THÀNH PHỐ HỒ CHI MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN GIỮA KÌ:
MÔN HỆ ĐIỀU HÀNH

Ninh Duy Huy - 19120533

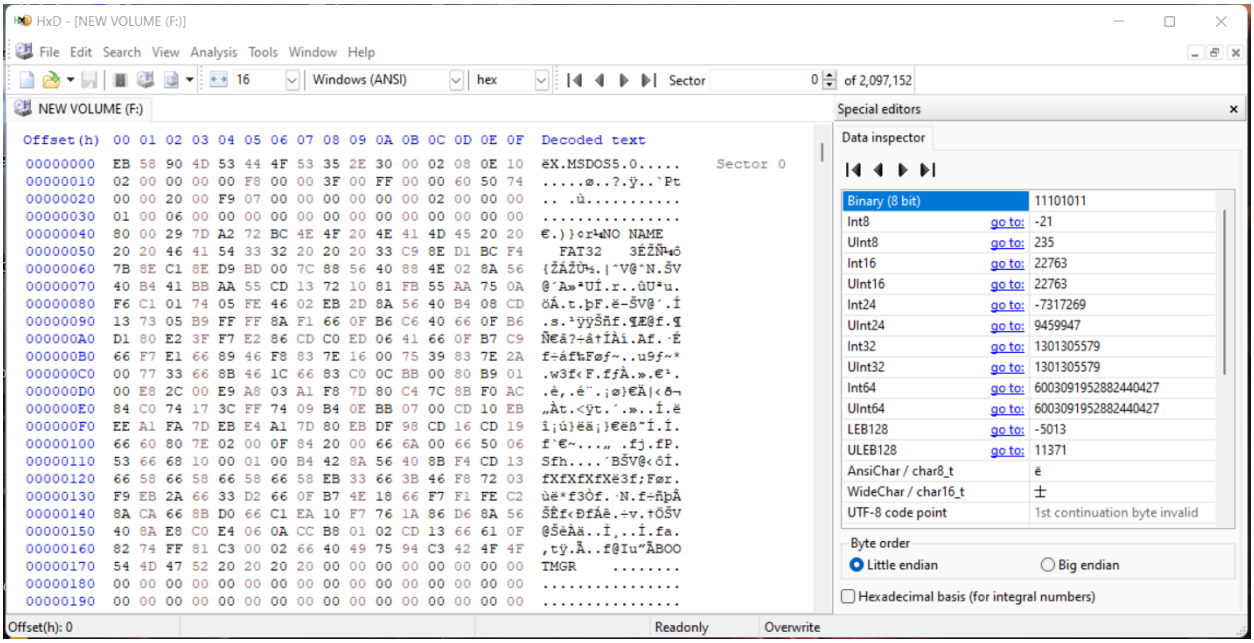
Cao Thanh Khiết - 19120544

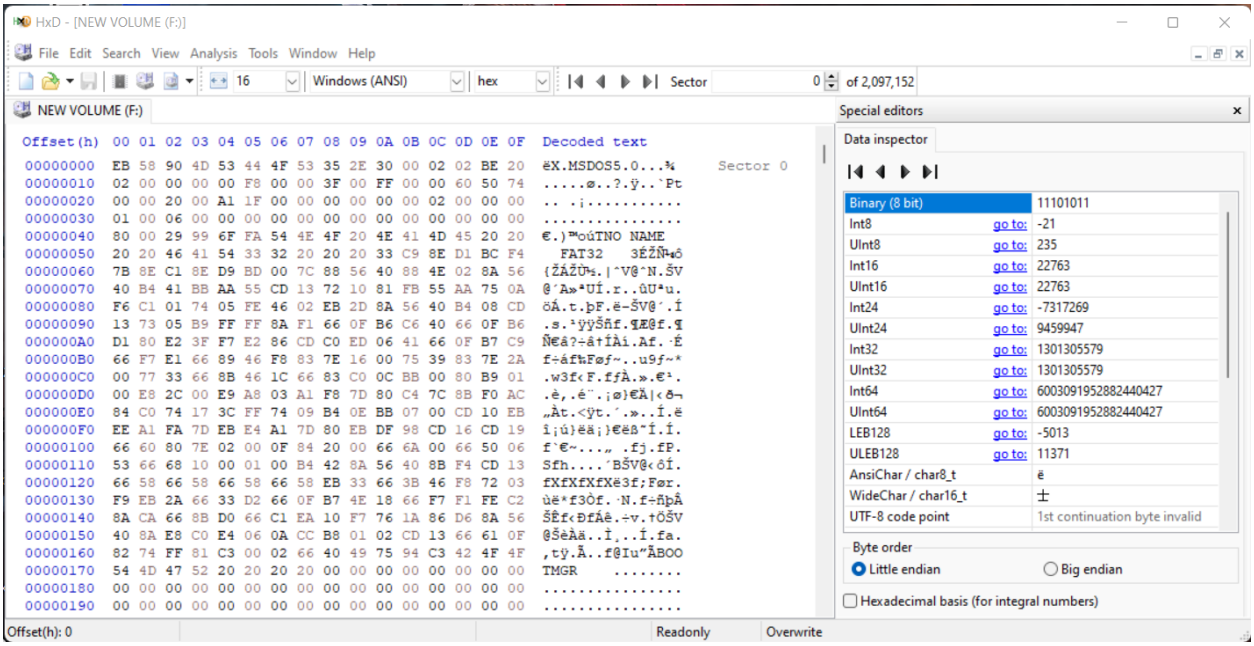
Hà Duy Lãm - 19120559

Bài 1: Phân tích nội dung các sector trên volume FAT32

Câu 1:

a) Xác định các thông số quan trọng sau:





1> Số byte của một sector:

- 2 byte tại offset 0B là: 00, 02

=> Số byte trên mỗi sector của vol là: 0200h = 512 (byte)

2> Kích thước volume theo MegaByte và MebiByte

- 4 byte tại offset 20 là: 00 00 20 00

=> Kích thước volume là: 00200000h = 2097152 MB = 16777216 Mbit

3> Số sector trước FAT:

