TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN GIỮA KÌ

HỆ ĐIỀU HÀNH

NGUYỄN QUỐC THẮNG - 19120366

ĐỖ XUÂN THANH – 19120368

NGÔ HỮU ĐANG - 19120467

Giảng viên: Thái Hùng Văn

MỤC LỤC

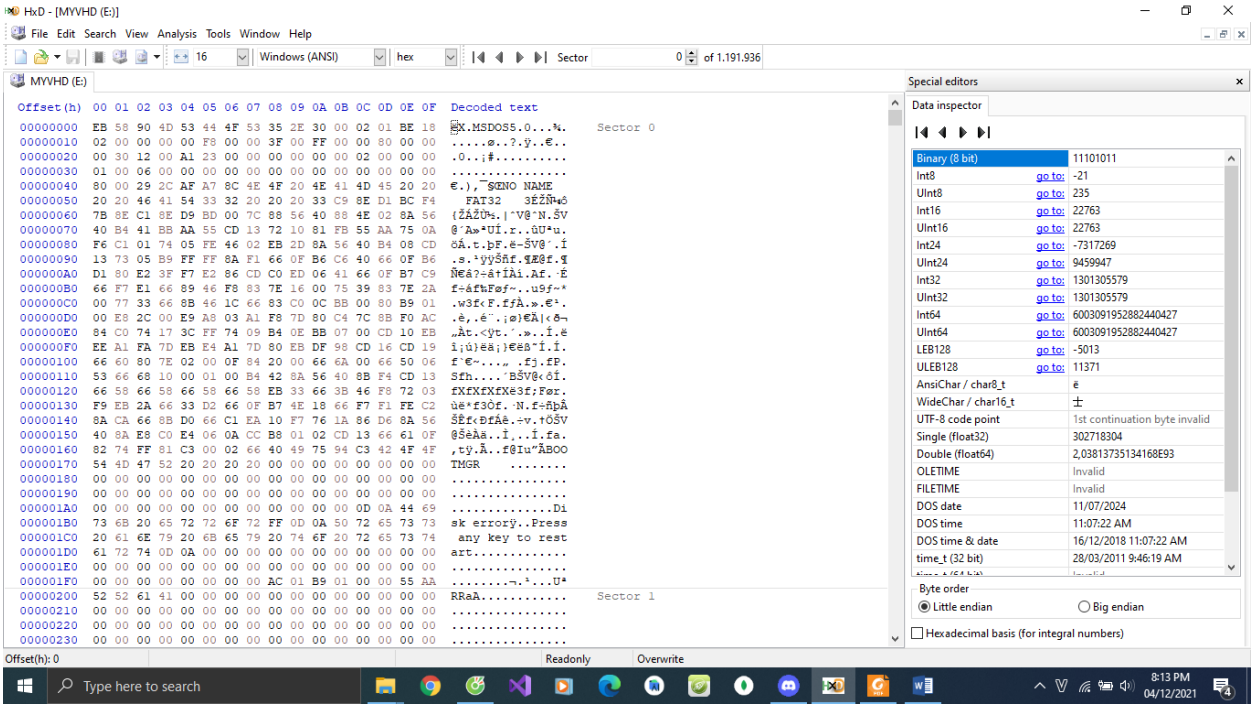
1. Thông tin thành viên

MSSV	Họ và tên	Công việc thực hiện	Tỉ lệ công việc
19120366	Nguyễn Quốc Thắng	Không có	0%
19120368	Đỗ Xuân Thanh	1A, 1B, 1C, 2A, viết báo cáo	50%
19120467	Ngô Hữu Đang	1D, 1E, 2B, chỉnh sửa báo cáo	50%

2. Bài tập 1: Phân tích nội dung sector

2.1. Xác định thông số

2.1.1. Dữ liệu nhận được



2.1.2. Số byte của một sector

Nằm ở offset 0B, 0C của boot sector: 00 02

Giá trị: 0200h = 512 byte

2.1.3. Kích thước volume theo MegaByte và MebiByte

Nằm ở offset 20 21 22 23: 00 23 12 00

Giá trị: 00122300h = 1188608 (sector)

Kích thước volume: 1199608.512 = 614199296 (byte)

614 199 296 byte = 614,199296 (MegaByte) = 585,75 (MebiByte)

2.1.4. Số sector trước FAT

Số sector trước FAT là số sector thuộc vùng Bootsector, nằm ở offset 0E, 0F: BE 18

Giá trị: 18BEh = 6334 (sector)

2.1.5. Số sector của một bảng FAT và số bảng FAT

Số sector của 1 bảng FAT:

Nằm ở offset 24 25 26 27: A1 23 00 00

Giá trị: 000023A1h = 9121 (sector)

