

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN

Assignment 3.04

| Giáo viên giảng dạy |

GV. Thái Hùng Văn

GV. Đặng Trần Minh Hậu

| Sinh Viên thực hiện |

22127119 – Hồ Phước Hoàn

22127303 – Nguyễn Lê Đức Nhân

MÔN HỌC: HỆ ĐIỀU HÀNH

Mã lớp: 22CLC05

TP Hồ Chí Minh, 03/2024

Table of Contents

I. Thông tin nhóm	1
II. Đánh giá mức độ hoàn thành	1
III. Bảng phân công công việc	1
IV. Bài 1.....	2
V. Bài 2.....	12
VI. References.....	26

I. Thông tin nhóm

STT	MSSV	Họ và tên	Email
10	22127119	Hồ Phước Hoàn	hphoan22@clc.fitus.edu.vn
25	22127303	Nguyễn Lê Đức Nhân	nldnhan22@clc.fitus.edu.vn

II. Đánh giá mức độ hoàn thành

Yêu cầu		Chú thích	Mức độ hoàn thành
Câu 1	Viết chương trình C/C++ tạo tự động trên thư mục gốc mỗi volume 100 tập tin	Hoàn thành	100%
	Dự đoán và lý giải số cluster của RDET trên vol FAT32 sau khi thực hiện thao tác A.	Hoàn thành	100%
	Tạo một đoạn script (và lưu lại thành file .BAT /.CMD) thực hiện việc xóa các file	Hoàn thành	100%
	Cứu lại F0.dat trên vol FAT bằng cách dùng HexEdit /WinHex	Hoàn thành	100%
Câu 2a	Xây dựng mô hình và thiết kế kiến trúc tổ chức cho một hệ thống tập tin	Hoàn thành	100%
	Bảo mật thông tin	Hoàn thành	100%
	An toàn dữ liệu	Hoàn thành	100%
	Các thao tác xóa, truy xuất dữ liệu	Hoàn thành	100%
Câu 2b	Tạo / định dạng volume MyFS.Dat	Hoàn thành	100%
	Thiết lập /Đổi /Kiểm tra mật khẩu truy xuất MyFS	Hoàn thành	100%
	Liệt kê danh sách các tập tin trong MyFS	Hoàn thành	100%
	Đặt /đổi mật khẩu truy xuất cho 1 tập tin trong MyFS	Đã cài đặt hàm nhưng bị lỗi debug	50%
	Chép (Import) 1 tập tin từ bên ngoài vào MyFS	Hoàn thành	100%
	Chép (Outport) 1 tập tin trong MyFS ra ngoài	Đã cài đặt hàm nhưng bị lỗi debug	50%
	Xóa 1 tập tin trong MyFS	Chưa cài đặt hàm	0%

III. Bảng phân công công việc

STT	Công việc	Người thực hiện
1	Thực hiện câu 1	Hồ Phước Hoàn
2	Thực hiện câu 2a	Nguyễn Lê Đức Nhân
3	Câu 2b Định dạng volume Import, Outport file Liệt kê danh sách file Đặt đổi mật khẩu Xóa tập tin	Hồ Phước Hoàn
4	Câu 2b Thiết kế mô hình Mật khẩu cho hệ thống Mã hóa dữ liệu Thiết kế giao diện	Nguyễn Lê Đức Nhân

IV. Bài 1

A.

```
const int NUM_FILES = 100;
const int CLUSTER_FAT = 4096; // 512 * 8 = 4096
const int CLUSTER_NTFS = 2048; // 512 * 4 = 2048
const int CLUSTER_FAT32 = 1024; // 512 * 2 = 1024
```

Đây là những biến toàn cục dùng để lưu các giá trị của lần lượt là: số lượng file cần ghi, số byte trên 1 cluster của FAT, NTFS và FAT32.

```
// hàm tạo 100 file trong volume FAT
void createFilesInFAT() {
    for (int n = 0; n < NUM_FILES; ++n) {
        string file_path = "E:\\F" + to_string(n) + ".Dat";
        ofstream file(file_path);
        if (file.is_open()) {
            string number = to_string(2020 + n);
            /*
                Tính số lượng dòng cần tạo
                Vì mỗi dòng sẽ có các số từ 2020 -> 2119 sẽ chiếm 4 byte mỗi số và '\n' chiếm 2
byte nên
                cứ 1 dòng thì sẽ có 6 byte và dòng cuối sẽ có 4 byte.
                Công thức tính số dòng cần thêm số:  $(4 + 2) * x - 2 = (4 - n \% 4) * CLUSTER\_FAT$ 
                x: số dòng cần thêm vào file, kết quả của x sẽ làm tròn xuống vì để không bị lỗi số
cluster
                Công thức trên cũng áp dụng trong các loại hệ thống tập tin: FAT, NTFS, FAT32
            */
            float num_lines = floor(((4 - n % 4) * CLUSTER_FAT + 2) / 6.0);
            for (int i = 1; i < num_lines; ++i) {
                file << number << '\n';
            }
            file << number; // Dòng cuối không có xuống dòng
            file.close();
        } else {
            // Không thể mở file
            cerr << "Unable to create file: " << file_path << endl;
        }
    }
}
```

Hàm ghi 100 file .Dat vào volume FAT

```
// hàm tạo 100 file trong volume NTFS
void createFilesInNTFS() {
    for (int n = 0; n < NUM_FILES; ++n) {
        string file_path = "F:\\F" + to_string(n) + ".Dat";
        ofstream file(file_path);
        if (file.is_open()) {
```

```

        string number = to_string(2020 + n);
        float num_lines = floor(((4 - n % 4) * CLUSTER_NTFS + 2) / 6.0);
        for (int i = 1; i < num_lines; ++i) {
            file << number << '\n';
        }
        file << number; // Dòng cuối không có xuống dòng
        file.close();
    } else {
        // Không thể mở file
        cerr << "Unable to create file: " << file_path << endl;
    }
}
}