- 2 byte tại offset 0E là: BE,20

=> Số sector trước FAT là: 20BEh = 8382 (sector)

4> Số sector của một bảng FAT và số bảng FAT :

- 2 byte tại offset 16 là: 00,00

=> Số sector của một bảng FAT là: 0

- 1 byte tại offset 10 là: 02

=> Số bảng FAT là: 02h = 2

5> Số entry (hiện tại) của RDET:

- 2 byte tại offset 11 là: 00,00

=> Số entry (hiện tại) của RDET là 0

6> Số sector của một cluster:

- 1 byte tại offset 0D là: 08

=> Số sector của một cluster là: 08h = 8 (sector)

7> Số cluster của volume:

Ta tìm số sector của volume:

- 4 byte tại offset 20: 00, 00, 20, 00

=> Số sector của volume là: 00200000h = 2097152(sector)

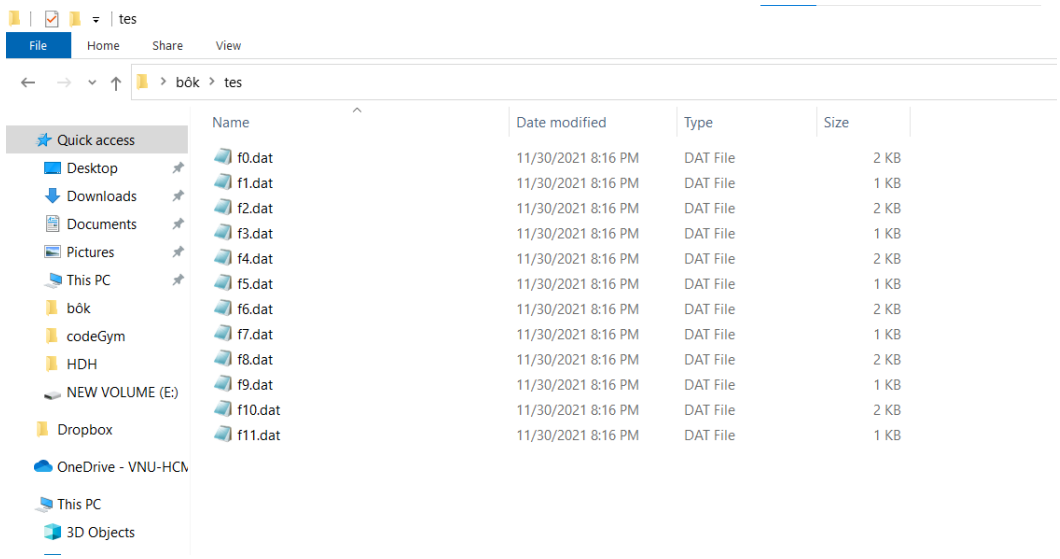
- Vậy số cluster của volume = số sector của volume : số sector trên 1 cluster

=> Số cluster của volume là: 2097152 : 8 = 262144(cluster)

b) Viết chương trình đưa vào (thư mục gốc) N tập tin F0.Dat, F1.Dat,… F<N>.Dat với nội dung của F<K>.Dat là các dòng văn bản mà mỗi dòng đều là giá trị K và chiếm (2 - K%2) cluster

- File “CreateFiles.py”: gồm 2 đối số đầu vào [số N] [thư mục gốc], nếu không nhập đối số thư mục gốc thì sẽ thư mục gốc được tính là thư mục hiện hành.

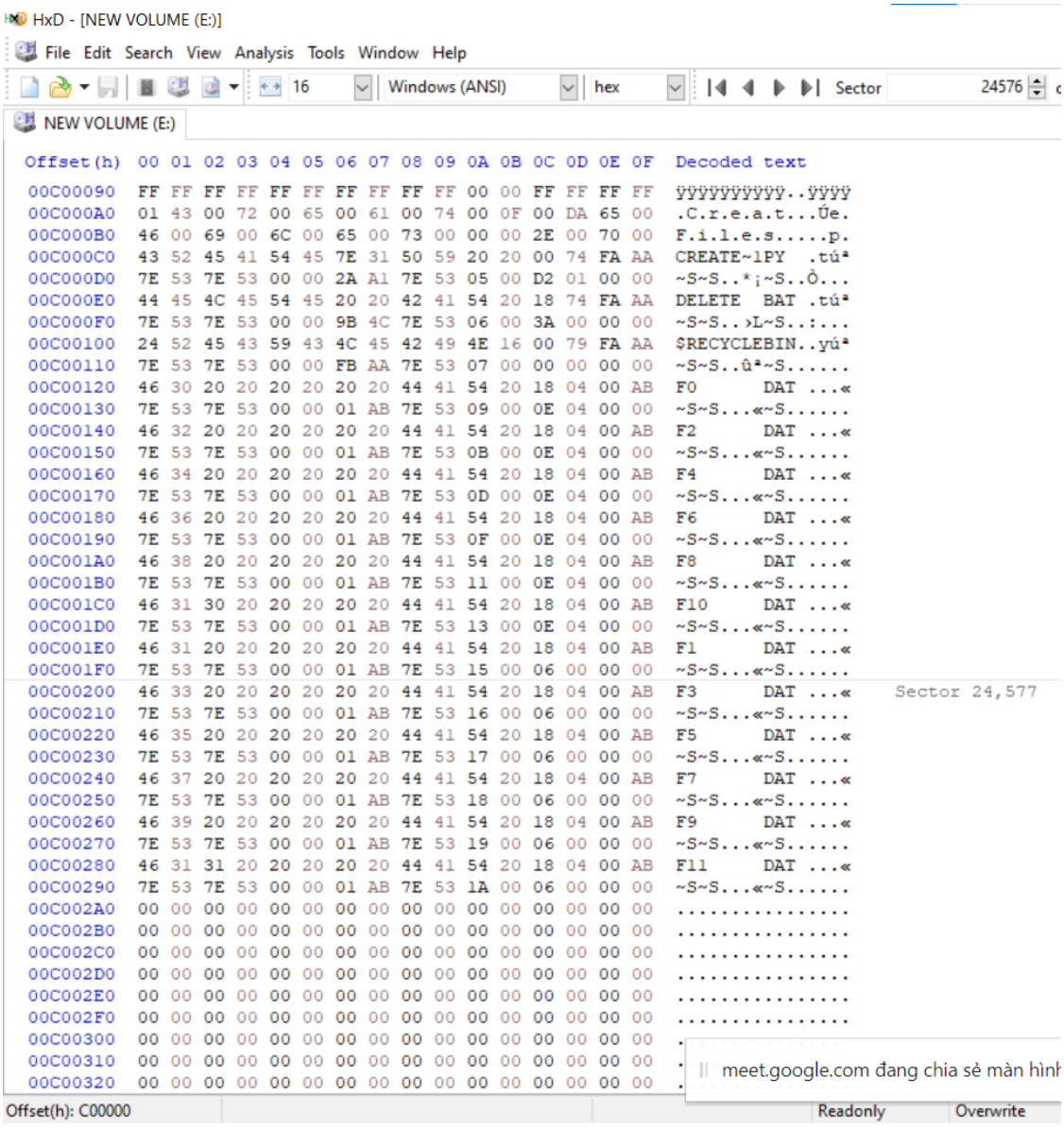
Ví dụ: **CreateFiles.py 11 tes** (11 số file .dat cần tạo, ‘test’ thư mục test trong thư mục hiện tại)



