



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

# **BÁO CÁO ĐỒ ÁN**

---

**MÔN HỌC: HỆ ĐIỀU HÀNH – CQ2018/4**

**GIÁO VIÊN HƯỚNG DẪN: THÁI HÙNG VĂN – ĐẶNG TRẦN MINH HẬU**

**THỰC HIỆN:**

**NGUYỄN GIA VĨ – 18120260**

**LÊ ĐỨC THIÊN – 19120664**

**TRẦN VĂN TÌNH - 19120686**

# A. THÔNG TIN CHUNG:

## 1. Thông tin bài làm:

- Ngôn ngữ lập trình: C++, Batch Script
- Môi trường lập trình: Visual Studio, Terminal

## 2. Thành viên và Phân công:

Họ và Tên	MSSV	Phân công	Mức độ hoàn thành
Nguyễn Gia Vĩ	18120260	Câu 1	85%
Lê Đức Thiện	19120664	Câu 2	95%
Trần Văn Tình	19120686		95%

## 3. Tham khảo:

### Câu 1:

- 1] TL bổ sung - môn HĐH - Tổ chức Hệ thống Tập tin GV: Thái Hùng Văn - ĐH KHTN – 06.06
- [2] <https://stackoverflow.com/questions/13805187/how-to-set-a-variable-inside-a-loop-for-f>

### Câu 2A:

- [1] [https://www.youtube.com/watch?v=q1r\\_s72mBc4&t=183s](https://www.youtube.com/watch?v=q1r_s72mBc4&t=183s)
- [2] <http://web.cs.ucla.edu/classes/fall10/cs111/scribe/11a/>
- [3] [https://en.wikipedia.org/wiki/Unix\\_filesystem](https://en.wikipedia.org/wiki/Unix_filesystem)
- [4] [CS 110 Lecture 3: Unix v6 Filesystem - YouTube](#)

### Câu 2B:

- [1] <https://www.youtube.com/watch?v=n2AAhiujAqs&t=3749s>
- [2] <https://docs.oracle.com/cd/E19795-01/817-7721-10/chapter2.html>

## B. CÂU 1:

Tiêu chí	Mức độ hoàn thành
Câu A	Hoàn thành
Câu B	Hoàn thành
Câu C	Hoàn thành
Câu D	Hoàn thành
Câu E	Chưa hoàn thành

### I- XÁC ĐỊNH CÁC THÔNG SỐ QUAN TRỌNG (CÂU A)

Volume của em có 128 byte đầu của Boot Sector như sau:

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  EB 58 90 4D 53 44 4F 53 35 2E 30 00 02 02 BE 20
00000010  02 00 00 00 00 F8 00 00 3F 00 FF 00 00 40 C9 37
00000020  00 00 20 00 A1 1F 00 00 00 00 00 00 02 00 00 00
00000030  01 00 06 00 00 00 00 00 00 00 00 00 00 00 00 00
00000040  80 00 29 89 CE C5 B8 4E 4F 20 4E 41 4D 45 20 20
00000050  20 20 46 41 54 33 32 20 20 20 33 C9 8E D1 BC F4
00000060  7B 8E C1 8E D9 BD 00 7C 88 56 40 88 4E 02 8A 56
00000070  40 B4 41 BB AA 55 CD 13 72 10 81 FB 55 AA 75 0A
```

- Số byte của một sector:

<Số nguyên 2 byte tại offset Bh> = 0200h = 512d (byte)

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  EB 58 90 4D 53 44 4F 53 35 2E 30 00 02 02 BE 20
00000010  02 00 00 00 00 F8 00 00 3F 00 FF 00 00 40 C9 37
00000020  00 00 20 00 A1 1F 00 00 00 00 00 00 02 00 00 00
00000030  01 00 06 00 00 00 00 00 00 00 00 00 00 00 00 00
00000040  80 00 29 89 CE C5 B8 4E 4F 20 4E 41 4D 45 20 20
00000050  20 20 46 41 54 33 32 20 20 20 33 C9 8E D1 BC F4
00000060  7B 8E C1 8E D9 BD 00 7C 88 56 40 88 4E 02 8A 56
00000070  40 B4 41 BB AA 55 CD 13 72 10 81 FB 55 AA 75 0A
```

- Kích thước volume theo MegaByte và MebiByte:

**SV** = <Số nguyên 4 byte tại offset 20h> = 00200000h = 2097152d (sector)

= 2097152 \* 512 = 1073741824 Byte

- Theo MegaByte: 1073741824 / 1000 / 1000 = 1073.741824 MegaByte
- Theo MebiByte: 1073741824 / 1024 / 1024 = 1024 MebiByte

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	EB	58	90	4D	53	44	4F	53	35	2E	30	00	02	02	BE	20
00000010	02	00	00	00	00	F8	00	00	3F	00	FF	00	00	40	C9	37
00000020	00	00	20	00	A1	1F	00	00	00	00	00	00	02	00	00	00
00000030	01	00	06	00	00	00	00	00	00	00	00	00	00	00	00	00
00000040	80	00	29	89	CE	C5	B8	4E	4F	20	4E	41	4D	45	20	20
00000050	20	20	46	41	54	33	32	20	20	20	33	C9	8E	D1	BC	F4
00000060	7B	8E	C1	8E	D9	BD	00	7C	88	56	40	88	4E	02	8A	56
00000070	40	B4	41	BB	AA	55	CD	13	72	10	81	FB	55	AA	75	0A

- Số sector trước FAT:

- Ta có trước FAT là boot sector.

⇒ Số sector trước FAT là số sector của vùng boot sector.

**SB** = <Số nguyên 2 byte tại offset Eh> = 20BEh = 8382d (sector).

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	EB	58	90	4D	53	44	4F	53	35	2E	30	00	02	02	BE	20
00000010	02	00	00	00	00	F8	00	00	3F	00	FF	00	00	40	C9	37
00000020	00	00	20	00	A1	1F	00	00	00	00	00	00	02	00	00	00
00000030	01	00	06	00	00	00	00	00	00	00	00	00	00	00	00	00
00000040	80	00	29	89	CE	C5	B8	4E	4F	20	4E	41	4D	45	20	20
00000050	20	20	46	41	54	33	32	20	20	20	33	C9	8E	D1	BC	F4
00000060	7B	8E	C1	8E	D9	BD	00	7C	88	56	40	88	4E	02	8A	56
00000070	40	B4	41	BB	AA	55	CD	13	72	10	81	FB	55	AA	75	0A

- Số sector của một bảng FAT và số bảng FAT:

- Số sector của một bảng FAT:

**SF** = <Số nguyên 4 byte tại offset 24h> = 00001FA1 = 8097d (sector).

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	EB	58	90	4D	53	44	4F	53	35	2E	30	00	02	02	BE	20
00000010	02	00	00	00	00	F8	00	00	3F	00	FF	00	00	40	C9	37
00000020	00	00	20	00	A1	1F	00	00	00	00	00	00	02	00	00	00
00000030	01	00	06	00	00	00	00	00	00	00	00	00	00	00	00	00
00000040	80	00	29	89	CE	C5	B8	4E	4F	20	4E	41	4D	45	20	20
00000050	20	20	46	41	54	33	32	20	20	20	33	C9	8E	D1	BC	F4
00000060	7B	8E	C1	8E	D9	BD	00	7C	88	56	40	88	4E	02	8A	56
00000070	40	B4	41	BB	AA	55	CD	13	72	10	81	FB	55	AA	75	0A

- Số bảng FAT:

**NF** = <Số nguyên 1 byte tại offset 10h> = 02h = 2d (bảng).

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	EB	58	90	4D	53	44	4F	53	35	2E	30	00	02	02	BE	20
00000010	02	00	00	00	00	F8	00	00	3F	00	FF	00	00	40	C9	37
00000020	00	00	20	00	A1	1F	00	00	00	00	00	00	02	00	00	00
00000030	01	00	06	00	00	00	00	00	00	00	00	00	00	00	00	00
00000040	80	00	29	89	CE	C5	B8	4E	4F	20	4E	41	4D	45	20	20
00000050	20	20	46	41	54	33	32	20	20	20	33	C9	8E	D1	BC	F4
00000060	7B	8E	C1	8E	D9	BD	00	7C	88	56	40	88	4E	02	8A	56
00000070	40	B4	41	BB	AA	55	CD	13	72	10	81	FB	55	AA	75	0A

- Số entry (hiện tại) của RDET:

- Với FAT32, Windows không tổ chức RDET trên SYSTEM, mà coi RDET như SDET – và lưu trên vùng DATA. [1]
- Bảng RDET sẽ nằm tại cluster trống đầu tiên (cluster 2 nếu nó không bị hư) & có kích thước bằng kích thước cluster. [1]

⇒ RDET = 1 cluster = 2 sector = 2 \* 512 byte = 1024 byte

- Ta có: 1 entry = 32 byte (Quy chuẩn chung của DOS và Windows)

⇒ RDET = 1024 / 32 = 32 entry

- Số sector của một cluster:

**SC** = <Số nguyên 1 byte tại offset Dh> = 02h = 2d (sector)

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	EB	58	90	4D	53	44	4F	53	35	2E	30	00	02	02	BE	20
00000010	02	00	00	00	00	F8	00	00	3F	00	FF	00	00	40	C9	37
00000020	00	00	20	00	A1	1F	00	00	00	00	00	00	02	00	00	00
00000030	01	00	06	00	00	00	00	00	00	00	00	00	00	00	00	00
00000040	80	00	29	89	CE	C5	B8	4E	4F	20	4E	41	4D	45	20	20
00000050	20	20	46	41	54	33	32	20	20	20	33	C9	8E	D1	BC	F4
00000060	7B	8E	C1	8E	D9	BD	00	7C	88	56	40	88	4E	02	8A	56
00000070	40	B4	41	BB	AA	55	CD	13	72	10	81	FB	55	AA	75	0A