```

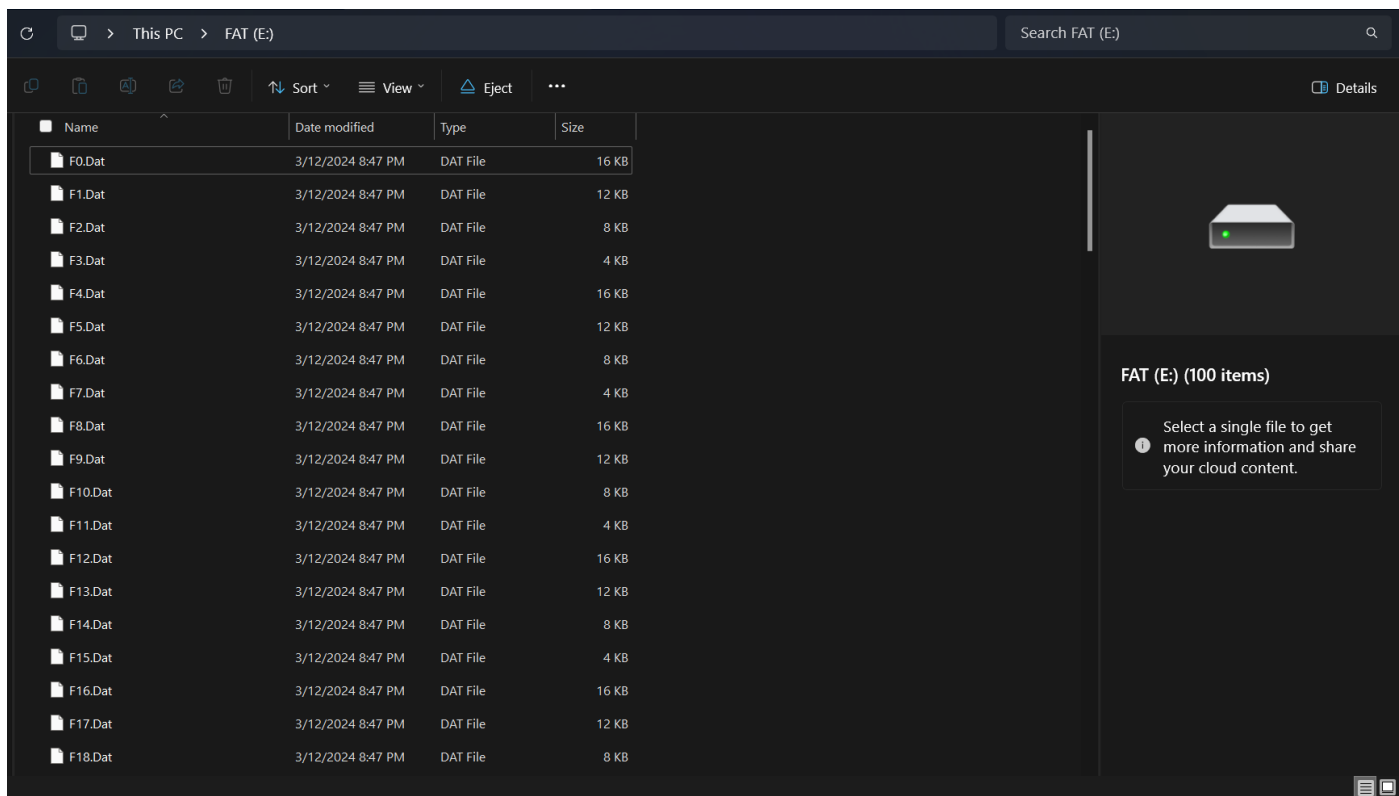
Hàm ghi 100 file .Dat vào volume NTFS

```

// hàm tạo 100 file trong volume FAT32
void createFilesInFAT32() {
    for (int n = 0; n < NUM_FILES; ++n) {
        string file_path = "G:\\F" + to_string(n) + ".Dat";
        ofstream file(file_path);
        if (file.is_open()) {
            string number = to_string(2020 + n);
            float num_lines = floor(((4 - n % 4) * CLUSTER_FAT32 + 2) / 6.0);
            for (int i = 1; i < num_lines; ++i) {
                file << number << '\n';
            }
            file << number; // Dòng cuối không có xuống dòng
            file.close();
        } else {
            // Không thể mở file
            cerr << "Unable to create file: " << file_path << endl;
        }
    }
}
}

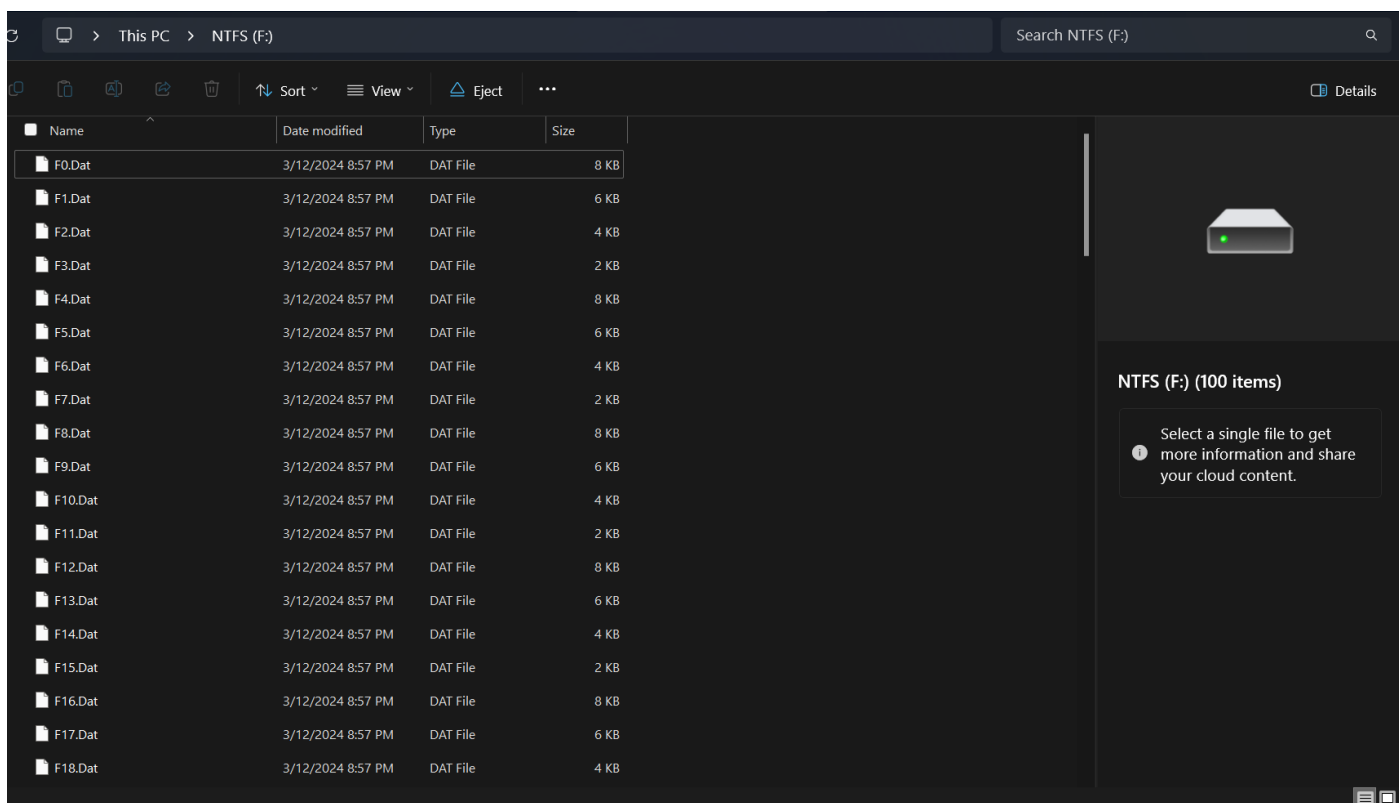
```

Hàm ghi 100 file .Dat vào volume FAT32



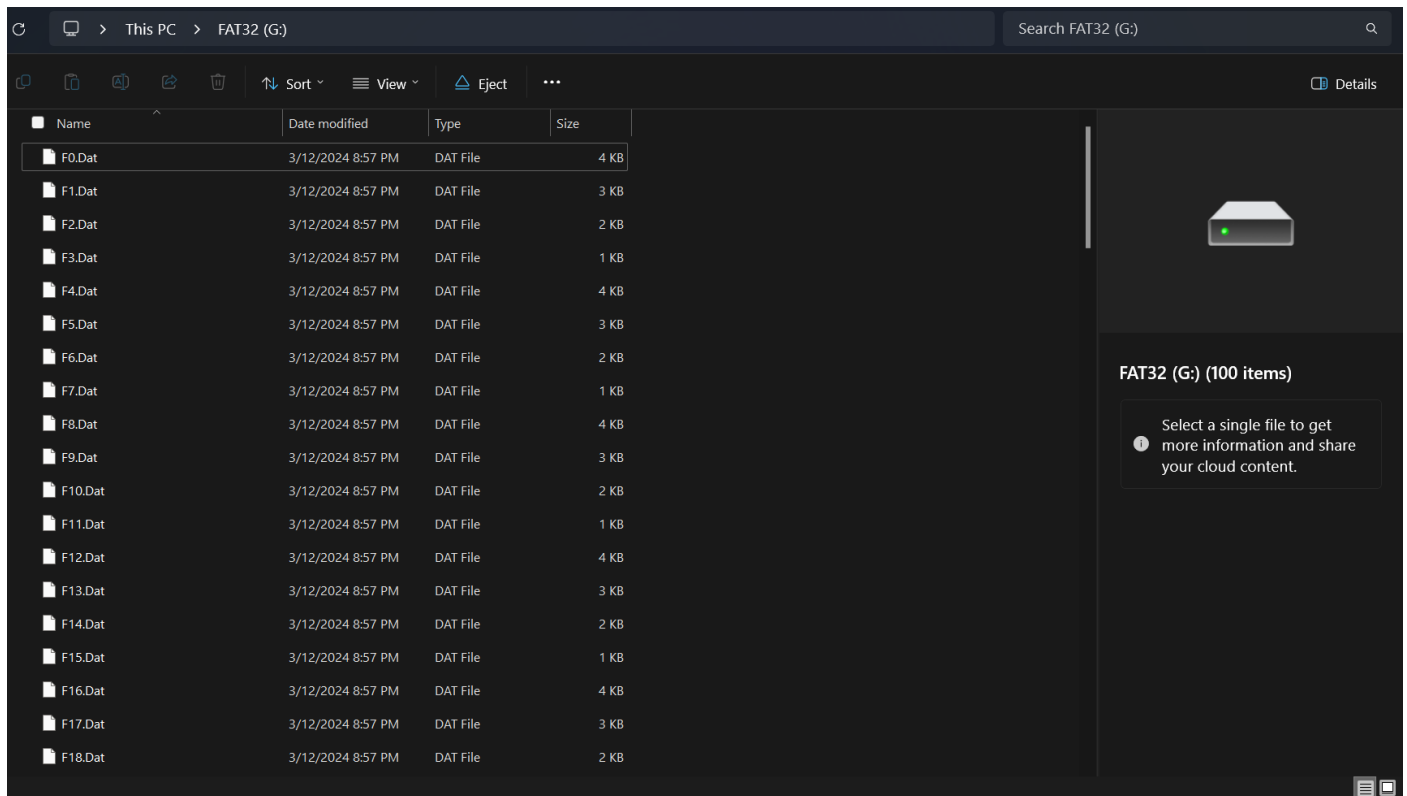
Kết quả chạy trong volume FAT

Ta thấy cứ 4 file 1 lần thì các file lại có kích thước giảm dần từ 16 KB → 4 KB. Mà trong format volume FAT thì mỗi cluster chiếm 4 KB ứng với 8 sector nên các file chiếm số cluster giảm dần từ 4 → 1 cluster.