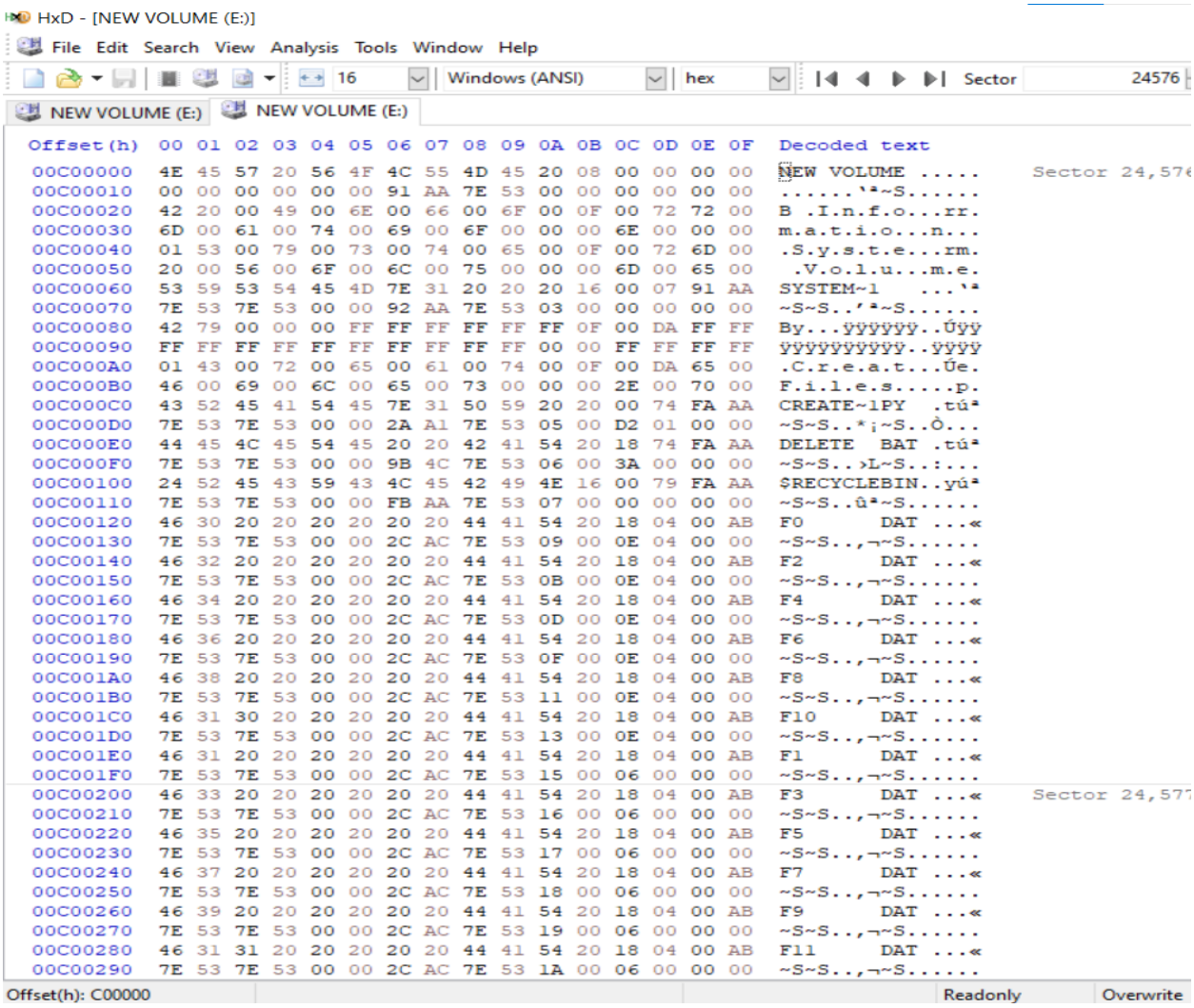
c) Dự đoán (có lý giải) số cluster của RDET khi N là:

Vì tên file ngắn nên chiếm 1 entry chính (32bytes) => mỗi cluster sẽ có 32 entry.

- N=11: số cluster của RDET là 1 vì 1 file (.dat) chiếm 1 entry chính (32 bytes) nên với 11 file thì nằm trong 1 cluster là đủ chính (32 bytes) nên với 11 file thì nằm trong 1 cluster là đủ.