- Số cluster của volume:

- Cluster chỉ tồn tại trên vùng dữ liệu (vùng DATA) – nơi chứa nội dung tập tin. [1]
- Nên số cluster của volume sẽ là số cluster của vùng Data.
- Số sector của vùng Data = Số sector của volume - Số sector của vùng System

⇒ SD = SV - SS (1)

- Số sector của vùng System = Số sector của boot sector + (Số sector của bảng FAT \* Số bảng FAT)

$$\Rightarrow SS = SB + (SF * NF) \text{ (2)}$$

- Từ (1) và (2) suy ra:

$$SD = SV - SB - (SF * NF) = 2097152 - 8382 - (8097 * 2) = 2072576 \text{ (sector)}$$

Vậy số sector của Data: 2072576

- Và 1 cluster = 2 sector

$$\Rightarrow \text{Số cluster của Data} = \text{Số cluster của volume} = 2072576 / 2 = 1036288 \text{ (cluster)}$$

- Công thức tính vị trí của cluster (theo sector)

- Nếu vùng DATA có SD sector & bắt đầu tại sector SS, mỗi cluster chiếm SC sector, cluster đầu tiên được đánh chỉ số là FC, thì ta sẽ có:
- Cluster K sẽ bắt đầu tại sector:  $SS + (K - FC) * SC$ . [1]
- Gọi A là vị trí bắt đầu sector, ta có:  $A = SS + (K - FC) * SC$ .

$$\Rightarrow K = (A - SS) / SC + FC \text{ (Công thức tính vị trí cluster theo vị trí sector)}$$

- Vì bảng RDET sẽ nằm tại cluster trống đầu tiên ở vùng Data như đã nói ở trên. Nên vị trí cluster bắt đầu của RDET sẽ tương ứng với vị trí cluster bắt đầu của vùng Data:

$$FC = \text{<Số nguyên 4 byte tại offset 2Ch>} = 00000002h = 2$$

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	EB	58	90	4D	53	44	4F	53	35	2E	30	00	02	02	BE	20
00000010	02	00	00	00	00	F8	00	00	3F	00	FF	00	00	40	C9	37
00000020	00	00	20	00	A1	1F	00	00	00	00	00	00	02	00	00	00
00000030	01	00	06	00	00	00	00	00	00	00	00	00	00	00	00	00
00000040	80	00	29	89	CE	C5	B8	4E	4F	20	4E	41	4D	45	20	20
00000050	20	20	46	41	54	33	32	20	20	20	33	C9	8E	D1	BC	F4
00000060	7B	8E	C1	8E	D9	BD	00	7C	88	56	40	88	4E	02	8A	56
00000070	40	B4	41	BB	AA	55	CD	13	72	10	81	FB	55	AA	75	0A

- Số lượng sector của vùng system sẽ tương đương với vị trí bắt đầu sector tại vùng Data do sector được đánh số từ 0:

$$SS = SB + (SF * NF) = 8382 + (8097 * 2) = 24576 \text{ (sector)}$$

$$\text{Vậy } K = (A - SS) / SC + FC = (A - 24576) / 2 + 2$$

## II- VIẾT CHƯƠNG TRÌNH TẠO FILE (CÂU B)

### 1. Trình bày thuật toán:

- Mời người dùng nhập vào số lượng file cần tạo: N
- Tạo vòng for bắt đầu đi từ  $i = 1$  đến N, với mỗi vòng lặp:
- Tạo file với tên tương ứng với giá trị  $(i - 1)$
- Đưa vào file các dòng văn bản là giá trị  $(i - 1)$
- Nếu  $(i - 1)$  chẵn thì:
  - Kích thước file bằng 2048 byte
- Nếu  $(i - 1)$  lẻ thì:
  - Kích thước file bằng 1024 byte

### 2. Cài đặt:

```
1 @echo off
2
3 set /P n=Please, enter the number of files:
4
5 FOR /L %%X IN (1,1,%%n) DO call :Label1 %%X
6 goto End
7
8 :Label1
9     set /A i=%%X-1
10    echo %%X >> G:\F%%i%.dat
11    echo %%X >> G:\F%%i%.dat
12    echo %%X >> G:\F%%i%.dat
13    echo %%X >> G:\F%%i%.dat
14    echo %%X >> G:\F%%i%.dat
15    echo %%X >> G:\F%%i%.dat
16    echo %%X >> G:\F%%i%.dat
17
18    set /A isEven=%%i %% 2
19
20    if %isEven%==0 (
21        fsutil file seteof G:\F%%i%.dat 2048
22    ) else (
23        fsutil file seteof G:\F%%i%.dat 1024
24    )
25    goto :eof
26
27 :End
28
29 pause
```

\* Ở source code em đã có note giải thích đầy đủ - Đây là bản lược bớt comment để tránh dài ảnh



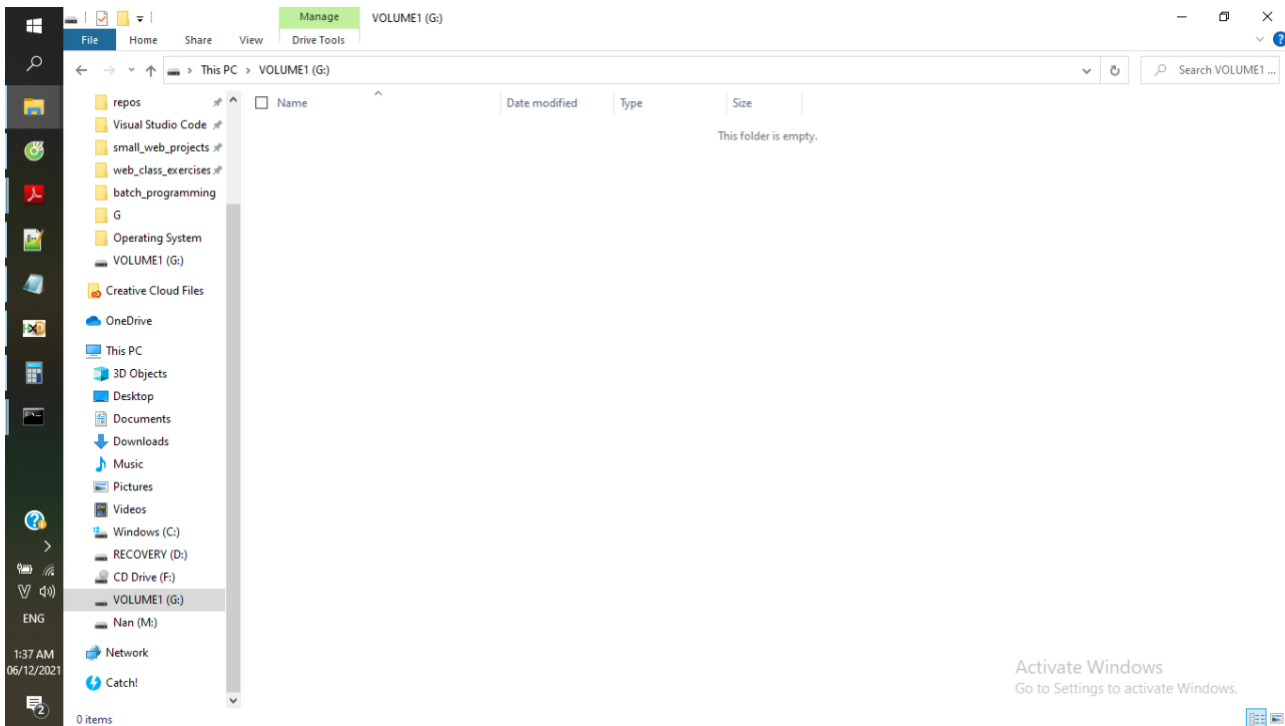
### 3. Giải thích thêm code:

Lý do dùng label mà không viết code bên trong scope của vòng lặp for:

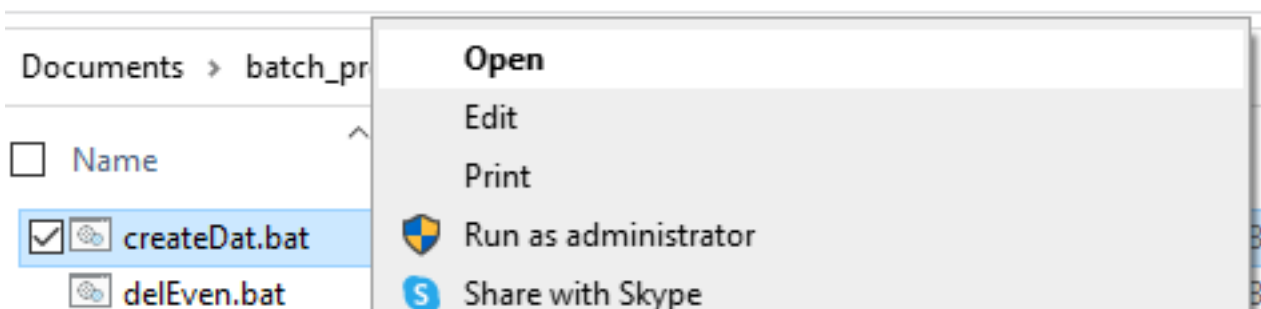
- Do vòng lặp for không hỗ trợ tạo biến hay set giá trị bên trong nó. Nhưng em cần một biến đếm để lưu trữ vị trí của vòng lặp và để xem xét tạo file thứ i, và một biến để kiểm tra file chắc là để tạo dung lượng phù hợp yêu cầu đề bài. Vòng lặp được tham khảo tại [2]

### 4. Chạy chương trình:

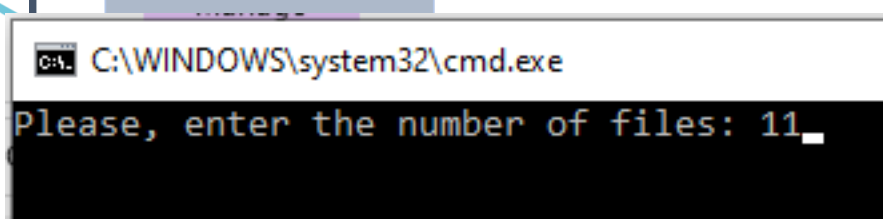
- B0: Kiểm tra ổ đĩa trước khi chạy chương trình.