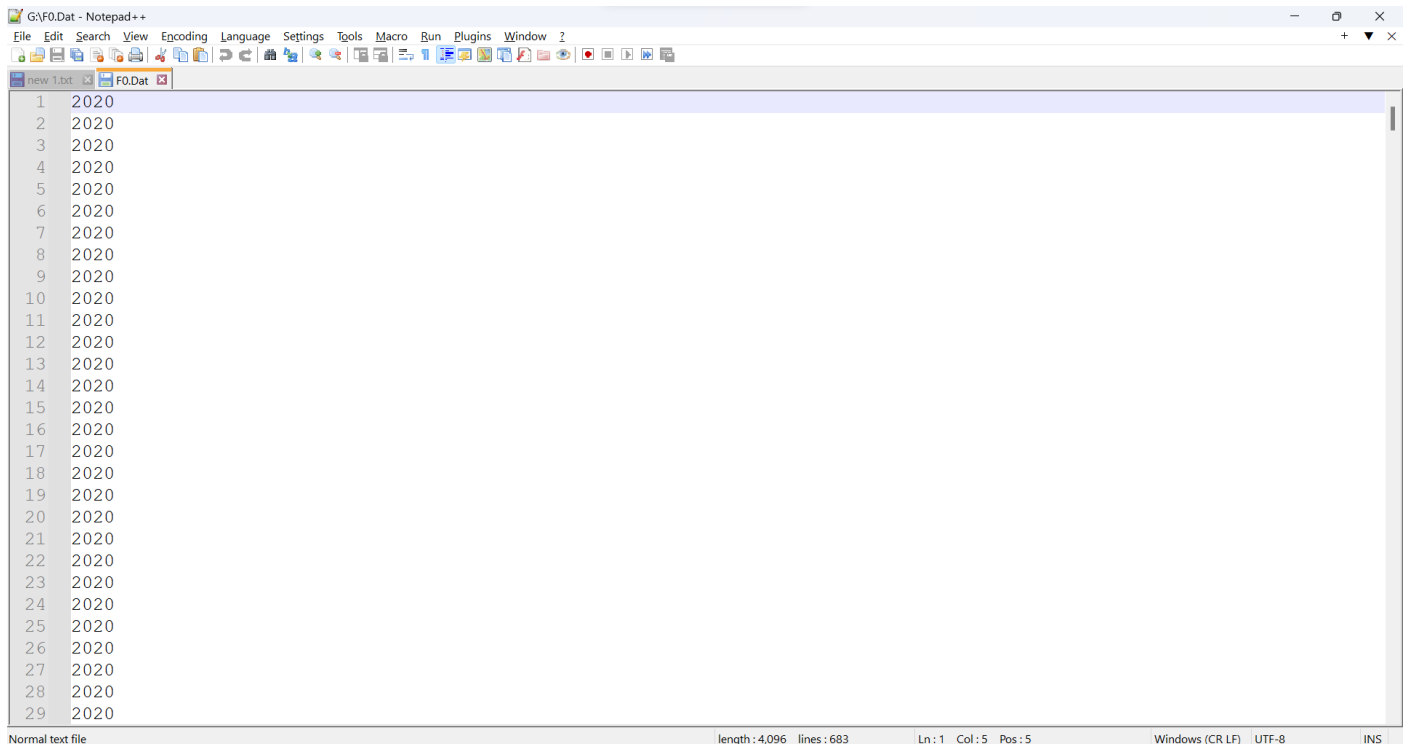
Kết quả chạy trong volume NTFS

Ta thấy cứ 4 file 1 lần thì các file lại có kích thước giảm dần từ 8 KB → 2 KB. Mà trong format volume FAT thì mỗi cluster chiếm 2 KB ứng với 4 sector nên các file chiếm số cluster giảm dần từ 4 → 1 cluster.



Kết quả chạy trong volume FAT32

Ta thấy cứ 4 file 1 lần thì các file lại có kích thước giảm dần từ 4 KB \rightarrow 1 KB. Mà trong format volume FAT thì mỗi cluster chiếm 1 KB ứng với 2 sector nên các file chiếm số cluster giảm dần từ 4 \rightarrow 1 cluster.



Ví dụ về file F0.Dat trong volume FAT

B.

Vì RDET bao gồm entry chính và entry phụ, ta có 100 file được lưu trữ, mỗi file là 1 entry, mỗi entry lại có 1 entry chính và chỉ 1 entry phụ bởi vì phần tên không dài quá 8 kí tự. Các entry đều có 32 byte nên khi lưu 1 file vào 1 entry

thì chiếm 64 byte (entry chính 32 byte + entry phụ 32 byte = 64 byte). Có 100 file như thế thì sẽ có $100 * 64 = 6400$ byte, mà mỗi cluster lại chiếm 1024 byte nên số cluster của RDET trên vol FAT32 sau khi thực hiện thao tác A sẽ là

$$\text{Số cluster} = \frac{6400}{1024} = 6.25 = 7 \text{ (cluster)}$$

Phải làm tròn lên 7 để đủ số cluster chứa các entry.

Vậy sẽ có 7 cluster của RDET trên vol FAT32 sau khi thực hiện thao tác A.

C.

```
@echo off

rem kiểm tra tham số đầu vào
if "%~1" equ "" (
    echo Invalid input! Missing input argument.
    echo Usage: %~f0 ^<directory^> ^<2n^>
    echo Example: %~f0 D:\test 20
    exit /b
)

if "%2" equ "" (
    echo Invalid input! Missing input argument.
    echo Usage: %~f0 ^<directory^> ^<2n^>
    echo Example: %~f0 D:\test 20
    exit /b
)

rem kiểm tra đường dẫn có tồn tại không
if not exist "%~1" (
    echo Directory does not exist.
    exit /b
)
```

Kiểm tra tính hợp lệ của input

```
rem chuyển đổi tham số đầu vào thành số nguyên
set /a "n=%2"
rem Kiểm tra nếu tham số đầu vào là số nguyên
if %2 neq 0 (
    echo %2
    if %n% equ 0 (
        echo %2 is not an integer.
        exit /b
    )
)

rem kiểm tra n có hợp lệ không (n phải là số chẵn và không âm)
if %n% lss 0 (
```



```

    echo Invalid input! ^<2n^> must be a non-negative integer
    exit /b
)
set /a "remainder=%n% %% 2"
if %remainder% neq 0 (
    echo Invalid input! ^<2n^> must be even
    exit /b
)

```

Gán số file muốn xóa vào n và kiểm tra tính hợp lệ của n

```

rem xóa các tập tin tương ứng
for /l %%i in (0,2,%%n%) do (
    rem xóa các tập tin mà không cần hỏi qua người dùng và loại bỏ thông báo lỗi nếu có
    rem nếu đường dẫn chỉ xuất phát từ ổ đĩa mà không đi qua thư mục nào vd: E:\ thì tham số đầu
    rem vào thứ nhất phải để là E: (bỏ đi dấu \)
    del "%~1\F%%i.Dat" /q 2>nul
)