HxD - [NEW VOLUME (E:)]
File Edit Search View Analysis Tools Window Help
16
Windows (ANSI)
hex
Sector
24576
NEW VOLUME (E:)
Offset (h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text
00C00090 FF FF FF FF FF FF FF FF FF FF 00 00 FF FF FF FF ÿÿÿÿÿÿÿÿÿÿ..ÿÿÿÿ
00C000A0 01 43 00 72 00 65 00 61 00 74 00 0F 00 DA 65 00 .C.r.e.a.t...Úe.
00C000B0 46 00 69 00 6C 00 65 00 73 00 00 00 2E 00 70 00 F.i.l.e.s.....p.
00C000C0 43 52 45 41 54 45 7E 31 50 59 20 20 00 74 FA AA CREATE~1PY .túª
00C000D0 7E 53 7E 53 00 00 2A A1 7E 53 05 00 D2 01 00 00 ~S~S..*¡~S..Ò...
00C000E0 44 45 4C 45 54 45 20 20 42 41 54 20 18 74 FA AA DELETE BAT .túª
00C000F0 7E 53 7E 53 00 00 9B 4C 7E 53 06 00 3A 00 00 00 ~S~S..›L~S..:...
00C00100 24 52 45 43 59 43 4C 45 42 49 4E 16 00 79 FA AA $RECYCLEBIN..yúª
00C00110 7E 53 7E 53 00 00 FB AA 7E 53 07 00 00 00 00 00 ~S~S..ûª~S......
00C00120 46 30 20 20 20 20 20 20 44 41 54 20 18 04 00 AB F0 DAT ...«
00C00130 7E 53 7E 53 00 00 01 AB 7E 53 09 00 0E 04 00 00 ~S~S...«~S......
00C00140 46 32 20 20 20 20 20 20 44 41 54 20 18 04 00 AB F2 DAT ...«
00C00150 7E 53 7E 53 00 00 01 AB 7E 53 0B 00 0E 04 00 00 ~S~S...«~S......
00C00160 46 34 20 20 20 20 20 20 44 41 54 20 18 04 00 AB F4 DAT ...«
00C00170 7E 53 7E 53 00 00 01 AB 7E 53 0D 00 0E 04 00 00 ~S~S...«~S......
00C00180 46 36 20 20 20 20 20 20 44 41 54 20 18 04 00 AB F6 DAT ...«
00C00190 7E 53 7E 53 00 00 01 AB 7E 53 0F 00 0E 04 00 00 ~S~S...«~S......
00C001A0 46 38 20 20 20 20 20 20 44 41 54 20 18 04 00 AB F8 DAT ...«
00C001B0 7E 53 7E 53 00 00 01 AB 7E 53 11 00 0E 04 00 00 ~S~S...«~S......
00C001C0 46 31 30 20 20 20 20 20 44 41 54 20 18 04 00 AB F10 DAT ...«
00C001D0 7E 53 7E 53 00 00 01 AB 7E 53 13 00 0E 04 00 00 ~S~S...«~S......
00C001E0 46 31 20 20 20 20 20 20 44 41 54 20 18 04 00 AB F1 DAT ...«
00C001F0 7E 53 7E 53 00 00 01 AB 7E 53 15 00 06 00 00 00 ~S~S...«~S......
00C00200 46 33 20 20 20 20 20 20 44 41 54 20 18 04 00 AB F3 DAT ...« Sector 24,577
00C00210 7E 53 7E 53 00 00 01 AB 7E 53 16 00 06 00 00 00 ~S~S...«~S......
00C00220 46 35 20 20 20 20 20 20 44 41 54 20 18 04 00 AB F5 DAT ...«
00C00230 7E 53 7E 53 00 00 01 AB 7E 53 17 00 06 00 00 00 ~S~S...«~S......
00C00240 46 37 20 20 20 20 20 20 44 41 54 20 18 04 00 AB F7 DAT ...«
00C00250 7E 53 7E 53 00 00 01 AB 7E 53 18 00 06 00 00 00 ~S~S...«~S......
00C00260 46 39 20 20 20 20 20 20 44 41 54 20 18 04 00 AB F9 DAT ...«
00C00270 7E 53 7E 53 00 00 01 AB 7E 53 19 00 06 00 00 00 ~S~S...«~S......
00C00280 46 31 31 20 20 20 20 20 44 41 54 20 18 04 00 AB F11 DAT ...«
00C00290 7E 53 7E 53 00 00 01 AB 7E 53 1A 00 06 00 00 00 ~S~S...«~S......
00C002A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00C002B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00C002C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00C002D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00C002E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00C002F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00C00300 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00C00310 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00C00320 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
meet.google.com đang chia sẻ màn hình
Offset(h): C00000
Readonly
Overwrite

- N=2021: tính số cluster = 2021/32 = 63.15 => số cluster là 64, vị trí các bản RDET không nằm trong một vùng liên tiếp thường là 1 bản RDET chiếm 1 cluster sau đó vị trí file cuối cùng cho biết vị trí của bản RDET tiếp theo.