- B1: Click đúp chuột trái vào file delEven.bat hoặc click chuột phải vào file chọn Open.



- B2: Nhập số lượng file muốn khởi tạo và nhấn enter.
















- B3: Nhấn phím bất kỳ khi chương trình đã kết thúc.


```
C:\WINDOWS\system32\cmd.exe

Please, enter the number of files: 11
File G:\F0.dat eof set
File G:\F1.dat eof set
File G:\F2.dat eof set
File G:\F3.dat eof set
File G:\F4.dat eof set
File G:\F5.dat eof set
File G:\F6.dat eof set
File G:\F7.dat eof set
File G:\F8.dat eof set
File G:\F9.dat eof set
File G:\F10.dat eof set
Press any key to continue . . .
```

- B4: Kiểm tra kết quả.

VOLUME1 (G:)


<input type="checkbox"/> Name	Date modified	Type	Size
 F0.dat	06/12/2021 1:40 AM	DAT File	2 KB
 F1.dat	06/12/2021 1:40 AM	DAT File	1 KB
 F2.dat	06/12/2021 1:40 AM	DAT File	2 KB
 F3.dat	06/12/2021 1:40 AM	DAT File	1 KB
 F4.dat	06/12/2021 1:40 AM	DAT File	2 KB
 F5.dat	06/12/2021 1:40 AM	DAT File	1 KB
 F6.dat	06/12/2021 1:40 AM	DAT File	2 KB
 F7.dat	06/12/2021 1:40 AM	DAT File	1 KB
 F8.dat	06/12/2021 1:40 AM	DAT File	2 KB
 F9.dat	06/12/2021 1:40 AM	DAT File	1 KB
 F10.dat	06/12/2021 1:40 AM	DAT File	2 KB



F0.dat - Notepad

File Edit Format View Help


```
0
0
0
0
0
0
0
0
```



F1.dat - Notepad

File Edit Format View Help

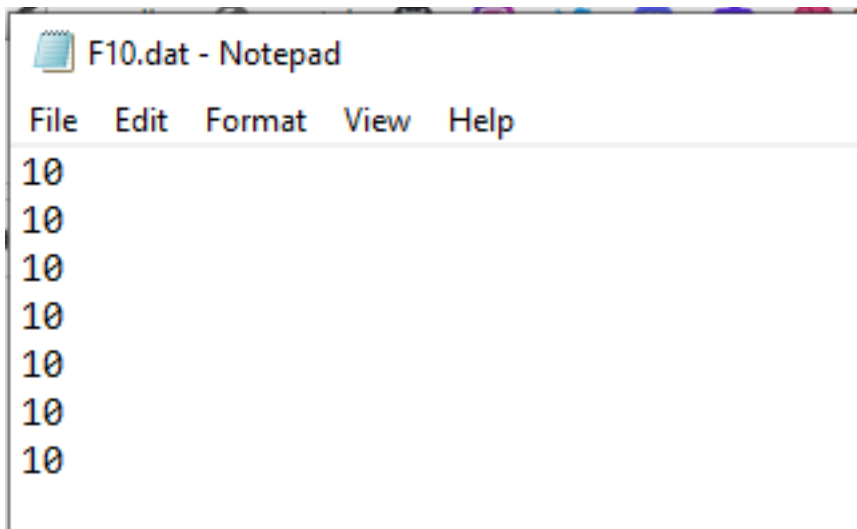
```
1
1
1
1
1
1
1
1
```



F2.dat - Notepad

File Edit Format View Help

```
2
2
2
2
2
2
2
2
```



### III- DỰ ĐOÁN (CÓ LÝ GIẢI) SỐ CLUSTER CỦA RDET (CÂU C)

#### 1. Khi N = 11:

- Số cluster của RDET sẽ là 1.

#### Lý giải

- Sau khi format, RDET mặc định sẽ có 1 cluster tương đương 32 entry (như đã phân tích ở câu A).
- Các tập tin cần tạo chỉ bao gồm các tên ngắn, nên không cần entry phụ để chứa tên dài

⇒ Mỗi file sẽ chỉ cần 1 entry để chứa thông tin.

⇒ Với N = 11 thì hệ điều hành chỉ cần 11 entry để chứa thông tin file, vẫn còn nhỏ hơn rất nhiều so với 32 entry mặc định của RDET.

**Do đó,** RDET không cần tạo thêm cluster để lưu thông tin file mà chỉ cần ở mặc định là đã đủ.

#### Chứng minh:

- Chỉ số cluster bắt đầu đầu của RDET:

<Số nguyên 4 byte tại offset 2Ch> = 00000002h = 2d

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	EB	58	90	4D	53	44	4F	53	35	2E	30	00	02	02	BE	20
00000010	02	00	00	00	00	F8	00	00	3F	00	FF	00	00	40	C9	37
00000020	00	00	20	00	A1	1F	00	00	00	00	00	00	02	00	00	00
00000030	01	00	06	00	00	00	00	00	00	00	00	00	00	00	00	00
00000040	80	00	29	45	C9	48	6E	4E	4F	20	4E	41	4D	45	20	20
00000050	20	20	46	41	54	33	32	20	20	20	33	C9	8E	D1	BC	F4
00000060	7B	8E	C1	8E	D9	BD	00	7C	88	56	40	88	4E	02	8A	56
00000070	40	B4	41	BB	AA	55	CD	13	72	10	81	FB	55	AA	75	0A

- Đến vị trí cluster của RDET để kiểm tra bằng công thức đã đề cập ở câu A:

$A = SS + (K - FC) * SC = 24576 + (K - 2) * 2$  (với K là chỉ số cluster và A là chỉ số sector)

$K = 2 \Rightarrow A = 24576$

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text	
00C00000	56	4F	4C	55	4D	45	31	20	20	20	20	08	00	00	00	00	VOLUME1 .....	Sector 24,576
00C00010	00	00	00	00	00	00	EE	B4	85	53	00	00	00	00	00	00	.....i'...S.....	
00C00020	42	20	00	49	00	6E	00	66	00	6F	00	0F	00	72	72	00	B .I.n.f.o...rr.	
00C00030	6D	00	61	00	74	00	69	00	6F	00	00	00	6E	00	00	00	m.a.t.i.o...n...	
00C00040	01	53	00	79	00	73	00	74	00	65	00	0F	00	72	6D	00	.S.y.s.t.e...rm.	
00C00050	20	00	56	00	6F	00	6C	00	75	00	00	00	6D	00	65	00	.V.o.l.u...m.e.	
00C00060	53	59	53	54	45	4D	7E	31	20	20	20	16	00	89	ED	B4	SYSTEM~1 ..%i'	
00C00070	85	53	85	53	00	00	EE	B4	85	53	03	00	00	00	00	00	...S...i'...S.....	
00C00080	46	30	20	20	20	20	20	20	44	41	54	20	10	87	F3	B4	F0 DAT .#ó'	
00C00090	85	53	85	53	00	00	F4	B4	85	53	06	00	00	08	00	00	...S...ô'...S.....	
00C000A0	46	31	20	20	20	20	20	44	41	54	20	10	8B	F3	B4		F1 DAT .<ó'	
00C000B0	85	53	85	53	00	00	F4	B4	85	53	08	00	00	04	00	00	...S...ô'...S.....	
00C000C0	46	32	20	20	20	20	20	44	41	54	20	10	8F	F3	B4		F2 DAT ..ó'	
00C000D0	85	53	85	53	00	00	F4	B4	85	53	09	00	00	08	00	00	...S...ô'...S.....	
00C000E0	46	33	20	20	20	20	20	44	41	54	20	10	92	F3	B4		F3 DAT .'ó'	
00C000F0	85	53	85	53	00	00	F4	B4	85	53	0B	00	00	04	00	00	...S...ô'...S.....	
00C00100	46	34	20	20	20	20	20	44	41	54	20	10	94	F3	B4		F4 DAT ."ó'	
00C00110	85	53	85	53	00	00	F4	B4	85	53	0C	00	00	08	00	00	...S...ô'...S.....	
00C00120	46	35	20	20	20	20	20	44	41	54	20	10	98	F3	B4		F5 DAT .~ó'	
00C00130	85	53	85	53	00	00	F4	B4	85	53	0E	00	00	04	00	00	...S...ô'...S.....	
00C00140	46	36	20	20	20	20	20	44	41	54	20	10	9B	F3	B4		F6 DAT .>ó'	
00C00150	85	53	85	53	00	00	F4	B4	85	53	0F	00	00	08	00	00	...S...ô'...S.....	
00C00160	46	37	20	20	20	20	20	44	41	54	20	10	9E	F3	B4		F7 DAT .žó'	
00C00170	85	53	85	53	00	00	F4	B4	85	53	11	00	00	04	00	00	...S...ô'...S.....	
00C00180	46	38	20	20	20	20	20	44	41	54	20	10	A1	F3	B4		F8 DAT .;ó'	
00C00190	85	53	85	53	00	00	F4	B4	85	53	12	00	00	08	00	00	...S...ô'...S.....	
00C001A0	46	39	20	20	20	20	20	44	41	54	20	10	A4	F3	B4		F9 DAT .#ó'	
00C001B0	85	53	85	53	00	00	F4	B4	85	53	14	00	00	04	00	00	...S...ô'...S.....	
00C001C0	46	31	30	20	20	20	20	44	41	54	20	10	A7	F3	B4		F10 DAT .šó'	
00C001D0	85	53	85	53	00	00	F4	B4	85	53	15	00	00	08	00	00	...S...ô'...S.....	
00C001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
00C001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	