```

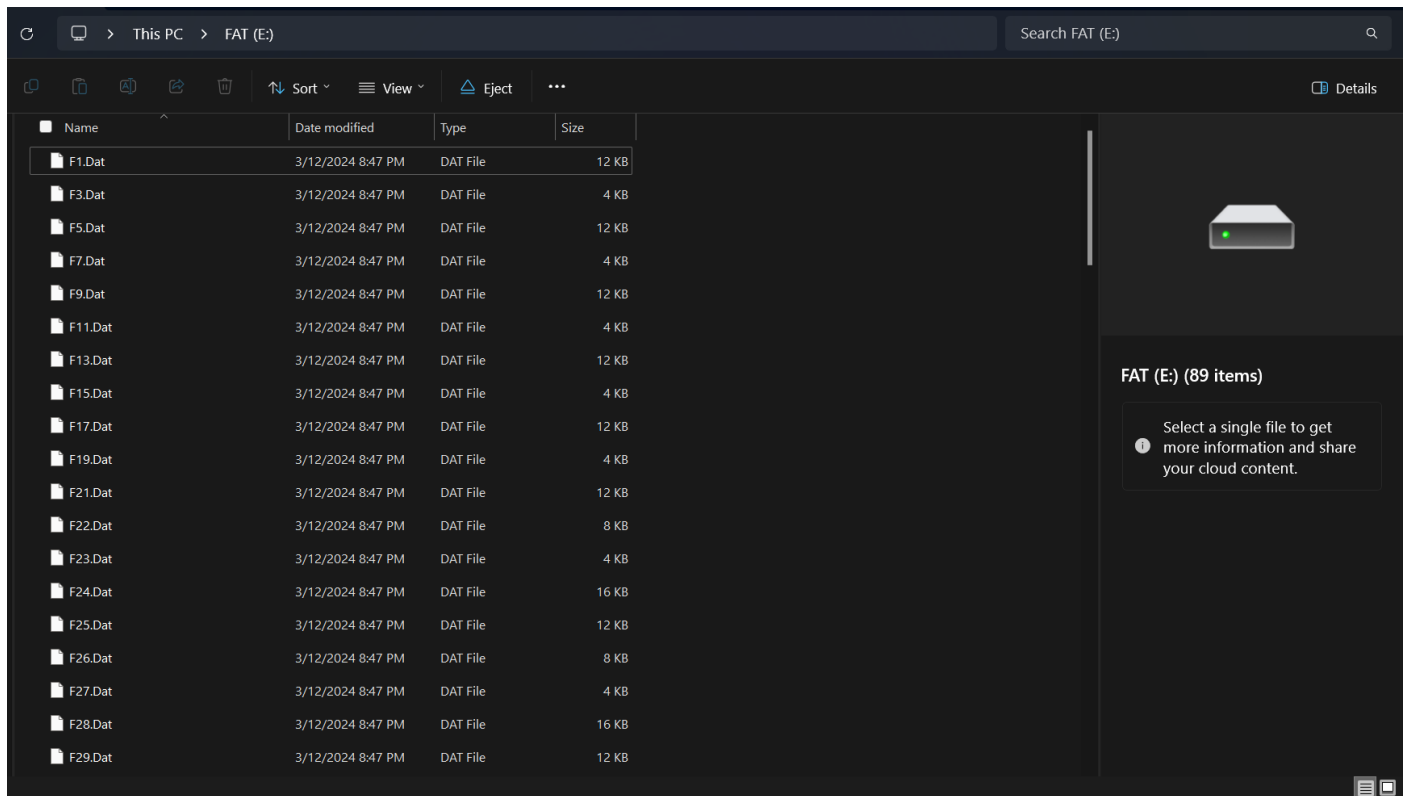
Xóa file bằng vòng lặp

```

Phuoc Hoan on Wednesday at 11:51 AM
> C:\Users\Phuoc Hoan\OneDrive - VNU-HCMUS\Work Space\My Uni\Operating Systems\Exercise\Assignment 03_04\test.bat D:\test 0
* History restored
> C:\Users\Phuoc Hoan\OneDrive - VNU-HCMUS\Work Space\My Uni\Operating Systems\Exercise\Assignment 03_04\test.bat E:
20
20
files delete successfully

```

Xóa các file chẵn với <2n> = 20 trong volume FAT bằng command line



Các file còn lại sau khi xóa bằng command line

D.

Trong volume FAT, để cứu file F0.Dat, ta tìm tới RDET của file F0.Dat. Tại đây vì file có tên <= 8 kí tự nên chỉ có 1 entry phụ và 1 entry chính. Khi file bị xóa, byte đầu của mỗi entry sẽ chuyển thành E5. Để khôi phục:

+ Tại entry chính, byte đầu tiên sẽ là kí tự đầu tiên của file là F, vì vậy trong editor ta sẽ chuyển E5 → 46h (F)

+ Tại entry phụ, byte đầu tiên sẽ đánh số thứ tự của entry phụ, vì chỉ có 1 entry phụ nên byte đầu tiên sẽ là 00000001 mà vì đây cũng là entry phụ cuối cùng nên 2 bit cao nhất sẽ là 01 → byte thực tế là 01000001b = 41h, ta chuyển E5 → 41h

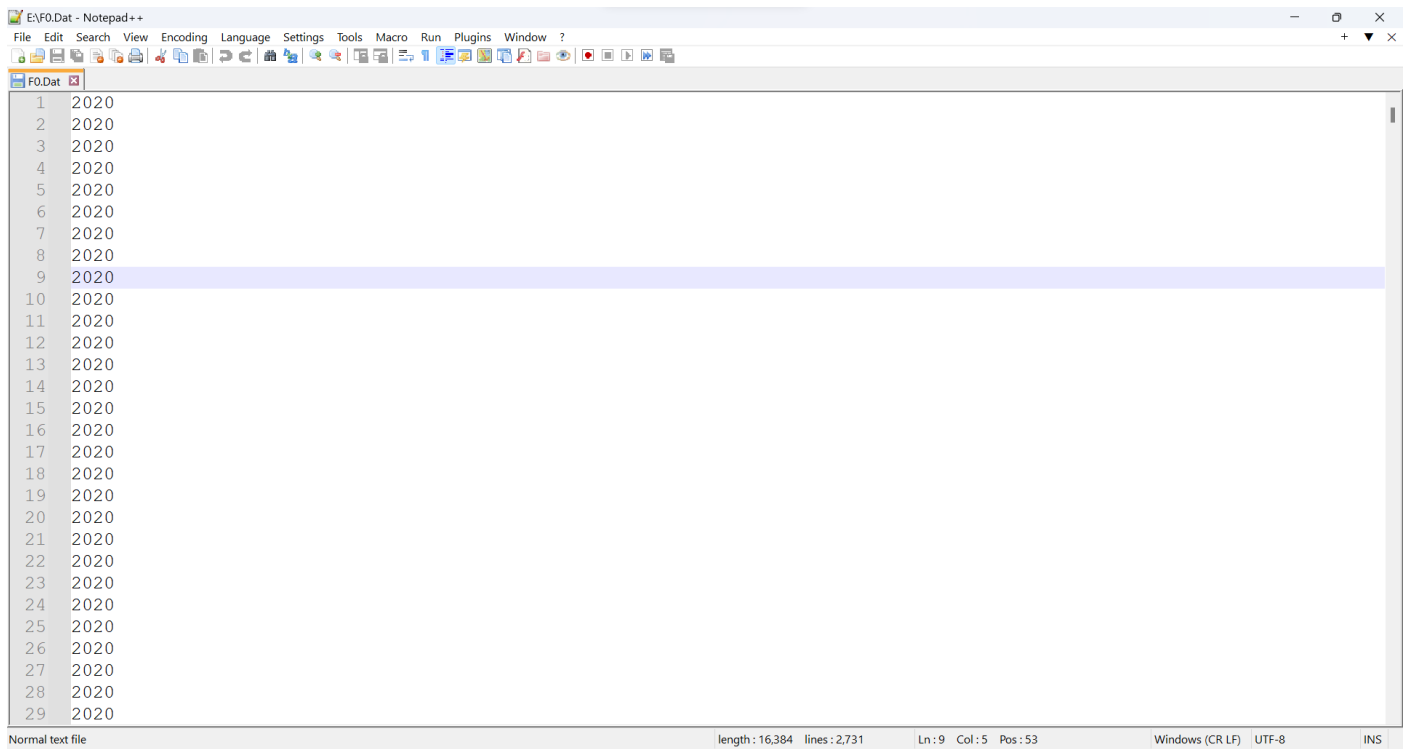
Startup	Drive E: (FAT) x																0123456789ABCDEF
00:1FE0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00:1FF0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00:2000	46	41	54	20	20	20	20	20	20	20	08	00	00	00	00	00	FAT
00:2010	00	00	00	00	00	00	C4	A5	6C	58	00	00	00	00	00	00Ã¥lX.....
00:2020	42	20	00	49	00	6E	00	66	00	6F	00	0F	00	72	72	00	B .I.n.f.o...rr.
00:2030	6D	00	61	00	74	00	69	00	6F	00	00	00	6E	00	00	00	m.a.t.i.o...n..
00:2040	01	53	00	79	00	73	00	74	00	65	00	0F	00	72	6D	00	.S.y.s.t.e...rm.
00:2050	20	00	56	00	6F	00	6C	00	75	00	00	00	6D	00	65	00	.V.o.l.u...m.e.
00:2060	53	59	53	54	45	4D	7E	31	20	20	20	16	00	A8	C3	A5	SYSTEM~1 .."Ã¥
00:2070	6C	58	6C	58	00	00	C4	A5	6C	58	02	00	00	00	00	00	lXlX..Ã¥lX.....
00:2080	E5	46	00	30	00	2E	00	44	00	61	00	0F	00	95	74	00	ãF.0...D.a...t.
00:2090	00	00	FF	FF	FF	FF	FF	FF	FF	FF	00	00	FF	FF	FF	FF	..ÿÿÿÿÿÿÿÿ..ÿÿÿÿ
00:20A0	E5	30	20	20	20	20	20	20	44	41	54	20	00	12	F9	A5	ã0 DAT ..ù¥
00:20B0	6C	58	6D	58	00	00	FA	A5	6C	58	04	00	00	40	00	00	lXmX..ú¥lX...@..
00:20C0	41	46	00	31	00	2E	00	44	00	61	00	0F	00	17	74	00	AF.1...D.a...t.
00:20D0	00	00	FF	FF	FF	FF	FF	FF	FF	FF	00	00	FF	FF	FF	FF	..ÿÿÿÿÿÿÿÿ..ÿÿÿÿ
00:20E0	46	31	20	20	20	20	20	20	44	41	54	20	00	13	F9	A5	F1 DAT ..ù¥
00:20F0	6C	58	6D	58	00	00	FA	A5	6C	58	08	00	FE	2F	00	00	lXmX..ú¥lX..þ/..
00:2100	E5	46	00	32	00	2E	00	44	00	61	00	0F	00	98	74	00	ãF.2...D.a...t.
00:2110	00	00	FF	FF	FF	FF	FF	FF	FF	FF	00	00	FF	FF	FF	FF	..ÿÿÿÿÿÿÿÿ..ÿÿÿÿ
00:2120	E5	32	20	20	20	20	20	20	44	41	54	20	00	14	F9	A5	ã2 DAT ..ù¥
00:2130	6C	58	6D	58	00	00	FA	A5	6C	58	0B	00	FC	1F	00	00	lXmX..ú¥lX..ü...
00:2140	41	46	00	33	00	2E	00	44	00	61	00	0F	00	14	74	00	AF.3...D.a...t.
00:2150	00	00	FF	FF	FF	FF	FF	FF	FF	FF	00	00	FF	FF	FF	FF	..ÿÿÿÿÿÿÿÿ..ÿÿÿÿ
00:2160	46	33	20	20	20	20	20	20	44	41	54	20	00	14	F9	A5	F3 DAT ..ù¥
00:2170	6C	58	6D	58	00	00	FA	A5	6C	58	0D	00	00	10	00	00	lXmX..ú¥lX.....
00:2180	E5	46	00	34	00	2E	00	44	00	61	00	0F	00	97	74	00	ãF.4...D.a...-t.
00:2190	00	00	FF	FF	FF	FF	FF	FF	FF	FF	00	00	FF	FF	FF	FF	..ÿÿÿÿÿÿÿÿ..ÿÿÿÿ