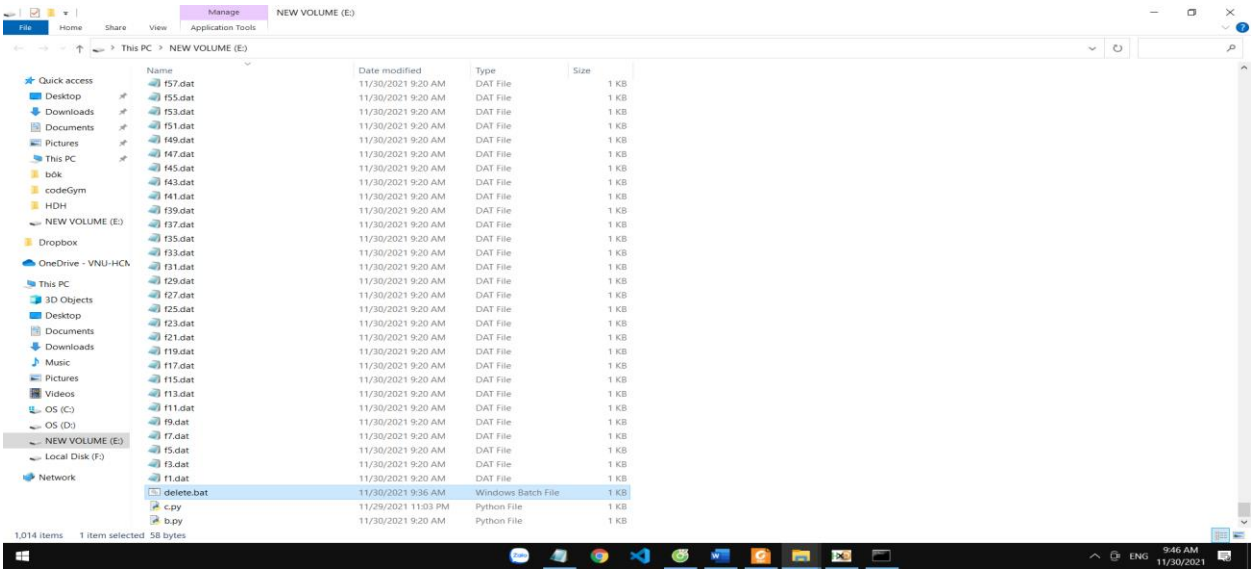


d) Xóa toàn bộ các file F<K>.Dat với K chẵn bằng cách tạo và chạy một file .BAT với vòng lặp FOR bên trong (nếu không thể làm được thì xóa bằng tay qua công cụ File Explorer của HĐH). Sau đó cứu F0.dat bằng cách dùng HexEdit (diễn giải chi tiết các nội dung được điều chỉnh trên sector /các nội dung sector có sử dụng cần đưa vào trong báo cáo.

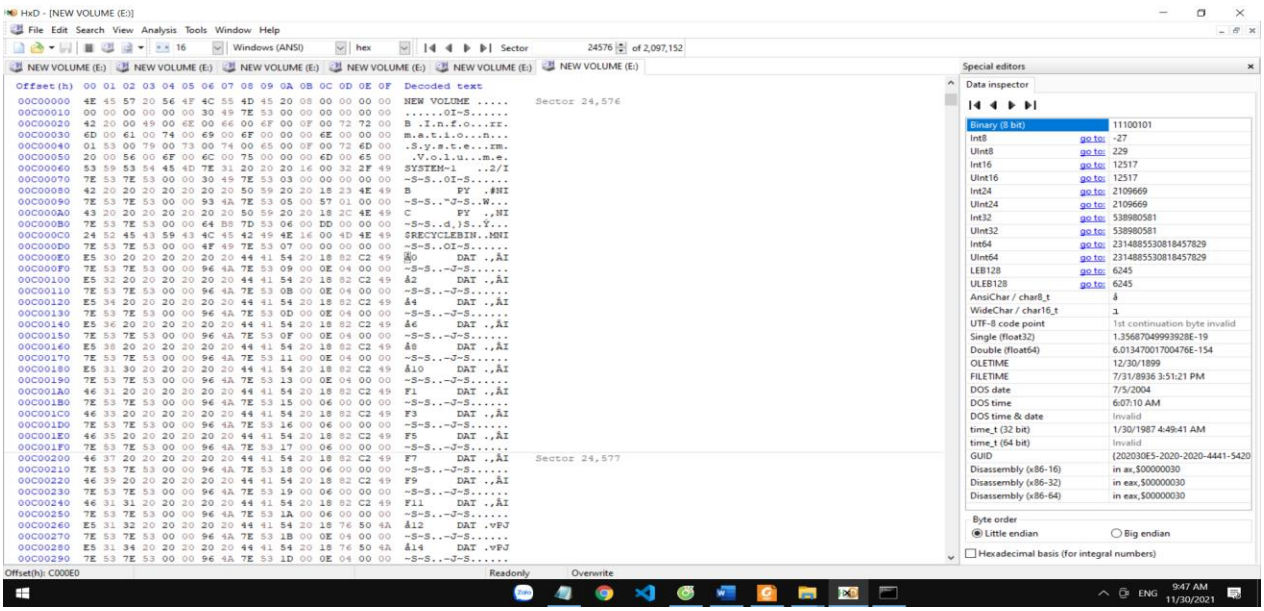
- Xóa các file chẵn bằng file “delete.bat”: file nhận đối số đầu vào là số lượng file và thực hiện xóa các file chẵn.

Vd: delete.bat 2021 thực hiện xóa các file chẵn từ 0 – 2020.

  - Trên file explorer.



- Trên HxD.