- Entry đầu tiên của RDET cho thấy tên volumn.
- Entry thứ 2-3-4 là thông tin của thư mục system volumn information.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00C00000	56	4F	4C	55	4D	45	31	20	20	20	20	08	00	00	00	00	VOLUME1 .....
00C00010	00	00	00	00	00	00	EE	B4	85	53	00	00	00	00	00	00	.....i'...S.....
00C00020	42	20	00	49	00	6E	00	66	00	6F	00	0F	00	72	72	00	B .I.n.f.o...rr.
00C00030	6D	00	61	00	74	00	69	00	6F	00	00	00	6E	00	00	00	m.a.t.i.o...n...
00C00040	01	53	00	79	00	73	00	74	00	65	00	0F	00	72	6D	00	.S.y.s.t.e...rm.
00C00050	20	00	56	00	6F	00	6C	00	75	00	00	00	6D	00	65	00	.V.o.l.u...m.e.
00C00060	53	59	53	54	45	4D	7E	31	20	20	20	16	00	89	ED	B4	SYSTEM~1 ..%i'
00C00070	85	53	85	53	00	00	EE	B4	85	53	03	00	00	00	00	00	...S...i'...S.....
00C00080	46	30	20	20	20	20	20	20	44	41	54	20	10	87	F3	B4	F0 DAT .#ó'

- Với thông tin ở entry chính chỉ ra rằng:

Cluster bắt đầu của nó là:

Word cao: <Số nguyên 2 byte tại offset 14Ch> = 0000h

00C00060	53	59	53	54	45	4D	7E	31	20	20	20	16	00	89	ED	B4	SYSTEM~1 ..%i'
00C00070	85	53	85	53	00	00	EE	B4	85	53	03	00	00	00	00	00	...S...i'...S.....

Word thập: <Số nguyên 2 byte tại offset 1ACh> = 0003h

```
00C00060 53 59 53 54 45 4D 7E 31 20 20 20 16 00 89 ED B4 SYSTEM~1 ..%í'
00C00070 85 53 85 53 00 00 EE B4 85 53 03 00 00 00 00 00 ...S...S...í'...S....
```

⇒ Cluster bắt đầu: 00000003h = 3d

**Kết luận:** Cluster bắt đầu của RDET là 02 mà cluster bắt đầu của system volumn information là 03  
⇒ RDET chỉ có 1 cluster.

## 2. Khi N = 2021:

- Số cluster của RDET sẽ là 64

### Lý giải:

- Khi tạo ra 2021 file thì tương ứng RDET cũng sẽ cần 2021 entry chính để lưu trữ thông tin cho các file này.
- Do các file tên chỉ gồm cao nhất là F2021 ⇒ 5 ký tự không dấu nên hệ điều hành sẽ không cần tạo entry phụ để lưu trữ tên dài.
- Ta có: 1 entry là 32 byte

⇒ 2021 entry + 4 entry (1 lưu trữ tên và 3 để lưu thư mục hệ thống) = 2025 \* 32  
= 64800 byte

- Lại có: 1 cluster = 2 sector = 1024 byte

⇒ Số cluster:  $64800 / 1024 = 63.28125$

**Vậy,** cần có ít nhất 64 cluster để lưu trữ.

### Chứng minh:

- Do RDET được lưu trữ ở vùng Data với số lượng cluster lớn nên chỉ có thể chứng minh bằng các duyệt qua bảng FAT tại vị trí sector 8382 (là số lượng sector của vùng trước FAT đã tìm ở câu A).
- Với bảng FAT ở chỉ mục cluster thứ 2 trở đến cluster 30h = 48d

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text	
00417C00	F8	FF	FF	0F	FF	FF	FF	FF	30	00	00	00	FF	FF	FF	0F	øÿÿ.ÿÿÿÿ0...ÿÿÿ.	Sector 8,382
00417C10	FF	FF	FF	0F	FF	FF	FF	0F	07	00	00	00	FF	FF	FF	0F	ÿÿÿ.ÿÿÿ....ÿÿÿ.	
00417C20	FF	FF	FF	0F	0A	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	ÿÿÿ....ÿÿÿ.ÿÿÿ.	
00417C30	0D	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	10	00	00	00	...ÿÿÿ.ÿÿÿ....	
00417C40	FF	FF	FF	0F	FF	FF	FF	0F	13	00	00	00	FF	FF	FF	0F	ÿÿÿ.ÿÿÿ....ÿÿÿ.	
00417C50	FF	FF	FF	0F	16	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	ÿÿÿ....ÿÿÿ.ÿÿÿ.	
00417C60	19	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	1C	00	00	00	...ÿÿÿ.ÿÿÿ....	
00417C70	FF	FF	FF	0F	FF	FF	FF	0F	1F	00	00	00	FF	FF	FF	0F	ÿÿÿ.ÿÿÿ....ÿÿÿ.	
00417C80	FF	FF	FF	0F	22	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	ÿÿÿ."...ÿÿÿ.ÿÿÿ.	
00417C90	25	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	28	00	00	00	%...ÿÿÿ.ÿÿÿ.(...	
00417CA0	FF	FF	FF	0F	FF	FF	FF	0F	2B	00	00	00	FF	FF	FF	0F	ÿÿÿ.ÿÿÿ.+...ÿÿÿ.	
00417CB0	FF	FF	FF	0F	2E	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	ÿÿÿ....ÿÿÿ.ÿÿÿ.	
00417CC0	61	00	00	00	32	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	a...2...ÿÿÿ.ÿÿÿ.	
00417CD0	35	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	38	00	00	00	5...ÿÿÿ.ÿÿÿ.8...	
00417CE0	FF	FF	FF	0F	FF	FF	FF	0F	3B	00	00	00	FF	FF	FF	0F	ÿÿÿ.ÿÿÿ.;...ÿÿÿ.	
00417CF0	FF	FF	FF	0F	3E	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	ÿÿÿ.>...ÿÿÿ.ÿÿÿ.	
00417D00	41	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	44	00	00	00	A...ÿÿÿ.ÿÿÿ.D...	
00417D10	FF	FF	FF	0F	FF	FF	FF	0F	47	00	00	00	FF	FF	FF	0F	ÿÿÿ.ÿÿÿ.G...ÿÿÿ.	
00417D20	FF	FF	FF	0F	4A	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	ÿÿÿ.J...ÿÿÿ.ÿÿÿ.	
00417D30	4D	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	50	00	00	00	M...ÿÿÿ.ÿÿÿ.P...	
00417D40	FF	FF	FF	0F	FF	FF	FF	0F	53	00	00	00	FF	FF	FF	0F	ÿÿÿ.ÿÿÿ.S...ÿÿÿ.	
00417D50	FF	FF	FF	0F	56	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	ÿÿÿ.V...ÿÿÿ.ÿÿÿ.	
00417D60	59	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	5C	00	00	00	Y...ÿÿÿ.ÿÿÿ.\...	
00417D70	FF	FF	FF	0F	FF	FF	FF	0F	5F	00	00	00	FF	FF	FF	0F	ÿÿÿ.ÿÿÿ._...ÿÿÿ.	
00417D80	FF	FF	FF	0F	92	00	00	00	63	00	00	00	FF	FF	FF	0F	ÿÿÿ.'...c...ÿÿÿ.	
00417D90	FF	FF	FF	0F	66	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	ÿÿÿ.f...ÿÿÿ.ÿÿÿ.	

- Tại vị trí cluster thứ 48 lại trở đến chỉ mục cluster thứ 61h = 97d

00417D50	FF	FF	FF	0F	56	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	ÿÿÿ.V...ÿÿÿ.ÿÿÿ.
00417D60	59	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	5C	00	00	00	Y...ÿÿÿ.ÿÿÿ.\...
00417D70	FF	FF	FF	0F	FF	FF	FF	0F	5F	00	00	00	FF	FF	FF	0F	ÿÿÿ.ÿÿÿ._...ÿÿÿ.
00417D80	FF	FF	FF	0F	92	00	00	00	63	00	00	00	FF	FF	FF	0F	ÿÿÿ.'...c...ÿÿÿ.
00417D90	FF	FF	FF	0F	66	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	ÿÿÿ.f...ÿÿÿ.ÿÿÿ.

⇒ Có thể thấy đây là cách truy xuất tìm duyệt khá cực nhọc để tính toán. Hiện em chưa tìm ra giải pháp nào khả quan hơn.

## IV- XÓA TOÀN BỘ CÁC FILE F<K>.DAT VỚI K CHẴN (CÂU D1)

### 1. Trình bày thuật toán:

- Tạo biến index để chạy song song với vòng For (do vòng for không hỗ trợ lấy được index các lần lặp).
- Tạo vòng for duyệt qua tất cả các file theo điều kiện tên file là F\*.dat (khởi đầu bằng F và có đuôi là .dat):
- Nếu index là chẵn thì tiến hành xóa file tương ứng với index.
- Cộng biến index thêm 1 đơn vị để sử dụng cho lần lặp kế tiếp.