Template Results - Drive.bt					
Name	Value	Start	Size	Typ	
▼ direntry[2]	**Erased name 'F0.Dat' (Archive)	2080h	40h	struct FAT_DIRECT	
> long_entry		2080h	20h	struct FAT_LONGI	
> short_entry	**Erased name '?0.DAT' (Archive)	20A0h	20h	struct FAT_SHORTI	
> possibleDeletedData		8000h	200h	struct FAT_FILE_D	
▼ direntry[3]	F1.Dat (Archive)	20C0h	40h	struct FAT_DIRECT	
> long_entry		20C0h	20h	struct FAT_LONGI	

Trước khi chuyển

Sau khi sửa nội dung bảng FAT xong

Ta thấy file đã quay trở lại



The screenshot shows a Notepad++ window titled "E\F0.Dat - Notepad++". The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and ?. The toolbar contains various icons for file operations and editing. The text area displays a list of 29 lines, each starting with a line number (1-29) followed by the year "2020". The 9th line is highlighted in blue. The status bar at the bottom indicates "Normal text file", "length : 16,384", "lines : 2,731", "Ln : 9", "Col : 5", "Pos : 53", "Windows (CR LF)", "UTF-8", and "INS".

Line	Content
1	2020
2	2020
3	2020
4	2020
5	2020
6	2020
7	2020
8	2020
9	2020
10	2020
11	2020
12	2020
13	2020
14	2020
15	2020
16	2020
17	2020
18	2020
19	2020
20	2020
21	2020
22	2020
23	2020
24	2020
25	2020
26	2020
27	2020
28	2020
29	2020

Nội dung file vẫn giữ nguyên như ban đầu

Vậy là khôi phục file F0.Dat thành công!

V. Bài 2

A.

1. Kiến trúc tổ chức của Volume bao gồm phần System chứa thông tin quản lý và phần Data được quản lý theo đơn vị cluster

BootSector	BootSector(bản sao)	Bảng quản lý cluster	Bảng quản lý cluster(bản sao)	Bảng Thư mục (Data và bảng này xen kẽ nhau)	Data
------------	---------------------	----------------------	-------------------------------	---	------

(kích thước của volume được người dùng định dạng và có giới hạn là 1GiB, từ đó các vùng trên hệ thống cũng được quy định phù hợp)

a. Phần System

- BootSector (bảng chính và bản sao)

Offset(hex)	Số byte	Ý nghĩa
0	2	Số byte trên sector (thường là 512)
2	1	Số sector trên cluster (Sc)
3	1	Số sector thuộc vùng boot sector (Sb)
4	1	Số bảng FAT (Nf)
5	4	Kích thước volume (Sv), max = 1gb
9	4	Kích thước mỗi bảng FAT (Sf)
D	1	Key mật khẩu volume
E	1	Cluster bắt đầu của RDET
F	1	Sector chứa bản lưu
10	4	Tên filesystem ("myFS")

- Bảng quản lý cluster (bảng chính và copy)
- Dựa vào thông số dung lượng mà người dùng nhập vào sẽ quyết định chọn loại bảng và kích thước phù hợp cho hệ thống tập tin (sẽ nói ở phần sau)

Trạng thái của Cluster	Kích thước phần tử của bảng			Ghi chú
	1.5B	2B	4B	
Trống	0	0	0	=FREE
Hư	FF7	FFF7	0FFFFFFF7	=BAD
Cluster cuối của file	FFF	FFFF	0FFFFFFF	= EOF
Chứa nội dung file	2 .. FEF	2 .. FFEF	2..0FFFFFFEF	

b. Phần Data

• Bảng thư mục

Offset(hex)	Số byte	Ý nghĩa
0	7	Tên ngắn (Ascii)
7	3	Tên mở rộng
A	1	Thuộc tính trạng thái
B	1	Key Password
C	3	Giờ tạo (giây, phút, giờ: 1 byte)
F	3	Ngày tạo (ngày, tháng, năm – 2024: 1 byte)
12	3	Giờ truy cập gần nhất (như trên)
15	3	Ngày truy cập gần nhất (như trên)
18	4	Cluster bắt đầu
1C	4	Kích thước nội dung tập tin

• Data

- Lưu entry chính và phụ. Mỗi entry có 1 entry chính và có thể có 1 hoặc nhiều entry phụ. Entry chính ghi vào volume trước rồi mới tới entry phụ, ngược lại so với FAT32. Để ghi entry, đầu tiên tìm cluster thích hợp, sau đó ghi entry đến hết cluster. Trong cluster đó, số entry được ghi vào tương ứng số file được lưu, số file được lưu đó sẽ ứng với số cluster được lưu trong bảng quản lý cluster. Sau đó ghi nội dung của số file đó vào số cluster đã có trong bảng quản lý cluster. Cluster tiếp theo sau đó sẽ là cluster quản lý entry tiếp theo và cứ như vậy.