Quá trình cứu F0.dat

Dưới đây là Sector 0 của volume. Để đơn giản ta chỉ copy 1 đoạn nhỏ của sector 0.

```
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 2E 02   ëX.MSDOS5.0.....
00000010   02 00 00 00 00 F8 00 00 3F 00 FF 00 80 00 00 00   .....ø..?.ÿ.€...
00000020   00 28 1F 00 E9 1E 00 00 00 00 00 00 02 00 00 00   .(..é...........
00000030   01 00 06 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

➔Byte order: Little Endian

Đầu tiên, ta sẽ tính vị trí các sector sau để thuận lợi cho việc phục hồi file:

Số sector trên 1 cluster = 1 byte tại Offset D = 02 = 2 sector/cluster

Số sector vùng BootSector = 2 byte tại Offset E = **02 2E** = 558

Size của bảng FAT = 4 byte tại Offset 24 = **1E E9** = 7913

Vị trí bảng FAT đầu tiên = số sector vùng BootSector (bởi vì bảng FAT đầu tiên nằm sau vùng BootSector) = 558

Vị trí bảng FAT thứ 2 = số sector vùng BootSector + Size của bảng FAT đầu tiên = 558 + 7913 = 8471

Vị trí của bảng RDET = số sector vùng BootSector + 2 x Size của bảng FAT = 558 + 2 x 7913 = 16384

Tiếp theo, ta tiến hành phục hồi file F0.dat

Đầu tiên ta cần biết bản chất khi xóa file sẽ như nào và từ đó ta sẽ đi tới cách khôi phục:

Khi một file được lưu vào thì nó sẽ tạo 1 Entry chính có kích thước 32 byte tại bảng RDET (ở đây vì tên ngắn nên ta sẽ không xét tới entry phụ), các byte này lưu giữ thông tin của file như tên ngắn, tên mở rộng, cluster bắt đầu, kích thước file, …. và tạo ra những phần tử trên bảng quản lí Cluster (thường sẽ có 2 bảng FAT1 và FAT2) theo cơ chế **Phần tử thứ K sẽ quản lí Cluster K)**. Và khi xóa file thì nội dung file vẫn được giữ nguyên ngoại trừ các con trỏ giữ vị trí cluster sẽ được thay đổi, nghĩa là những thông tin 32 byte trên bảng RDET vẫn giữ nguyên chỉ ngoại trừ byte đầu tiên sẽ được chuyển thành **E5**, và những phần tử trên bảng quản lí Cluster sẽ chuyển tất cả thành 0.

Khôi phục file:

Đầu tiên ta sẽ tới vị trí của bảng RDET (tại sector **16384**) để phục hồi lại tên file F0.dat bằng cách đổi byte đầu tiên là **E5 → 46** (vì 46 là mã Hex của kí tự **F**), vậy là ta đã khôi phục được tên file F0.dat, file đó sẽ được hiển thị lại khi ta mở File Explorer trên máy chúng ta.

Dưới đây là sector tại bảng RDET:

Ta sẽ xét 32 byte (2 dòng) màu cam ở dưới để phục hồi F0.dat

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00800000  54 45 53 54 20 56 4F 4C 55 4D 4D 08 00 00 00 00  TEST VOLUMM.....

00800010  00 00 00 00 00 00 4C 4F 7E 53 00 00 00 00 00 00  ......LO~S......

00800020  E5 20 00 49 00 6E 00 66 00 6F 00 0F 00 72 72 00  å .I.n.f.o...rr.
```

```
00800030  6D 00 61 00 74 00 69 00 6F 00 00 00 6E 00 00 00  m.a.t.i.o...n...
00800040  E5 53 00 79 00 73 00 74 00 65 00 0F 00 72 6D 00  åS.y.s.t.e...rm.
00800050  20 00 56 00 6F 00 6C 00 75 00 00 00 6D 00 65 00   .V.o.l.u...m.e.
00800060  E5 59 53 54 45 4D 7E 31 20 20 20 16 00 C2 4B 4F  åYSTEM~1   ..ÂKO
00800070  7E 53 7E 53 00 00 4C 4F 7E 53 03 00 00 04 00 00  ~S~S..LO~S......
00800080  E5 30 20 20 20 20 20 20 44 41 54 20 18 04 70 4F  å0      DAT ..pO
00800090  7E 53 7E 53 00 00 48 50 7E 53 06 00 76 05 00 00  ~S~S..HP~S..v...
008000A0  E5 31 20 20 20 20 20 20 44 41 54 20 18 04 70 4F  å1      DAT ..pO
008000B0  7E 53 7E 53 00 00 71 4F 7E 53 00 00 00 00 00 00  ~S~S..qO~S......
```

Phân tích:

Tiếp theo ta cần biết vị trí cluster bắt đầu của nội dung file đó bằng cách lấy 2 byte tại Offset **14** (2 byte cao) ghép với 2 byte tại Offset **1A** đó là :

Cluster bắt đầu = **00 00 00 06** = 6 ➔ Cluster bắt đầu của nội dung file là cluster thứ 6.

Và kích thước của file là 4 byte tại Offset **1C** = **05 76** = 1398. Theo như phân tích BootSector ở trên ta có được 1 Cluster sẽ có 2 sector, mà 1 sector = 512 byte, vậy 1 cluster sẽ chỉ chứa được 1024 byte, mà file này có kích thước là 1398 ➔ File này sẽ chiếm 2 cluster.

Tiếp theo, ta sẽ đến vị trí bảng FAT đầu tiên (sector **558**).

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00045C00  F8 FF FF 0F FF FF FF FF FF FF FF 0F 00 00 00 00  øÿÿ.ÿÿÿÿÿÿÿÿ.....
00045C10  FF FF FF 0F FF FF FF 0F 00 00 00 00 00 00 00 00  ÿÿÿ.ÿÿÿ.........
00045C20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00045C30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00045C40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00045C50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Ta có:

- Vì đây là FAT32 nên mỗi phần tử trong bảng quản lí Cluster sẽ chiếm 4 byte, Vậy 1 dòng trên sẽ có 4 phần tử, và được bắt đầu bằng phần tử thứ 0. Theo cách tính như vậy, ta tính được Cluster thứ 6 và 7 sẽ được đánh dấu màu đỏ như trên.
- Cluster kết thúc của FAT32 là byte **0F FF FF FF**, vì đây là Little Endian nên khi lưu vào sẽ là **FF FF FF 0F.**
- Vậy ta sẽ đổi Cluster thứ 6 sẽ giữ giá trị **7** và cluster thứ 7 sẽ mang giá trị **End**

Ta sẽ đổi lại thành:

```
00045C10  FF FF FF 0F FF FF FF 0F 07 00 00 00 FF FF FF FF
```

Ta sẽ làm điều này tương tự trên bảng FAT 2 để đồng bộ với nhau.

➔ Và thế là file đã được phục hồi.

e) Viết chương trình phục hồi các tập tin đã xóa:

Ý tưởng: tương tự việc phục hồi file f0 ở câu d.

Đầu tiên, ta sẽ đọc giá trị của BootSector để xác định xem các thông số của bảng quản lí Cluster và bảng RDET.