## 2. Cài đặt:

```
1 @echo off
2 cls
3 set /A index=0
4
5 FOR /R "G:" %%f IN (F*.dat) DO call :Label1 %%f
6 goto End
7
8 :Label1
9     set file=%1
10    ::echo index=[%index%] : %file%
11    set /A checkEven=%index% %% 2
12    if %checkEven%==0 (
13        echo file: %file% is deleted
14        del %file%
15    )
16    set /A index+=1
17    goto :eof
18
19 :End
20 pause
21
```

\* Ở source code em đã có note giải thích đầy đủ - Đây là bản lược bớt comment để tránh dài ảnh

## 3. Giải thích thêm code:

Lý do dùng label mà không viết code bên trong scope của vòng lặp for:












- Do vòng lặp for không hỗ trợ tạo biến hay set giá trị bên trong nó. Nhưng em cần một biến đếm để lưu trữ vị trí của vòng lặp và để xem xét tạo file thứ i, và một biến để kiểm tra file chẵn lẻ để tạo dung lượng phù hợp yêu cầu đề bài. Vòng lặp được tham khảo tại [2]

## 4. Chạy chương trình:

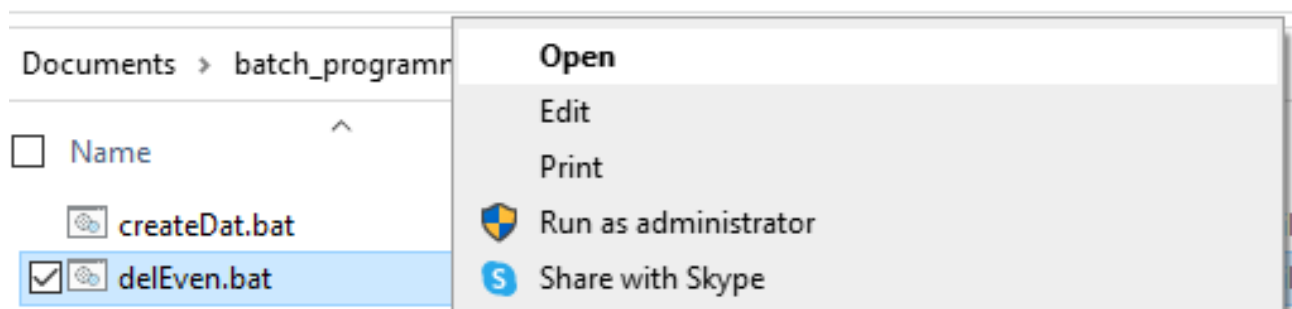
- B0: Kiểm tra ổ đĩa trước khi chạy chương trình.



VOLUME1 (G:)

<input type="checkbox"/> Name	Date modified	Type	Size
 F0.dat	06/12/2021 1:40 AM	DAT File	2 KB
 F1.dat	06/12/2021 1:40 AM	DAT File	1 KB
 F2.dat	06/12/2021 1:40 AM	DAT File	2 KB
 F3.dat	06/12/2021 1:40 AM	DAT File	1 KB
 F4.dat	06/12/2021 1:40 AM	DAT File	2 KB
 F5.dat	06/12/2021 1:40 AM	DAT File	1 KB
 F6.dat	06/12/2021 1:40 AM	DAT File	2 KB
 F7.dat	06/12/2021 1:40 AM	DAT File	1 KB
 F8.dat	06/12/2021 1:40 AM	DAT File	2 KB
 F9.dat	06/12/2021 1:40 AM	DAT File	1 KB
 F10.dat	06/12/2021 1:40 AM	DAT File	2 KB

- B1: Click đúp chuột trái vào file delEven.bat hoặc click chuột phải vào file chọn Open.








- B2: Nhấn phím bất kỳ khi chương trình kết thúc.























```
C:\WINDOWS\system32\cmd.exe
file: G:\F0.dat is deleted
file: G:\F2.dat is deleted
file: G:\F4.dat is deleted
file: G:\F6.dat is deleted
file: G:\F8.dat is deleted
file: G:\F10.dat is deleted
Press any key to continue . . .
```

- B4: Kiểm tra kết quả.

> VOLUME1 (G:)

<input type="checkbox"/> Name	Date modified	Type	Size
 F1.dat	06/12/2021 1:40 AM	DAT File	1 KB
 F3.dat	06/12/2021 1:40 AM	DAT File	1 KB
 F5.dat	06/12/2021 1:40 AM	DAT File	1 KB
 F7.dat	06/12/2021 1:40 AM	DAT File	1 KB
 F9.dat	06/12/2021 1:40 AM	DAT File	1 KB

> VOLUME1 (G:)

<input type="checkbox"/> Name	Date modified	Type	Size
 F1.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F3.dat	06/12/2021 12:16 ...	DAT File	1 KB
<input type="checkbox"/>  F5.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F7.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F9.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F11.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F13.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F15.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F17.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F19.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F21.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F23.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F25.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F27.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F29.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F31.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F33.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F35.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F37.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F39.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F41.dat	06/12/2021 12:16 ...	DAT File	1 KB
 F43.dat	06/12/2021 12:16 ...	DAT File	1 KB

Activ

## V- CỨU F0.DAT (CÂU D2)

Tổng quát:

- B1: Xác định entry chứa thông tin file đã bị xoá.
- B2: Phục hồi byte đầu của entry chính và các entry phụ nếu có.
- B3: Xác định vị trí cluster bắt đầu của file.
- B4: Đến phục hồi các chỉ mục cluster của file ở FAT.
- Ở bước 3: Nếu các file bị xoá là liên tiếp nhau thì phải tiến hành xác định thêm kích thước của file ở entry chính. Sau đó tính toán số lượng cluster file sẽ chiếm rồi khôi phục theo đúng số lượng đó ở bảng FAT

Cụ thể:

- Xác định entry chính chứa thông tin tập tin ở RDET.
- Chỉ số cluster bắt đầu đầu của RDET ở boot sector:

<Số nguyên 4 byte tại offset 2Ch> = 00000002h = 2d

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	EB	58	90	4D	53	44	4F	53	35	2E	30	00	02	02	BE	20
00000010	02	00	00	00	00	F8	00	00	3F	00	FF	00	00	40	C9	37
00000020	00	00	20	00	A1	1F	00	00	00	00	00	00	02	00	00	00
00000030	01	00	06	00	00	00	00	00	00	00	00	00	00	00	00	00
00000040	80	00	29	45	C9	48	6E	4E	4F	20	4E	41	4D	45	20	20
00000050	20	20	46	41	54	33	32	20	20	20	33	C9	8E	D1	BC	F4
00000060	7B	8E	C1	8E	D9	BD	00	7C	88	56	40	88	4E	02	8A	56
00000070	40	B4	41	BB	AA	55	CD	13	72	10	81	FB	55	AA	75	0A

- Đến vị trí cluster của RDET để kiểm tra bằng công thức đã đề cập ở câu A:

$A = SS + (K - FC) * SC = 24576 + (K - 2) * 2$  (với K là chỉ số cluster và A là chỉ số sector)

$K = 2 \Rightarrow A = 24576$

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text	
00C00000	56	4F	4C	55	4D	45	31	20	20	20	20	08	00	00	00	00	VOLUME1 .....	Sector 24,576
00C00010	00	00	00	00	00	00	3A	0C	86	53	00	00	00	00	00	00	.....:tS.....	
00C00020	42	20	00	49	00	6E	00	66	00	6F	00	0F	00	72	72	00	B .I.n.f.o...rr.	
00C00030	6D	00	61	00	74	00	69	00	6F	00	00	00	6E	00	00	00	m.a.t.i.o...n...	
00C00040	01	53	00	79	00	73	00	74	00	65	00	0F	00	72	6D	00	.S.y.s.t.e...rm.	
00C00050	20	00	56	00	6F	00	6C	00	75	00	00	00	6D	00	65	00	.V.o.l.u...m.e.	
00C00060	53	59	53	54	45	4D	7E	31	20	20	20	16	00	C7	39	0C	SYSTEM~1 ..Ç9.	
00C00070	86	53	86	53	00	00	3A	0C	86	53	03	00	00	00	00	00	tStS...:tS.....	
00C00080	E5	30	20	20	20	20	20	20	44	41	54	20	10	AE	0D	0D	ã0 DAT .@...	
00C00090	86	53	86	53	00	00	0E	0D	86	53	06	00	00	08	00	00	tStS...:tS.....	
00C000A0	46	31	20	20	20	20	20	20	44	41	54	20	10	B3	0D	0D	F1 DAT .³...	
00C000B0	86	53	86	53	00	00	0E	0D	86	53	08	00	00	04	00	00	tStS...:tS.....	
00C000C0	E5	32	20	20	20	20	20	20	44	41	54	20	10	B6	0D	0D	ã2 DAT .q...	
00C000D0	86	53	86	53	00	00	0E	0D	86	53	09	00	00	08	00	00	tStS...:tS.....	
00C000E0	46	33	20	20	20	20	20	20	44	41	54	20	10	B9	0D	0D	F3 DAT .³...	
00C000F0	86	53	86	53	00	00	0E	0D	86	53	0B	00	00	04	00	00	tStS...:tS.....	
00C00100	E5	34	20	20	20	20	20	20	44	41	54	20	10	BC	0D	0D	ã4 DAT .4...	
00C00110	86	53	86	53	00	00	0E	0D	86	53	0C	00	00	08	00	00	tStS...:tS.....	
00C00120	46	35	20	20	20	20	20	20	44	41	54	20	10	C3	0D	0D	F5 DAT .Ã...	
00C00130	86	53	86	53	00	00	0E	0D	86	53	0E	00	00	04	00	00	tStS...:tS.....	
00C00140	E5	36	20	20	20	20	20	20	44	41	54	20	10	C6	0D	0D	ã6 DAT .E...	
00C00150	86	53	86	53	00	00	0F	0D	86	53	0F	00	00	08	00	00	tStS...:tS.....	
00C00160	46	37	20	20	20	20	20	20	44	41	54	20	10	01	0E	0D	F7 DAT ....	
00C00170	86	53	86	53	00	00	0F	0D	86	53	11	00	00	04	00	00	tStS...:tS.....	
00C00180	E5	38	20	20	20	20	20	20	44	41	54	20	10	04	0E	0D	ã8 DAT ....	
00C00190	86	53	86	53	00	00	0F	0D	86	53	12	00	00	08	00	00	tStS...:tS.....	
00C001A0	46	39	20	20	20	20	20	20	44	41	54	20	10	07	0E	0D	F9 DAT ....	
00C001B0	86	53	86	53	00	00	0F	0D	86	53	14	00	00	04	00	00	tStS...:tS.....	
00C001C0	E5	31	30	20	20	20	20	20	44	41	54	20	10	0A	0E	0D	ã10 DAT ....	
00C001D0	86	53	86	53	00	00	0F	0D	86	53	15	00	00	08	00	00	tStS...:tS.....	
00C001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	
00C001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....	