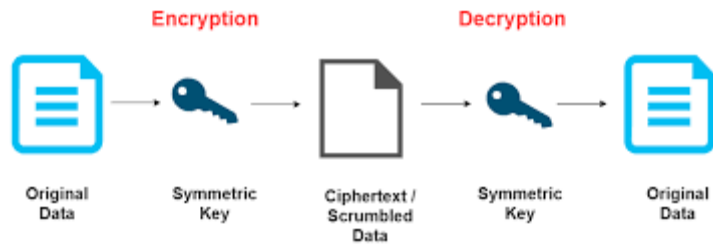
2. Cách định dạng volume

S_V	Kích thước volume
S_C	Kích thước cluster
S_R	Kích thước bảng thư mục
N_F	Số bảng quản lý cluster
S_F	Kích thước bảng quản lý cluster
S_B	Số sector vùng BootSector (trước bảng quản lý cluster)
S_D	Kích thước nơi lưu trữ data

- Từ kích thước volume người dùng qui định trước ta tính toán được S_C , S_R , N_F và S_B
- Từ đó ta tìm được S_F dựa trên đẳng thức $S_B + N_F * S_F + S_R + S_D = S_V$
(lần lượt thử từng giá trị $S_F = 1, 2, 3, \dots$ và S_D phù hợp và kiểm tra 2 thông số này có phù hợp nhau không cho đến khi hợp lý)
- Lưu các thông số trên vào đúng các offset và kích thước được qui định ở bảng BootSec trên
- Tạo 1 vùng đệm có kích thước ($S_R * 512$) byte mang toàn giá trị 0 và lưu vào S_R sector bắt đầu tại sector ($S_B + N_F * S_F$)
- Tạo một nội dung đặc biệt rồi ghi xuống & đọc lên từng cluster từ cluster 2 đến cluster $[(S_D + S_C - 1) / S_C]$. Nếu không thành công trong việc đọc / ghi, hoặc thời gian đọc / ghi quá lâu, hoặc nội dung đọc được không giống nội dung ghi thì gán cho phần tử quản lý tương ứng trên bảng quản lý cluster giá trị FF7, FFF7 hoặc 0FFFFFFF (cluster hư), ngược lại gán 0.

3. Việc bảo mật thông tin

- Encryption thông tin file khi import vào volume
- Chuyển đổi data trong file thành loại data mà người khác không thể đọc được bằng phương pháp mã hóa đối xứng.



- Data file sẽ được chuyển từ data đọc được (con người đọc có thể hiểu) sang loại data không đọc được (sử dụng bằng các thuật toán mã hóa. Sau đó được lưu vào phần Data trong hệ thống tập tin của chúng ta
- Từ đó bảo mật thông tin trong file cho dù người khác có thể lấy thông tin của file qua volume nhưng sẽ không thể đọc được do cần phải sử dụng cùng khóa giải mã (Symmetric key) để có thể chuyển lại thành data đọc được
- Access Control
 - Bằng cách sử dụng các thuộc tính trạng thái ta có thể tránh việc xóa nhầm những tài liệu quan trọng (file system). Hoặc che dấu trong hệ thống (hidden).
 - Ngoài ra ta còn ngăn chặn cho việc import file ra bên ngoài volume khi trạng thái là Read-only.
 - Khi điều chỉnh những trạng thái này người dùng (người thêm các file vào hoặc người tạo volume đó) được yêu cầu thiết lập mật khẩu để đảm bảo thống nhất và an toàn thông tin.
- Sử dụng mật khẩu bằng hàm băm (hash function)
 - Sử dụng mật khẩu, người dùng khi muốn bảo mật volume hoặc file sẽ thực hiện đặt mật khẩu, mật khẩu này không được lưu trữ bình thường mà được biến đổi thành 1B và lưu vào trong hệ thống (trong BootSec và bảng thư mục)
 - Hệ thống sẽ chuyển đổi chuỗi dữ liệu đầu vào (input) và xử lý bằng thuật toán băm ta sẽ có 1 đầu ra duy nhất (với những thuật toán băm kém hơn thì vẫn có trường hợp đụng độ giá trị băm (2 input cùng cho 1 output)
 - Là quá trình 1 chiều nghĩa là chỉ có thể biến đổi input sang output cho nên người dùng dù có lấy được giá trị băm cũng không thể lừa hệ thống để lấy được dữ liệu từ file cũng như volume.
- Thực hiện xóa không cho phục hồi
 - Xóa không cho phục hồi ở đây có nghĩa là ghi đè lên dữ liệu cũ bằng một dữ liệu khác nhiều lần để đảm bảo các phương pháp khôi phục dữ liệu không thực hiện được
 - Các dữ liệu ghi đè cũng được chọn ngẫu nhiên (dữ liệu rác) để tăng độ phức tạp trong việc khôi phục
 - Dù là vậy nhưng cũng chỉ có thể ngăn chặn những biện pháp cấp thấp vẫn có thể phục hồi dữ liệu (cách này rất tốn tài nguyên) và việc ghi đè nhiều lần cũng có rủi ro về hiệu suất nếu làm việc trên hệ thống lớn

4. Việc an toàn dữ liệu

- Luôn có nhưng bản sao (backup) cho các phần quản lý hệ thống như BootSector và bảng quản lý cluster
- Nếu xảy ra sự cố mất mát dữ liệu quan trọng trong phần hệ thống như BootSector hay bảng quản lý cluster, lúc đó ta không thể truy cập lại được vào các file ở bên trong vì đã mất đi vị trí cần tìm cluster chứa dữ liệu hoặc tệ hơn không còn có thể tương tác với cả hệ thống nếu mất đi các thông số quản lý như S_C , S_R , S_B ...
- Đảm bảo tính tính nguyên tử (Atomicity): mọi thay đổi về dữ liệu phải đảm bảo trọn vẹn, nếu các tiến trình thực hiện thành công hoặc là sẽ không có bất kỳ sự thay đổi nào về dữ liệu nếu có sự cố tiến trình xảy ra.
 - Ví dụ: ta muốn thay đổi tên của file trong volume, kết quả ta muốn có sẽ là file sẽ có tên mới hoặc là tên cũ chứ không phải là cả hai cũng như không xảy ra gì hết
 - Áp dụng thuật toán Shadow Paging: khi thực hiện thay đổi dữ liệu ta sẽ tạo và lưu trữ 1 bản sao của phần đang thực hiện thay đổi. Các thay đổi khi nào sẽ được thực hiện trên bản sao đó và khi thay đổi kết thúc, kết quả nếu đúng như mong đợi sẽ được ta sao chép từ bản sao và bản gốc. Nếu như kết quả sai, thì ta sẽ xóa đi bản sao và bản gốc được giữ nguyên như trước khi thay đổi.
- Đảm bảo tính nhất quán dữ liệu (Consistency): các dữ liệu trong volume phải đảm bảo đúng và theo tuân theo cấu trúc của hệ thống
 - Ví dụ: khi đưa 1 file vào volume, ta phải đảm bảo tên file hợp lệ (không có các ký tự đặc biệt \backslash , \backslash , $*$, $<$, $>$, $;$), và có 1 nơi lưu trữ cho file

- Áp dụng: ví dụ số cluster trong volume phải luôn bằng tổng số cluster đã sử dụng và cluster trống, và đảm bảo kích thước của file ta đưa phải hợp lệ với kích thước trống còn lại thì mới thực hiện việc thêm vào

5. Việc xóa dữ liệu

a. Xóa bình thường (thiết kế để hạn chế việc chúng bị chép đè),

- Dựa vào input đầu vào của người dùng (dãy ký tự tên file cần xóa), hệ thống sẽ xác định vị trí chính xác bảng thư mục đang quản lý file đó (nếu không có sẽ trả về không tìm thấy).
- Dựa vào thông tin cluster bắt đầu và kích thước file trong bảng, xác định các cluster chứa data và thực hiện xóa chúng để thành các cluster trống.
- Chính byte đầu tiên trong bảng thành E5, các byte sau vẫn giữ nguyên.
- Bằng cách xóa này ta có thể khôi phục lại được dữ liệu vì việc tìm lại trong bảng quản lý có byte đầu E5. (Khi thực hiện chép file vào ta sẽ ưu tiên các bảng có byte đầu là 0 trước sau đó mới đến E5 vậy nên sẽ có khả năng các bảng E5 không bị ảnh hưởng).
- Từ đó ta có thể tìm lại file bị xóa bằng cách phục hồi dữ liệu dựa vào cluster bắt đầu và kích thước tập tin đó trong phần data.

b. Xóa không cho phục hồi.

- Thực hiện y như trên nhưng thay vì chỉ xóa dữ liệu ở các cluster đi, ta ghi đè các giá trị rác vào các cluster chứa data file.
- Ta cũng chỉnh byte đầu thành E5 và cũng ghi đè các giá trị rác vào 31 byte còn lại.
- Việc ghi đè các giá trị rác nên được thực hiện nhiều lần để đảm bảo người khác không thể dùng các thiết bị đặc biệt có thể khôi phục dữ liệu đã bị ghi đè.

