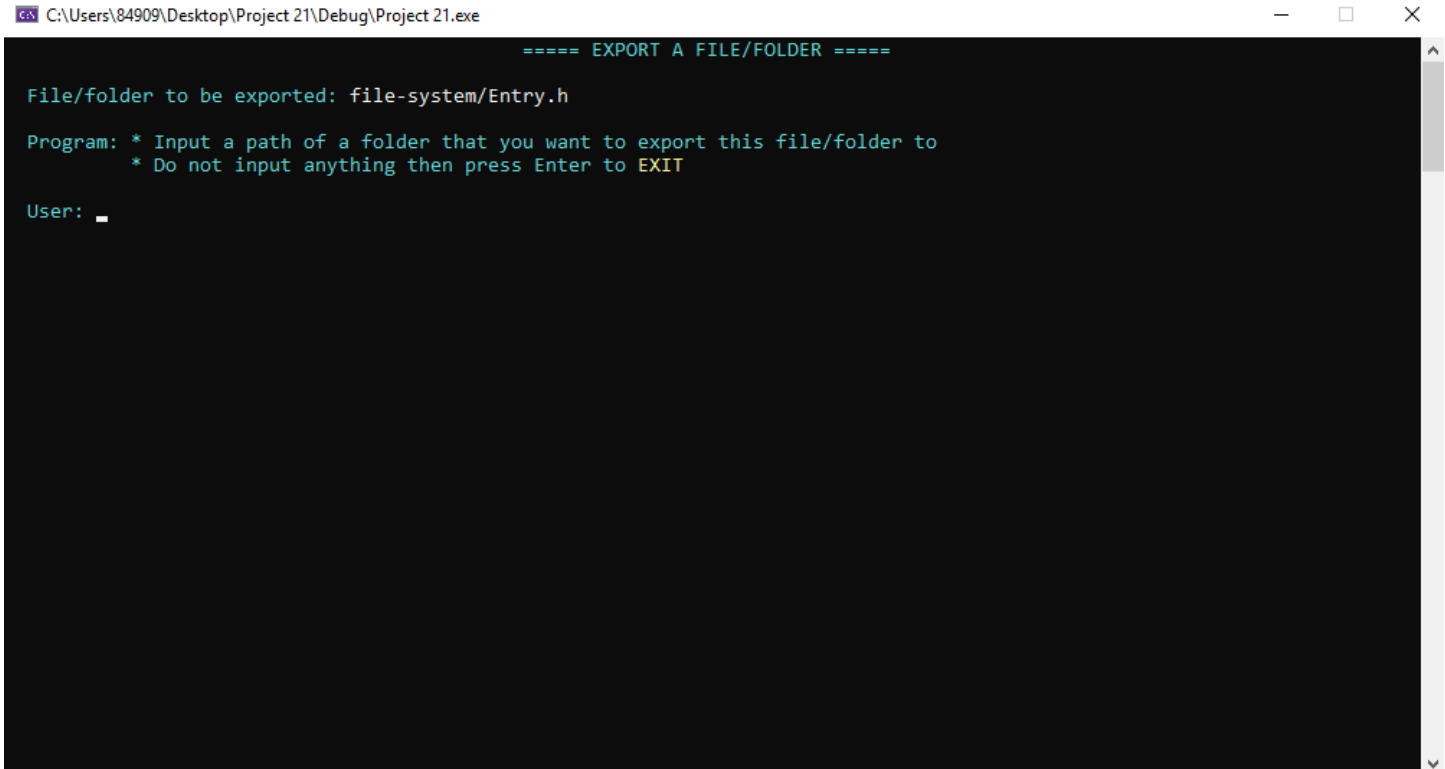
## 7. Import a file/folder



A screenshot of a Windows command prompt window titled "C:\Users\84909\Desktop\Project 21\Debug\Project 21.exe". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The command prompt displays the following text:

```
===== IMPORT A FILE/FOLDER =====  
  
Program: * Input a path of a folder or a file that you want to import to this volume  
         * Do not input anything then press Enter to EXIT  
  
User: _
```

## 8. Export a file/folder



```
C:\Users\84909\Desktop\Project 21\Debug\Project 21.exe

==== EXPORT A FILE/FOLDER ====

File/folder to be exported: file-system/Entry.h

Program: * Input a path of a folder that you want to export this file/folder to
         * Do not input anything then press Enter to EXIT

User: _
```

## 9. Nhận xét

Tuy chỉ là giao diện trên Console, nhưng nhóm em tự đánh giá giao diện của phần mềm khá thân thiện với người dùng và dễ dàng sử dụng.

## TỔNG KẾT **Phần IX**

### 1. Nhận xét về NHT

Theo nhóm tự đánh giá, cấu trúc NHT của nhóm rất ngắn gọn và tối giản, lược bỏ toàn bộ những chi tiết dư thừa và chỉ để lại những thông tin cốt lõi của dữ liệu. NHT cũng không được chia thành sector hay cluster dẫn đến việc có những vùng trống trong dữ liệu, làm tối ưu hoá việc lưu trữ dữ liệu trên ổ cứng, tiết kiệm không gian.

Bù lại đó chính là do vùng dữ liệu được lưu liên tiếp nhau và các ENTRY cũng được lưu liên tiếp nhau như vậy thì việc delete một file/folder trong Volume sẽ phải thực hiện dời dữ liệu rất nhiều để sắp xếp lại vùng dữ liệu và danh sách entry theo thứ tự như vậy trên một bộ dữ liệu nhỏ thì không thật sự cảm thấy NHT chậm ở khâu delete tuy nhiên khi dữ liệu được lưu trữ càng lớn, thì sẽ ảnh hưởng đáng kể đến tốc độ của chức năng này.

Cũng vì không được lưu theo cluster/sector nên không có vùng trống bị dư thừa, có vẻ đây là một ưu điểm nhưng cũng là một nhược điểm của NHT. Vì khả năng phục hồi lại dữ liệu vừa xóa là không có.

Nhóm em tin rằng NHT sẽ có tốc độ truy xuất nội dung file nhanh (open) vì chỉ cần đọc toàn bộ Entry vào RAM thì việc truy xuất lại dữ liệu được lưu trong volume rất nhanh chóng, không rườm rà phức tạp và cũng không bị phân mảnh. Nếu có cơ hội và thời gian, nhóm em mong muốn thử sức với chức năng open.

## 2. Một số hướng phát triển thêm

- Xử lý đường dẫn có tên file/folder chứa các kí tự Unicode.
- Phát triển giao diện Console thành GUI (Graphic User Interface).
- Nghiên cứu thêm về bảo mật để tạo ra nhiều lớp bảo mật hơn nữa, thay vì chỉ qua những bước đơn giản như hiện tại.
- Đồng bộ ModifiedTime và ModifiedDate với múi giờ của máy tính (chương trình đang sử dụng múi giờ chuẩn GMT+0).
- Tự động đổi tên nếu file/folder import vào bị trùng tên.
- Thêm chức năng mở file trực tiếp từ chương trình.
- Tự động nhận biết định dạng của file và hiển thị (Ví dụ: .docx sẽ hiển thị Microsoft Office Word...).
- Lưu được những **file** có kích thước lớn hơn 4GB.

## 3. Demo hướng dẫn sử dụng phần mềm

Video demo hướng dẫn sử dụng các chức năng của phần mềm chúng em để ở đường dẫn sau:

<https://drive.google.com/drive/folders/1f5LoOeF98EOQaTkgaz7YC5ao4HxyqAVJ>