- Tìm các entry có chứa byte đầu là E5 (byte này được set về E5 mỗi khi xóa tập tin).
- Tìm entry có chứa tên F0.dat bên cột phải (Decoded text). Do byte đầu đã được chuyển thành E5 nên phải tìm file có tên ã0.dat với ã tương đương E5 trong bảng mã ANSI

00C00060	53 59 53 54 45 4D 7E 31 20 20 20 16 00 C7 39 0C	SYSTEM~1 ..Ç9.
00C00070	86 53 86 53 00 00 3A 0C 86 53 03 00 00 00 00 00	†S†S...†S.....
00C00080	E5 30 20 20 20 20 20 20 44 41 54 20 10 AE 0D 0D	â0 DAT .@..
00C00090	86 53 86 53 00 00 0E 0D 86 53 06 00 00 08 00 00	†S†S...†S.....
00C000A0	46 31 20 20 20 20 20 20 44 41 54 20 10 B3 0D 0D	F1 DAT .³..
00C000B0	86 53 86 53 00 00 0E 0D 86 53 08 00 00 04 00 00	†S†S...†S.....

- Khôi phục lại ký tự F tương đương 46h tại byte đã bị set thành E5.

00C00060	53 59 53 54 45 4D 7E 31 20 20 20 16 00 C7 39 0C	SYSTEM~1 ..Ç9.
00C00070	86 53 86 53 00 00 3A 0C 86 53 03 00 00 00 00 00	†S†S...†S.....
00C00080	46 30 20 20 20 20 20 20 44 41 54 20 10 AE 0D 0D	F0 DAT .@..
00C00090	86 53 86 53 00 00 0E 0D 86 53 06 00 00 08 00 00	†S†S...†S.....
00C000A0	46 31 20 20 20 20 20 20 44 41 54 20 10 B3 0D 0D	F1 DAT .³..
00C000B0	86 53 86 53 00 00 0E 0D 86 53 08 00 00 04 00 00	†S†S...†S.....

- Xác định cluster bắt đầu của file:
- Word cao: <Số nguyên 2 byte tại offset 14Ch> = 0000h

00C00080	46 30 20 20 20 20 20 20 44 41 54 20 10 AE 0D 0D	F0 DAT .@..
00C00090	86 53 86 53 00 00 0E 0D 86 53 06 00 00 08 00 00	†S†S...†S.....

- Word thấp: <Số nguyên 2 byte tại offset 1ACh> = 0006h

00C00080	46 30 20 20 20 20 20 20 44 41 54 20 10 AE 0D 0D	F0 DAT .@..
00C00090	86 53 86 53 00 00 0E 0D 86 53 06 00 00 08 00 00	†S†S...†S.....

⇒ Cluster bắt đầu: 00000006h = 6d

- Di chuyển đến bảng FAT để khôi phục danh sách chỉ số các cluster chứa nội dung file.
- Do vị trí bảng FAT nằm ngay sau boot sector
- Nên số sector của vùng boot sector sẽ là vị trí của bảng FAT do vị trí sector được tính từ 0.
- Đã có từ câu A: SB = 8382d (sector).

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00417C00	F8	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	øÿÿ.ÿÿÿÿÿÿÿ.ÿÿÿ. Sector 8,382
00417C10	FF	FF	FF	FF	FF	FF	FF	FF	00	00	00	00	00	00	00	00	ÿÿÿ.ÿÿÿ.....
00417C20	FF	FF	FF	FF	00	00	00	00	00	00	00	00	FF	FF	FF	FF	ÿÿÿ.....ÿÿÿ.
00417C30	00	00	00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00	.....ÿÿÿ.....
00417C40	00	00	00	00	FF	FF	FF	FF	00	00	00	00	00	00	00	00	....ÿÿÿ.....
00417C50	FF	FF	FF	FF	00	00	00	00	00	00	00	00	00	00	00	00	ÿÿÿ.....
00417C60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00417C70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

- Sau khi bị xóa, hệ điều hành sẽ thiết lập các chỉ số về 0 (để chỉ cluster này đã trống) ở bảng FAT.
- Nên tiến hành tìm đến chỉ mục cluster thứ 6 và các cluster trống liên tiếp kế cạnh.
- Liên tiếp là do ổ đĩa vừa format chưa có sự lộn xộn về sắp xếp các chỉ mục chỉ mục cluster nên danh sách chỉ mục cluster của file bị xóa sẽ là liên tiếp nhau. Bên cạnh đó xóa đi các file xen kẽ nhau giúp ta dễ dàng nhận biết các chỉ mục cluster trống liên tiếp nhau là chỉ thuộc về 1 file.
- Ở bảng FAT32, 1 chỉ mục cluster sẽ chiếm 4 byte.







Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00417C00	F8	FF	FF	0F	FF	FF	FF	FF	FF	FF	FF	0F	FF	FF	FF	0F	øÿÿ.ÿÿÿÿÿÿÿ.ÿÿÿ.
00417C10	FF	FF	FF	0F	FF	FF	FF	0F	00	00	00	00	00	00	00	00	ÿÿÿ.ÿÿÿ. ....
00417C20	FF	FF	FF	0F	00	00	00	00	00	00	00	00	FF	FF	FF	0F	ÿÿÿ.....ÿÿÿ.
00417C30	00	00	00	00	00	00	00	00	FF	FF	FF	0F	00	00	00	00	.....ÿÿÿ.....
00417C40	00	00	00	00	FF	FF	FF	0F	00	00	00	00	00	00	00	00	....ÿÿÿ.....
00417C50	FF	FF	FF	0F	00	00	00	00	00	00	00	00	00	00	00	00	ÿÿÿ.....
00417C60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00417C70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

- Đổi chỉ mục cluster như format danh sách liên kết của hệ điều hành, chỉ mục cluster trước sẽ trỏ đến chỉ mục cluster sau. Chỉ mục cluster cuối cùng sẽ là 0FFFFFFFh.
- Hai chỉ mục cluster trống ở vị trí 6 và 7. Vậy chỉ mục cluster thứ 6 sẽ chứa giá trị 00000007h (để trỏ đến chỉ mục cluster thứ 7), còn chỉ mục cluster thứ 7 sẽ chứa giá trị 0FFFFFFFh để báo kết thúc độ dài.

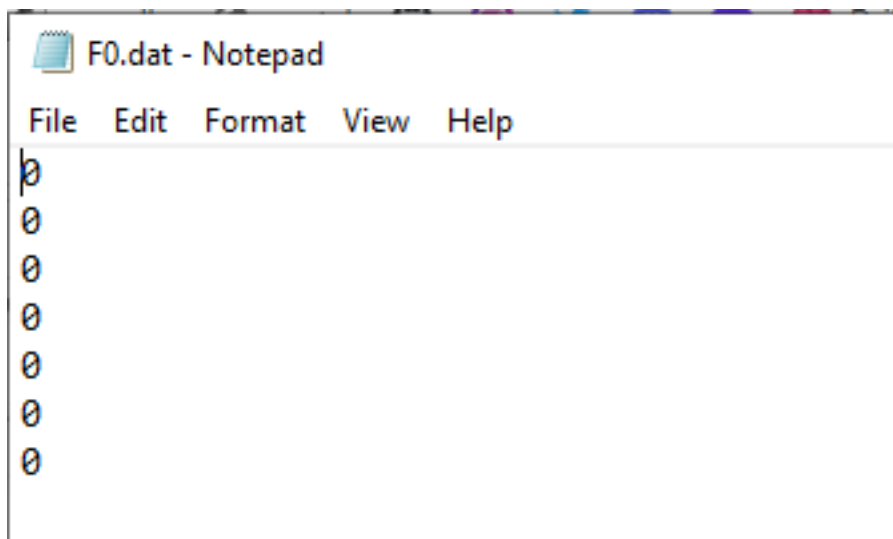
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00417C00	F8	FF	FF	0F	FF	FF	FF	FF	FF	FF	FF	0F	FF	FF	FF	0F	øÿÿ.ÿÿÿÿÿÿÿ.ÿÿÿ.
00417C10	FF	FF	FF	0F	FF	FF	FF	0F	07	00	00	00	FF	FF	FF	0F	ÿÿÿ.ÿÿÿ. ....ÿÿÿ.
00417C20	FF	FF	FF	0F	00	00	00	00	00	00	00	00	FF	FF	FF	0F	ÿÿÿ.....ÿÿÿ.
00417C30	00	00	00	00	00	00	00	00	FF	FF	FF	0F	00	00	00	00	.....ÿÿÿ.....
00417C40	00	00	00	00	FF	FF	FF	0F	00	00	00	00	00	00	00	00	....ÿÿÿ.....
00417C50	FF	FF	FF	0F	00	00	00	00	00	00	00	00	00	00	00	00	ÿÿÿ.....
00417C60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00417C70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

- Tiến hành save lại để lưu những gì vừa thay đổi.
- Kiểm tra ở volume xem đã khôi phục file với dung lượng chính xác chưa.