6. Cách truy xuất dữ liệu

a. Import

Chép file từ bên ngoài vào rồi sau đó lưu nội dung file và các thông số vào entry, nếu muốn lưu mật khẩu cho file thì lưu còn không thì mặc định key mật khẩu sẽ khởi tạo là 0, sau đó tìm cluster chứa file trong bảng quản lý cluster, ghi nội dung entry vào volume và ghi nội dung file vào volume.

b. Export

Khi muốn export file, nếu file có mật khẩu thì sẽ kiểm tra mật khẩu của file trước, nếu đúng thì cho phép export bằng cách đọc nội dung file từ volume (truy xuất vị trí và những cluster chứa nội dung file); nếu không đúng thì sẽ thoát.

B.

- **Hiển thị menu của file system gồm:**

1. **Tạo volume mới**
2. **Truy cập vào volume đã tạo**

```
Set MyFS Menu:  
1. Create new volume  
2. Access existed volume  
3. Exit
```

Hiển thị menu của file system

Mỗi volume bắt buộc phải có mật khẩu nên ở mỗi bước truy cập volume đã tạo thì đều phải nhập mật khẩu, nếu đúng thì tiếp tục vào bảng chức năng, không thì thoát ra và chạy lại menu.

```
Set MyFS Menu:  
1. Create new volume  
2. Access existed volume  
3. Exit  
Your choice: 1  
File system name: MyFS1.Dat  
Set Volume Password: 1  
Set volume size: 1024  
Successfully create new volume  
MyFS Management Menu:  
1. change Volume Password  
2. List Files  
3. Change File Password  
4. Import File  
5. Export File  
6. Delete File temporarily  
7. Delete File permanently  
8. Recover File  
9. Show volume left  
0. Exit
```

Hiển thị các chức năng của file system

```

1  #include "volume.h"
2  #include <typeinfo>
3  using namespace std;
4  class UserInterface
5  {
6  private:
7      Volume* v;
8  public:
9      void StartProgram() {
10         int n;
11         while (true) {
12             do
13             {
14                 cout << "Set MyFS Menu:\n"
15                     << "1. Create new volume\n"
16                     << "2. Access existed volume\n"
17                     << "3. Exit\n"
18                     << "Your choice: ";
19                 cin >> n;
20             } while (n < 1 || n > 3);
21             switch (n)
22             {
23             case 1: {
24                 v = new Volume;
25                 cout << "Successfully create new volume\n";
26                 v->setData(new Data);
27                 v->setClusterTable(new ClusterTable, v);

```

Đoạn code tạo menu

```

case 1: {
    v = new Volume;
    cout << "Successfully create new volume\n";
    v->setData(new Data);
    v->setClusterTable(new ClusterTable, v);
    v->getClusterTable()->writeFat(v);
    while (true) {
        int choice = showMenu();
        if (processChoice(choice)) { // processChoice(choice) = 1 thì exit ra Set MyFS Menu
            break;
        }
    }
    break;
}
case 2: {
    string volumeFileName, password;
    cout << "File system name: ";
    cin >> volumeFileName;
    cout << "Volume Password: ";
    cin >> password;
    if (getFileSize(volumeFileName) != UINT32_MAX) {
        // có tồn tại file system
        v = new Volume(volumeFileName, password);
        if (stringHash(password) == v->getBootSec()->getVolumePassword()) {
            cout << "Password is correct\n";
            // đọc hết bảng fat, entry
            v->setData(new Data);
            v->setClusterTable(new ClusterTable, v);
            v->getClusterTable()->readAllFat(v);

```

Đoạn code truy cập vào từng mục của menu

```

76     }
77 }
78 int showMenu()
79 {
80     int choice;
81     do
82     {
83         cout << "MyFS Management Menu:\n"
84             << "1. change Volume Password\n"
85             << "2. List Files\n"
86             << "3. Change File Password\n"
87             << "4. Import File\n"
88             << "5. Export File\n"
89             << "6. Delete File temporarily\n"
90             << "7. Delete File permanently\n"
91             << "8. Recover File\n"
92             << "9. Show volume left\n" // xem kích thước volume còn lại bao nhiêu
93             << "0. Exit\n"
94             << "Your choice: ";
95         cin >> choice;
96     } while (choice < 0 || choice > 8);
97     return choice;
98 }
99 bool processChoice(int choice)
100 {
101     string password, fileName;
102     bool ok = 0;

```

Đoạn code tạo danh sách các chức năng

```

98 }
99 bool processChoice(int choice)
100 {
101     string password, fileName;
102     bool ok = 0;
103     switch (choice)
104     {
105     case 1: {
106         cout << "New Password: ";
107         cin >> password;
108         v->getBootSec()->changeVolumePassword(password, v);
109         cout << "change Volume Password successfully\n";
110         break;
111     }
112     case 2: {
113         v->getData()->listAllFile();
114         cout << '\n';
115         break;
116     }
117     case 3: {
118         cout << "File name to import: ";
119         cin >> fileName;
120         break;
121     }
122     case 4: {
123         cout << "File name to import: ";

```

Đoạn code truy cập vào từng chức năng

1. Tạo / định dạng volume MyFS.Dat

```

2. Volume::Volume() {
3.     string fileName, password;
4.     cout << "File system name: "; // tên file system cần lưu
5.     cin >> fileName;
6.     if (findExistedFile(fileName)) {
7.         fileName = convertFileName(fileName);
8.         cout << "File system name has existed\n"
9.             << "File system name after change: " << fileName << '\n';
10.    }
11.    cout << "Set Volume Password: ";
12.    cin >> password;
13.    uint32_t volSize;
14.    cout << "Set volume size: ";
15.    cin >> volSize;
16.    this->setVolumeFileName(fileName);
17.    this->setBootSec(new BootSec(volSize, password));
18.    uint32_t size = this->getBootSec()->getSv() * 512;
19.    vector<uint8_t> nullBuffer(size, '\0');

```

```

20.     fstream fout(fileSystemName, ios::out | ios::binary);
21.     if (!fout)
22.     {
23.         cout << "Fail to open file\n";
24.         return;
25.     }
26.     fout.write(reinterpret_cast<const char*>(nullBuffer.data()), size);
27.     this->getBootSec()->writeBootSector(this); // viết bootsector vào disk
28. }

```

Đoạn code tạo định dạng cho file system

```

Set MyFS Menu:
1. Create new volume
2. Access existed volume
3. Exit
Your choice: 1
File system name: MyFS1.Dat
Set Volume Password: 1
Set volume size: 1024
Successfully create new volume

```

Đây là sau khi tạo volume

Startup	MyFS.Dat	MyFS1.Dat x																														
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000:0000	00	02	08	02	02	00	00	20	00	FC	07	00	00	A1	02	01
0000:0010	4D	79	46	53	00	00	00	00	00	00	00	00	00	00	00	00	M	y	F	S
0000:0020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:0030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:0050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:0060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:0070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:0080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:0090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:00A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:00B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:00C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:00D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:00E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:00F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:0100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:0110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:0120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:0130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:0140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:0150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:0160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:0170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:0180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:0190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:01A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:01B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:01C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:01D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:01E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000:01F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Sau khi tạo xong thì ghi luôn bootsector vào volume

2. Thiết lập /Đổi /Kiểm tra mật khẩu truy xuất MyFS

```
void BootSec::changeVolumePassword(string password, Volume* vol) {
    this->setVolumePassword(stringHash(password));
    // tìm vị trí volume password trong bootsector để ghi lại trong cả bootsector và phần backup bootsector
    uint32_t idx = 13;
    WriteByte<uint8_t>(idx, this->getVolumePassword(), vol);
    WriteByte<uint8_t>(idx + 512, this->getVolumePassword(), vol);
}
```

Đoạn code thay đổi mật khẩu volume

```
// mã hóa password bằng hash
int stringHash(const string& str) {
    const int tableSize = 256;
    const double a = 0.357840;
    int hashValue = 0;
    for (char ch : str)
        hashValue = static_cast<int>(hashValue * a + ch) % tableSize;

    return hashValue;
}
```

Mã hóa mật khẩu bằng hash, tạo ra key và lưu trữ key, khi kiểm tra mật khẩu thì hash mật khẩu nhập vào, nếu đúng với key đã lưu thì tiếp tục.