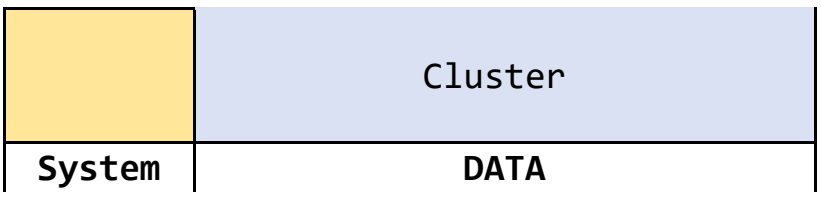
Tiếp theo, ta sẽ ghi đè lên các byte đầu tiên (giá trị E5) trong bảng RDET thành giá trị khác tức là ta đã khôi phục được tên, sau đó nhìn tiếp các thông số trên entry này để xác định được Cluster bắt đầu của nội dung tập tin, và tính toán nội dung tập tin bao nhiêu byte và so sánh với kích thước của cluster từ đó sẽ rút ra được nội dung file này sẽ chiếm bao nhiêu cluster.

Và cuối cùng, ta sẽ đến vị trị bảng FAT và đổi các giá trị phần tử tương ứng với cluster của nó.

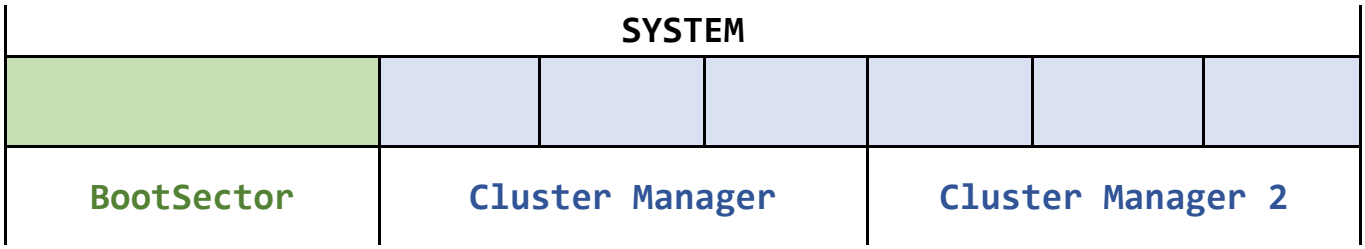
Bài 2: Xây dựng mô hình thiết kế kiến trúc tổ chức file.

A/ Tổ chức hệ thống tập tin trên volume:

Volume sẽ được chia thành 2 vùng chính: vùng hệ thống và vùng dữ liệu.



Vùng hệ thống (System): dùng để chứa các thông tin quan trọng dùng để quản lí volume, nhờ nó mà ta có thể biết được định dạng của volume để thực hiện các thao tác truy xuất,… Vùng System có kích thước nhỏ hơn nhiều so với Data và được truy xuất rất thường xuyên.



Sector đầu tiên của vùng System sẽ là BootSector được thiết kế như sau:

Offset	Size(byte)	Nội dung
0	8	Tên Volume
8	2	Số byte trên Sector, mặc định 512
A	1	Số sector trên cluster
B	1	Số bảng Cluster Manager, mặc định là 2
C	4	Kích thước Volume
10	4	Kích thước 1 bảng Cluster Manager
14	4	Cluster bắt đầu của RDET, mặc định là 2
18	4	Số lượng cluster trống (-1 nếu không có)
1C	2	Sector chứa bản lưu của BootSector
1E	32	Mật khẩu truy cập Volume
…	…	Byte trống

Hiện tại, vùng BootSector chiếm 62 bytes ở sector đầu tiên, những byte trống còn lại ta sẽ để dành cho những phát triển thêm sau này.

→Volume này ta thiết kế có mật khẩu thì mới truy cập được để tăng tính **bảo mật** dữ liệu trong volume, mật khẩu ta quy định là 32 bytes.

Bảng quản lí cluster:

Đầu tiên, ta cần biết về Cluster (xét trên vùng dữ liệu): là một đơn vị truy xuất (thường là lũy thừa của 2) bao gồm 1 dãy N sector liên tiếp, khi đọc ghi sẽ theo từng cụm N sector, bởi vì việc đọc ghi lần lượt từng sector trên 1 dãy sector rất lớn rất không hiệu quả và rất khó quản lí. Và ta cần phải chọn kích thước Cluster sao cho hợp lí để có thể tối ưu tốt nhất khả năng truy xuất của volume cũng như sự phân mảnh của tập tin, nếu

volume của chúng ta chứa nhiều file rất nhỏ mà kích thước của cluster lại rất lớn thì sẽ dẫn đến tình trạng gọi là internal-fragmentation, bởi vì dù file có nhỏ hơn kích thước cluster rất nhiều nhưng khi được đọc ghi thì vẫn theo một lượng là size của Cluster.

Bảng này dùng để quản lí chuỗi các cluster chứa nội dung của tập tin, bảng này có kích thước nhỏ nên sẽ được nạp vào RAM lúc chạy để tăng hiệu suất khi ta cần xác định vị trí các Cluster trống để mà thực hiện các thao tác đọc, ghi, …. Mỗi bảng là một dãy các phần tử, phục vụ cho nhu cầu quản lí các Cluster bởi vì truy xuất vị trí của Cluster là một thao tác rất thường xuyên,vậy nên bảng này rất quan trọng.

Để đảm bảo **tính an toàn** ta sẽ có 2 bảng quản lí cluster nằm kề sau BootSector, bảng này rất quan trọng nên ta sẽ tổ chức theo hình thức danh sách liên kết kết hợp chỉ mục, với hình thức này ta sẽ giải quyết được vấn đề External-Fragmentation. Theo cơ chế mỗi phần tử được dùng để quản lý một cluster trên vùng dữ liệu theo dạng chỉ mục (phần tử K quản lý cluster K). Ta quy định mỗi phần tử K sẽ chiếm 32 bits và giá trị phần tử **K** như sau:

Ví dụ về tổ chức thông tin trên bảng quản lý Cluster:

		3	**8**	**EOF**	**4**	**FREE**	**5**	**EOF**	**FREE**	**EOF**	**…**	**FREE**
0	1	2	3	4	5	6	7	8	9	10	…	n

Khi file được lưu vào bộ nhớ, dữ liệu sẽ được phân mảnh ra thành từng Cluster riêng lẻ và được nối nhau bằng danh sách liên kết, bảng quản lý Cluster sẽ làm nhiệm vụ quản lý đó. Vì Cluster bắt đầu ở vùng Data là 2 nên 2 phần tử 0, 1 sẽ là vị trí trống.