> VOLUME1 (G:)

<input type="checkbox"/> Name	Date modified	Type	Size
 F0.dat	06/12/2021 1:40 AM	DAT File	2 KB
 F1.dat	06/12/2021 1:40 AM	DAT File	1 KB
 F3.dat	06/12/2021 1:40 AM	DAT File	1 KB
 F5.dat	06/12/2021 1:40 AM	DAT File	1 KB
 F7.dat	06/12/2021 1:40 AM	DAT File	1 KB
 F9.dat	06/12/2021 1:40 AM	DAT File	1 KB

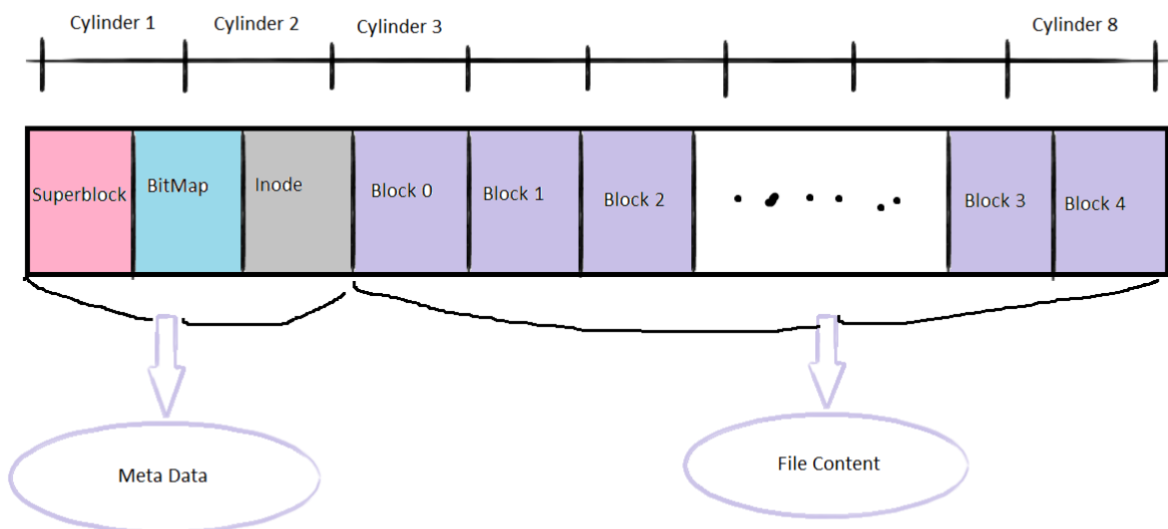
- Mở file kiểm tra nội dung phục hồi có chính xác không.



## C. CÂU 2:

### I – PHẦN A

<u>Tiêu chí</u>	<u>Tỉ lệ hoàn thành (tự đánh giá)</u>
1. Bảo mật	100%
2. Tránh hư hỏng, mất mát	80%
3. Tốc độ truy xuất	90%
4. Tập tin đa dạng có thể tổ chức hệ thống thư mục phân cấp	90%
5. Kích thước	100%
6. Nội dung theo cụm 512byte	100%





## 1. Mức ngữ nghĩa và ý tưởng của thiết kế:

### a) Sắp xếp lại địa chỉ block theo Cylinder:

- Mỗi block là 1 sector
- Với mỗi đĩa ta sẽ chia địa chỉ các sector, block theo thứ tự tăng dần đúng với thứ tự tăng dần của Cylinder

VD: Với Cylinder 1: Các block lần lượt là 1,2,3

Với Cylinder 2: Các block lần lượt là 4,5,6

Có nghĩa là những block liên tiếp đa phần sẽ nằm trên một Cylinder

### b) Thiết kế file:

Volume sẽ được chia làm hai phần: *Meta data* và *File Content*

- Về phần Meta data: Chứa các phần sau đây

+ Superblock: Chứa các thông tin về volume như là:

Tên volume, **Username**, **Password** số lượng block, kích thước của mỗi block, ...

+ Bitmap: Dãy bit lưu trữ thông tin hiện tại của các block trong File Content

VD1: Nếu bit 965 bằng 0: Có nghĩa là Block 965 trong File Content không sử dụng được

VD2: Nếu bit thứ 100 bằng 1: Có nghĩa là Block thứ 100 trong File Content có thể sử dụng

+ Inode Table: Lưu trữ các thông tin cơ bản về thư mục hoặc file chẳng hạn như:

Tên, loại file ( Directory, txt, ...) , encrypted(0: không bảo mật, 1: file bảo mật), key chìa khóa mở file mã hóa,...

Contents: Sẽ lưu các chỉ số block chứa thông tin của file theo thứ tự

Mật khẩu: chuỗi mã hóa của mật khẩu ( ví dụ hashfunction: SHA-256)

## 2. Cu thể từng phần cài đặt để thích hợp với đề bài:

### a) Cụ thể để thiết kế đảm bảo độ bảo mật:

Super Block:

Tên trường	Tính chất
Tên	Lưu tên volume
Num_block	Số lượng block
Size_block	Kích thước mỗi block

User	Tên User truy cập vào volume
Password	Mật khẩu sau khi mã hóa qua công nghệ SHA-256

Inode table:

Tên trường	Tính chất
Tên File	Lưu tên của file
Size	Kích thước của file
Encrypted?	1: Tập tin đã mã hóa
Content	Dãy các block trực tiếp hoặc gián tiếp đã dùng để lưu giữ file này
Password	Lưu mật khẩu sau khi mã hóa qua công nghệ chẳng hạn SHA-256
Last-Open	Thời gian mở tập tin gần nhất có thể là mở để xem hay là sửa
User-Last-Open	Tên của user vừa mới Open

#### \*Giải thích vì sao file có tính bảo mật:

+ Đối với một người mượn máy tính để sử dụng:

Thì tập tin có tính bảo mật vì ở super block đã thiết kế khi vào volume phải đăng nhập mới được vào, nếu có vào được thì khi sửa đổi hoặc mở file đều lưu thông tin Last-Open, User-Last-Open để ta biết được ai đã làm ra điều này.

+ Đối với một người có thể lấy file và đọc từng byte của file:

Thì điều tạm thời ngăn chặn người này lại đó là cấu trúc của file vì đây là cấu trúc tự định nghĩa mật dù có tham khảo nhưng với các số liệu khác nhau chẳng hạn: block = 1024 byte,

Và tên file cách biệt với nội dung tập tin ít nhiều cũng làm khó dễ cho người đọc.

+ Đối với một người chuyên nghiệp có khả năng đọc từng byte của file cũng như đoán được vị trí kiến trúc tập tin:

Một trong những kỹ thuật mã hóa đã được dùng cho dù hacker có thể xác định vị trí của password thì dãy bit password này cũng chỉ là dãy password đã sau khi mã hóa (VD: SHA-256)

Hacker không thể biết rõ password đó là gì để mở được tập tin.

Còn đối với những tập tin đã encrypted=1 thì chắc chắn các tập tin này đã được mã hóa với một chìa khóa duy nhất là key chỉ có chủ nhân mới có chìa khóa này.

Sơ lược về key khi mã hóa file: key chính là password sau khi biến đổi qua SHA-256 lần 1

Còn Password: lưu trong file là mật khẩu sau khi mã hóa hai lần.

*Password lưu trên hệ thống file:  $encrypted\_pass = SHA(SHA(real\_password))$*

*Key giải mã file:  $key = SHA(real\_password)$*

Sở dĩ dùng 2 lần SHA-256 để lưu mật khẩu vì hacker sẽ lấy được dãy mã hóa mật khẩu này thông qua đọc từng bit.

Tuy nhiên, Dãy mã hóa 1 lần  $SHA(real\_password)$  mới là key để mở file này, chứ không phải dãy mã hóa 2 lần

Đây là công nghệ mã hóa mật khẩu mới nhất và khó dò mật khẩu nhất đảm bảo hacker không thể lấy nội dung tập tin dù có biết được từng bit của dữ liệu.

### **b) Thiết kế đảm bảo tốc độ đọc file.**

- + Ta chia đĩa cứng thành group các cylinder ( cylinder là tập hợp các tracks trên các mặt đĩa khác nhau)
- + Với ý tưởng này ta thấy rằng ta cần sắp xếp các file nên nằm trên cùng một cylinder sẽ cho ta cách đọc file nhanh hơn vì đỡ phải di chuyển đầu đọc
- + Khi tạo file, sửa file ta sẽ dùng các block tiếp theo có thể sử dụng được với các block liên tiếp giúp ta hi vọng chúng nằm trên một cylinder.
- + Tìm kiếm các block tiếp theo dựa vào bitmap từ vị trí hiện tại ta dò tìm qua bitmap những block rỗng gần nhất theo sau block đó
- + Cách tổ chức các block theo thứ tự cylinder có nghĩa rằng các block liên tiếp nhau có thể sẽ có khả năng nằm trên một cylinder tối ưu việc tốn kém thời gian di chuyển đầu đọc giảm seeks time khi đọc các file nằm trên các block liên tiếp.