0000:0000	00 02 08 02 02 00 00 20 00 FC 07 00 00 97 02 01ü...-	0123456789ABCDEF	Type	Value
0000:0010	4D 79 46 53 00 00 00 00 00 00 00 00 00 00 00 00	MyFS.....		Binary	10010111
0000:0020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		Signed Byte	-105
0000:0030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		Unsigned B...	151

Trước khi đổi mật khẩu tại offset D: key = 151

```
File system name: MyFS1.Dat
Volume Password: sf
File System does not exist
Set MyFS Menu:
1. Create new volume
2. Access existed volume
3. Exit
Your choice: 1
File system name: MyFS1.Dat
Set Volume Password: haha
Set volume size: 1024
Successfully create new volume
```

```
Your choice: 1
New Password: hoandeptra
change Volume Password successfully
```

Đổi mật khẩu

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF	Type	Value
0000:0000	00	02	08	02	02	00	00	20	00	FC	07	00	00	A1	02	01ü....	Binary	10100001
0000:0010	4D	79	46	53	00	00	00	00	00	00	00	00	00	00	00	00	MyFS.....	Signed Byte	-95
0000:0020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	Unsigned B...	161
0000:0030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		

Sau khi đổi mật khẩu thì: **key = 151**

3. Liệt kê danh sách các tập tin trong MyFS

```
case 2: {
    v->getData()->listAllFile();
    cout << '\n';
    break;
}
```

```
void Data::listAllFile() {
    if (entries.size()) {
        cout << setw(3) << left << "Attribute"
            << setw(20) << right << "CreateTime"
            << setw(20) << right << "LastAccessTime"
            << setw(12) << right << "Length"
            << "Name" << '\n';
        for (auto i : fileNames) {
            // Attribute
            cout << setw(3) << left;
            bitset<8> bits(entries[i.second].first.getAttribute());
            for (int i = 7; i >= 0; --i) {
                bool bit = bits.test(i); // xét bit tại đó là 1 hay 0
                if (i == 7 && bit) {
                    // readonly
                    cout << "r";
                }
                else if (i == 6 && bit) {
                    // hidden
                    cout << "h";
                }
                else if (i == 2 && bit) {
                    // archive
                    cout << "a";
                }
            }
            // CreateTime
            uint32_t tmp = 0;
            cout << setw(20) << right
                << tmp + entries[i.second].first.getCreateDay().getDay() << "/"
                << tmp + entries[i.second].first.getCreateDay().getMonth() << "/"
                << tmp + entries[i.second].first.getCreateDay().getYear() << " "
                << tmp + entries[i.second].first.getCreateTime().getHour() << ":"
                << tmp + entries[i.second].first.getCreateTime().getMinute() << ":"
                << tmp + entries[i.second].first.getCreateTime().getSecond();
            // LastAccessTime
            cout << setw(20) << right
                << tmp + entries[i.second].first.getAccessDay().getDay() << "/"
                << tmp + entries[i.second].first.getAccessDay().getMonth() << "/"
                << tmp + entries[i.second].first.getAccessDay().getYear() << " "
                << tmp + entries[i.second].first.getAccessTime().getHour() << ":"
                << tmp + entries[i.second].first.getAccessTime().getMinute() << ":"
                << tmp + entries[i.second].first.getAccessTime().getSecond();
            // Length
            cout << setw(12) << right
                << entries[i.second].first.getFileSize();
            // Name
            cout << i.first << '\n';
        }
    }
}
```



```

else {
    cout << "Volume has no file\n";
}
}

```

Code liệt kê danh sách tập tin

4. Đặt /đổi mật khẩu truy xuất cho 1 tập tin trong MyFS

```

case 3: {
    cout << "File name to import: ";
    cin >> fileName;
    v->getData()->changeFilePassword(v, fileName);
    break;
}

```

```

void Data::changeFilePassword(Volume* vol, string fileName) {
    if (vol->getData()->checkFilePassword(fileName)) {
        string password;
        cout << "New Password: ";
        cin >> password;
        entries[fileNames[fileName]].first.setPassword(password);
        uint32_t idx = getPositionEntry(vol, fileName) + 2;
        WriteByte<uint8_t>(idx, entries[fileNames[fileName]].first.getPassword(), vol);
    }
    else {
        // Không đúng password
        return;
    }
}

```

Code thay đổi file password

5. Chép (Import) 1 tập tin từ bên ngoài vào MyFS

```

case 4: {
    cout << "File name to import: ";
    cin >> fileName;
    bool ok = 0;
    cout << "File Password (Yes=1|No=0): ";
    cin >> ok;
    if (ok) {
        cout << "Password: ";
        cin >> password;
        v->getData()->writeFile(fileName, v, password);
    }
    else {
        v->getData()->writeFile(fileName, v, "");
    }
    break;
}

```

```

void Data::writeFile(string fileName, Volume* vol, string password) {
    // Import file
}

```



```

vector<uint8_t> content = importFile(fileName);
// Viết entry
writeEntry(fileName, vol, password);
// Xác định kích thước của dữ liệu
uint32_t dataSize = 0;
if (content.size()) {
    dataSize = entries[entries.size() - 1].first.getFileSize(); // Kích thước của dữ liệu
    // được trả bởi content
}
uint32_t idx = (vol->getBootSec()->getSb() + vol->getBootSec()->getNf() *
vol->getBootSec()->getSf() + entries[entries.size() - 1].first.getStartCluster() *
vol->getBootSec()->getSc()) * 512;
// Ghi nội dung tập tin vào volume
WriteCluster(idx, vol, content, dataSize);
uint32_t numClustersToAdd = static_cast<uint32_t>(ceil(float(dataSize) / (512 *
vol->getBootSec()->getSc())));
vol->changeNumEmptyCluster(-static_cast<int32_t>(numClustersToAdd));
}

```

Code import file

6. Chép (Output) 1 tập tin trong MyFS ra ngoài

```

void Data::exportFile(string fileName, Volume* vol) {
// vị trí để lấy file trong disk
if (fileNames.find(fileName) != fileNames.end()) { // check xem có tồn tại file trong disk
    // tìm các cluster lưu trữ file trong fat
    if (checkFilePassword(fileName)) {
        // change date time access

        vector<uint32_t> cluster;
        cluster.push_back(entries[fileNames[fileName]].first.getStartCluster());
        for (int i = entries[fileNames[fileName]].first.getStartCluster(); i <
vol->getClusterTable()->getElement().size() - 1; ++i) {
            if (vol->getClusterTable()->getElement()[i] == EOC) {
                // Khi đến cluster kết thúc file thì ngừng
                break;
            }
            cluster.push_back(vol->getClusterTable()->getElement()[i]);
            i = vol->getClusterTable()->getElement()[i];
        }
        // Xuất file ra ngoài
        fstream fout(fileName, ios::in | ios::out | ios::binary);
        if (!fout)
        {
            cout << "Fail to open file\n";
            return;
        }
        for (int i = 0; i < cluster.size(); ++i) {
            uint8_t* content = ReadCluster<uint8_t>((vol->getBootSec()->getSb() +
vol->getBootSec()->getNf() * vol->getBootSec()->getSf() + cluster[i] *
vol->getBootSec()->getSc()) * 512, vol, 1);
            fout.write(reinterpret_cast<const char*>(content), sizeof(content));
        }
        fout.close();
    }
    else {
        return;
    }
}
else {
    cout << "File does not exist\n";
}
}
}

```

Code export file

7. Xóa 1 tập tin trong MyFS (chưa làm)

Bổ sung:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
00	00	02	08	02	02	00	00	20	00	FC	07	00	00	31	02	01ü...1..
10	4D	79	46	53	00	00	00	00	00	00	00	00	00	00	00	00	MyFS.....
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Đây là đoạn byte sau khi lưu bootsector

0020:13E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020:13F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020:1400	46	30	00	00	00	00	00	44	41	54	23	A1	35	0D	15	13	F0.....DAT#i5...
0020:1410	03	00	35	0D	15	13	03	00	03	00	00	00	00	10	00	00	..5.....
0020:1420	41	46	30	2E	44	61	74	00	00	00	00	00	00	00	00	00	AF0.Dat.....
0020:1430	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020:1440	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020:1450	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020:1460	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020:1470	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020:1480	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020:1490	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020:14A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020:14B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020:14C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020:14D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020:14E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020:14F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020:1500	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020:1510	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Find Results	
Address	Value
Found 2 occurrences of 'F0'.	
201400h	F0
201421h	F0

Đây là đoạn byte entry sau khi lưu file F0.Dat vào

VI. References

- Batch file – Programming tutorial.* (n.d.). Retrieved from trytoprogram: <https://www.trytoprogram.com/batch-file/>
- Dat, N. H. (2019 , 10 19). *Encoding , Encryption và Hashing.* Retrieved from viblo.asia: <https://viblo.asia/p/encoding-encryption-va-hashing-gDVK2pJmLj>
- linkedin.com.* (2023, Sep 22). Retrieved from How can you ensure data consistency and integrity in file system algorithms?: <https://www.linkedin.com/advice/0/how-can-you-ensure-data-consistency-integrity>
- Stokes, I. (2020, March 31). *Encryption Algorithms - what are they, and how do they secure your data?* Retrieved from toptenreviews.com: <https://www.toptenreviews.com/encryption-algorithms>