Nếu phần tử K có giá trị **FREE** thì cluster K trên vùng dữ liệu đang ở trạng thái trống

Nếu phần tử K có giẩ trị **BAD** thì cluster K trên vùng dữ liệu đang ở trạng thái hư

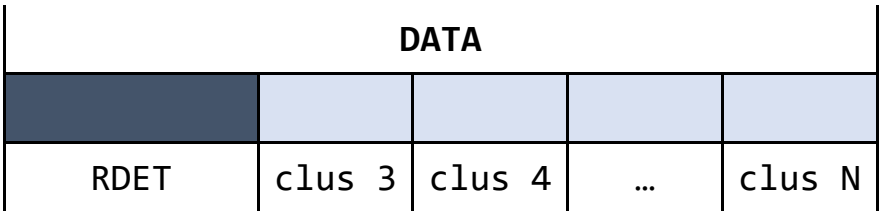
Nếu có giá trị khác FREE và BAD thì cluster K đang chứa nội dung của 1 tập tin và giá trị này cũng là chỉ số cluster kế tiếp chứa nội dung tập tin, nếu giá trị này là **EOF** thì đây chính là cluster cuối cùng của tập tin.

Trạng Thái của Cluster K	**Giá trị**	**Ghi chú**
Trống	0	FREE
Hư	0FFFFFF7	BAD

Cluster cuối của file	0FFFFFFF	EOF
Chứa nội dung, đánh dấu Cluster kế sau	2…0FFFFFFF	USED

Vùng Data:

Là vùng lưu trữ nội dung tập tin và thư mục, nằm ngay sau vùng System. Các sector trên vùng này được nhóm lại thành Cluster để có thể quản lí truy xuất hiệu quả hơn.



Bảng thư mục:

Tổ chức thông tin quản lí tập tin, thư mục thành hệ thống thư mục phân cấp, phần tử đơn vị là Entry. Gồm có 2 loại:

RDET (Root Directory Entry Table): tổ chức thông tin dữ liệu trong thư mục gốc.

SDET (Sub Directory Entry Table): tổ chức thông tin dữ liệu trong thư mục con.

Cluster bắt đầu của vùng Data sẽ là RDET. Bởi vì tên và thuộc tính của tập tin sẽ được lưu riêng vào một vùng để việc truy xuất dữ liệu sẽ nhanh hơn và hiệu quả.

RDET là một dãy các phần tử (entry), mỗi phần tử sẽ chứa thông tin là tên và thuộc tính của một tập tin (hoặc là trống nếu chưa thuộc tập tin nào). Mỗi entry sẽ chiếm 32 byte, các thông tin được tổ chức như sau:

Offset	Số byte	Ý nghĩa
0	8	Tên chính của tập tin
8	3	Tên mở rộng
B	1	Thuộc tính trạng thái
C	8	Mật khẩu truy cập
14	4	Cluster bắt đầu

18	2	Giờ cập nhập tập tin
1A	2	Ngày cập nhập tập tin
1C	4	Kích thước tập tin

Giải thích một số ý nghĩa của entry:

Tên chính: được lưu bằng 8 byte theo bảng mã ASCII, nếu tập tin không đủ 8 byte thì sẽ lưu bằng khoảng trắng.

Tên mở rộng: được lưu bằng 3 byte, nếu không đủ thì lưu bằng khoảng trắng

Thuộc tính trạng thái: được lưu bằng 1 byte. Chứa 5 thuộc tính luận lý, mỗi vị trí của bit tương ứng với một giá trị thuộc tính, quy ước bit 1 là thuộc tính ở trạng thái có và 0 sẽ là ngược lại. Các thuộc tính tương ứng với cái bit trong byte như sau:

0	0	**P**	**R**	**H**	**S**	**D**	**I**

P (Password): nếu có thuộc tính này thì tập tin sẽ yêu cầu password mới có thể mở được.

R (ReadOnly): nếu có thuộc tính này thì tập tin sẽ không thể sửa.

H (Hidden): nếu có thuộc tính này tập tin sẽ bị ẩn trong danh sách liệt kê.

S (System): hệ thống, cho biết tập tin có thuộc hệ điều hành không.

D (Directory): thuộc tính thư mục, nếu có thuộc tính này thì tập tin này không phải là tập tin bình thường mà là tập tin thư mục. Nội dung của thư mục là danh sách các tập tin và thư mục con của nó.

I (Important): nếu có thuộc tính này thì tập tin sẽ được đánh dấu là quan trọng và sẽ có 1 bản sao lưu.

Đối với SDET:

Nội dung của tập tin thư mục này giống y như trên RDET.

2 entry ở đầu bảng sẽ được dùng để mô tả về chính thư mục này và thư mục cha.

Đánh giá:

Thiết kế này sẽ thõa việc bảo mật thông tin bởi vì khi truy cập vào volume lẫn file đều yêu cầu mật khẩu từ người dùng.

Việc an toàn dữ liệu: Việc chọn thiết kế danh sách liên kết kết hợp chỉ mục sẽ giúp hạn chế tối đa việc mất mát hay hư hỏng về con trỏ (vì khả năng lỗi trên một con trỏ sẽ rất cao – khi đó vị trí cluster sẽ bị mất mát rất nghiêm trọng), giúp dữ liệu của ta được đảm bảo an toàn hơn.

Ta đã thiết kế bảng quản lí cluster và bảng nên việc quản lí cluster sẽ rất hiệu quả dẫn đến tốc độ truy xuất sẽ tăng rất đáng kể.

References:

Video bài giảng và slides của giảng viên Thái Hùng Văn.