Bitmap là dãy các bit: bit thứ  $i$  thể hiện trạng thái của block thứ  $i$  (  $= 0$  hư,  $= 1$ ) sử dụng được

0	1	2	3	...	....	...	...	...	$2^{24}$
---	---	---	---	-----	------	-----	-----	-----	----------

+ Với  $2^{24}$  bit, có thể biểu diễn được  $2^{24}$  block với mỗi block = sector = 512 byte

Dung lượng lưu trữ trừ phần File data có thể lên đến 8 GB ta có thể giảm số lượng này xuống 4gb bằng cách bỏ những chỉ số dư thừa phía cuối.

### **c) Đảm bảo tránh hư hỏng mất mát của tập tin:**

- + Với thiết kế liên tục theo cylinder như trên các block thiết kế sẽ liên tiếp theo cylinder trên đĩa, đảm bảo tránh hư hỏng vì các thông tin liên quan sẽ nằm liên tiếp trên cylinder dễ dàng khôi phục hơn.

#### d) Thiết kế đảm bảo có thể tổ chức thư mục phân cấp

- + Với phần Inode table ở trong phần meta data ta có thể tổ chức thư mục phân cấp ở phần Content chứa các Block lưu trữ thông tin của file.
- + Ta xem directory cũng như là một file với nội dung Block như sau nội dung của file có chứa Inode tiếp theo sẽ cho ta một thư mục phân cấp.

Name	Inode	Type
HDH.pdf	100	Pdf
User	15	directory

(Hình 1)

Như vậy ta có thể vào Inode directory (tên Win) đi tới Block trong trường Content lưu trữ thông tin (Hình 1) như trên thể hiện đầy đủ thông tin của directory tiếp theo ta có thể đi tới .Win/User

## II- PHẦN B:

Chức năng	Hạn chế	Tỉ lệ hoàn thành (tự đánh giá)
1.Tạo định dạng volume	Các cấu trúc như SuperBlock, Block còn ít thuộc tính	95%
2.Thay đổi cập nhật cài đặt mật khẩu	Mật khẩu lưu chưa được mã hóa	95%
3.Liệt kê tất cả các file		95%
4. Đặt mật khẩu cho tập tin	Chưa mã hóa mật khẩu đồng thời chưa mã hóa dữ liệu cho tập tin	95%
5.Import một file từ bên ngoài vào	Chỉ là đề xuất 27seudocode chưa chạy được chương trình	30%
6.Outport một file ra bên ngoài	Chỉ outport dữ liệu dạng chuỗi char* ra ngoài.	95%
7. Xóa File	Xóa file bằng cách quy ước tên file chưa xóa rõ ràng	95%

### 1. Tạo/ định dạng volume MyFs.Dat

```
struct SuperBlock {
    char username[20];
    char password[20];
    int num_inodes;
    int num_blocks;
    int size_blocks;
};

struct Block
{
    bool state;
    char data[512];
};

struct Inode { // Lưu trữ
    char name[10];
    char type[10];
    char password[10];
    bool state;
    int Block_Position[8];
};

void FileSystem::Create_Volume() {
    FILE* file;
    fopen_s(&file, "MyFS.Dat", "w+");
    if (file != NULL) {
        fwrite(&inf, sizeof(struct SuperBlock), 1, file);
        fwrite(node, sizeof(struct Inode), inf.num_inodes, file);
        fwrite(block, sizeof(struct Block), inf.num_blocks, file);
        fclose(file);
    }
}
```

```
Test cau 1: Tao Volume
Khoi tao password:
Nhap username
tinh
Nhap password
123
```

- Tạo các cấu trúc **Superblock**, **Block**, **Inode** sau đó dùng hàm **Init()** để Initial các giá trị mặc định cho chúng sau đó viết chúng vào file MyFS thông qua hàm **Create\_Volume()**

Hạn chế: Các cấu trúc còn ít thuộc tính để thuận tiện cho việc code hoàn thành 100%

## 2/ Thay đổi, cập nhật, cài đặt mật khẩu volume

```
bool FileSystem::ChangePassword() {
    if (strcmp(inf.username, "") == 0) {
        cout << "Khoi tao password:" << endl;
        cout << "Nhap username" << endl;
        cin.get(inf.username, 20, '\n');
        cin.ignore();
        cout << "Nhap password" << endl;
        cin.get(inf.password, 20, '\n');
        cin.ignore();
        return 1;
    }
    else {
        cout << "Nhap password cu" << endl;
        char pass[20];
        cin.get(pass, 20, '\n');
        cin.ignore();

        if (CheckPass(inf.username, pass) == 1) {
            cout << "Nhap password moi" << endl;
            cin.get(inf.password, 20, '\n');
            cin.ignore();

            return 1;
        }
        else {
            cout << "Password Sai" << endl;
            return 0;
        }
    }
    return 0;
}

void FileSystem::Update_Password_Volume() {
    ChangePassword();
    FILE* file;
    fopen_s(&file, "MyFS.Dat", "w+");
    if (file != NULL) {
        fwrite(&inf, sizeof(struct SuperBlock), 1, file);
        fwrite(node, sizeof(struct Inode), inf.num_inodes, file);
        fclose(file);
    }
}
```

- Viết các hàm cơ bản **ChangePassword()** khi người dùng đã có mật khẩu cần nhập mật khẩu cũ để đổi nếu chưa thì cài đặt mật khẩu. Sau đó đồng bộ phần Meta Data ( hay là SuperBlock và Inode) xuống file thông qua hàm **Update\_PassWord\_Volume()**
- Hạn chế: password chưa được mã hóa để đơn giản code hơn

## 3/ Liệt kê tất cả các file

```
// Câu 3: Liệt kê tất cả thông tin inode có nghĩa là tất cả các file
void FileSystem::List_Of_File() {
    for (int i = 0; i < inf.num_inodes; i++)
        cout << "inode " << i << "\t name" << " " << node[i].name << "\t" << "type:" << node[i].type << endl;
}
```

- Thiết kế giống kiến trúc Unix với bảng Inode liệt kê tất cả các file đồng nghĩa với việc liệt kê tất cả inode đang được dùng

```
inode 94      name:Default    type:Null
inode 95      name:Default    type:Null
inode 96      name:Default    type:Null
inode 97      name:Default    type:Null
inode 98      name:Default    type:Null
inode 99      name:Tinh.txt    type:Null
```

#### 4/ Đặt đổi mật khẩu cho một tập tin:

```
void FileSystem::Update_Password_Inode(int pos) {
    if (strcmp(node[pos].password, "") == 0) {
        cout << "Create new password for file" << endl;
        cin.get(node[pos].password, 20, '\n');
        cin.ignore();

        // Ghi password mới xuống file
        FILE* file;
        fopen_s(&file, "MyFS.Dat", "w+");
        if (file != NULL) {
            fwrite(&inf, sizeof(struct SuperBlock), 1, file);
            fwrite(node, sizeof(struct Inode), inf.num_inodes, file);
        }
        fclose(file);
    }
    else {
        cout << "Input your old password" << endl;
        char pass[20];
        cin.get(pass, 20, '\n');
        cin.ignore();
        if (strcmp(pass, node[pos].password) == 0) {
            cout << "Create new password for file" << endl;
            cin.get(node[pos].password, 20, '\n');
            cin.ignore();

            // Ghi password xuống file
            FILE* file;
            fopen_s(&file, "MyFS.Dat", "w+");
            if (file != NULL) {
                fwrite(&inf, sizeof(struct SuperBlock), 1, file);
                fwrite(node, sizeof(struct Inode), inf.num_inodes, file);
            }
            fclose(file);
        }
        else {
            cout << "Wrong Password" << endl;
        }
    }
}
```

```
Test Cau 4: Voi Inode la 10
Create new password for file
10
Check|||Current Inode 10 pass word is: 10
```

- Tóm tắt ý tưởng: Hàm đơn giản lấy vị trí inode sau đó đổi trường **password** trong Inode đó rồi đồng bộ hóa xuống file MyFS.dat
- Hạn chế chưa mã hóa password
- Giải thích kết quả chạy: Khi lưu mật khẩu xuống file . Sẽ xuất mật khẩu file đó lên console coi có phải là mật khẩu vừa lưu hay không

#### 5/ Import một file từ ngoài vào

```
void FileSystem::Import(char* data, int size, int inode) {
    // chia cum data thành các cum 512 byte
    //Truyen data vao block theo cum 512 byte
    //Tim block_position la block free
    //WriteBlock(block_position, data)
}
```

```
void FileSystem::WriteBlock(int pos, char* data) {
    memcpy(block[pos].data, data, 512);

    // Dong bo hoa xuống file MyFS.Dat
    FILE* file;
    fopen_s(&file, "MyFS.Dat", "w+");
    if (file != NULL) {
        fwrite(&inf, sizeof(struct SuperBlock), 1, file);
        fwrite(node, sizeof(struct Inode), inf.num_inodes, file);
        fwrite(block, sizeof(struct Block), inf.num_blocks, file);
    }
    fclose(file);
}
```

- Phần này em chưa code được chỉ là pseudocode thể hiện ý tưởng
- Bước đầu tạo một file mới trong hệ điều hành sau đó tìm những block còn trống rồi truyền char\* data vào các block còn trống nhờ vào hàm **WriteBlock()** và cập nhật chỉ số các block vừa viết vào ở thuộc tính **Block\_Position** các block mới được bỏ data vào .



```
void FileSystem::ReadBlock(int pos, char*& data) {
    data = new char[512];
    memcpy(data, block[pos].data, 512);
}
```

```
inode 96      name:Default  type:Null
inode 97      name:Default  type:Null
inode 98      name:%efault  type:Null
inode 99      name:Tinh.txt  type:Null
```