Số bảng FAT:

Nằm ở offset 10: 02

Giá trị: 02h =02 (bảng)

2.1.6. Số entry (hiện tại) của RDET

Nằm ở offset 11 12: 00 00

Giá trị: 0000h = 0

2.1.7. Số sector của một cluster

Nằm ở offset 0D: 01

Giá trị: 01d = 1 sector

2.1.8. Số cluster của volume

Số sector của volume nằm ở offset 13 14: 00 00

Giá trị: 0000h = 0 (sector)

Vì trên cluster luôn có 1 sector, nhưng 1 volume không có sector nào, nên trên volume cũng không có cluster, tức là số cluster của volume bằng 0

2.1.9. Công thức tính vị trí của cluster (theo sector)

Vị trí của cluster: Giá trị cluster đầu tiên + Số thứ tự cluster x Số sector trong một cluster

2.2. Viết chương trình đưa vào N tập tin

2.2.1. Tạo 1 file .DAT

```
void Create_DAT_file(string filename, int i)
{
    int cluster = 2 - i % 2;
    int byte = cluster * 1 * 512; // 1 cluster co 1 sector va co do lon 512 byte
    fstream f;
    f.open(filename, ios::out);
    if (cluster == 2)
    {
        // De chiem 2 cluster ta can noi dung kich co lon hon byte
        // 1 so int 4 byte nen luu it nhat byte/4 + 2 so luong so
        for (int i2 = 0; i2 < byte / 4 + 2; i2++)
        {
            f << i << endl;
        }
    }
    else
    {
        f << i << endl;
    }
    f.close();
}
```

Tham số truyền vào: tên file (filename): string, chỉ số (i): int

Các bước thực hiện

- Tính các chỉ số: số byte của file, số cluster của file
- Tạo một con trỏ file f để mở file có tên là filename để ghi
- Kiểm tra cluster
    + Nếu bằng 2: ghi số lượng số giá trị i cho đủ 2 cluster (ta đã tính ra là cần lấp số byte tương ứng giá trị byte, và 1 số nguyên 4 byte, cho nên viết số lượng số là byte/4, mỗi số là giá trị i đầu vào
    + Nếu bằng 1: ghi 1 số là đủ chiếm 1 cluster
- Đóng con trỏ file và đóng file

Kết quả sau khi thực hiện: 1 file .DAT với tên là biến filename, chiếm 2 cluster, nội dung gồm các số giá trị i thoả yêu cầu

2.2.2. Tạo các file .DAT theo tham số đầu vào

```
int main()
{
    fstream f;
    string direct;
    int N;
    do
    {
        cout << "Nhap so file can tao: "; cin >> N; cout << endl;
        cin.ignore();
    } while (N <= 0);
    cout << "Nhao duong dan den dia can tao " << endl;
    cout << "Chang han: C:\\MyComputer\\MyVHD\\" << endl;
    getline(cin, direct);
    for (int i = 0; i < N; i++)
    {
        stringstream builder;
        builder << direct << "F" << i << ".Dat";
        string filename = builder.str();
        Create_DAT_file(filename, i);
    }
    return 0;
}
```

Hàm main của chương trình

Các bước thực hiện:

- Nhập số file cần tạo (một số nguyên lớn hơn 0), nếu không thoả sẽ yêu cầu nhập lại

- Chọn đường dẫn lưu file (ở đây là đường dẫn đến ổ đĩa FAT32 đã format, nhập như mẫu)

- Tạo một vòng lặp N lần

+ Xác định đường dẫn file: ghép đường dẫn đã nhập với chữ F, giá trị i và file mở rộng .DAT (Ví dụ: Nếu nhập đường dẫn D:\\MyVHD\\ thì tên file sẽ là D:\\MyVHD\\F0.Dat (các vòng lặp tiếp theo sẽ là F1.Dat, F2.Dat, …)

+ Thực hiện hàm Create_DAT_file với tham số filename và i đã có

Kết quả trả về: N file từ F0.Dat đến F(N-1).Dat thoả mãn điều kiện bài toán

2.3. Dự đoán số cluster của RDET

Cluster bắt đầu của RDET ñược lưu bằng số nguyên 4 byte ở offset 2Ch trên Boot sector: Các byte 2C 2D 2E 2F: 02 00 00 00 nên số cluster bắt đầu là 00000002h = 2

2.3.1. N=11

N = 11, tức là thêm 11 file từ F0 đến F10

Số cluster lúc này = 2 (bắt đầu) + 2 (F0) + 1 (F1) + 2 (F2) + 1 (F3) + … + 1 (F9) + 2 (F10) = 2 + 2.(6) + 1.(5) = 19

2.3.2. N=2021

N = 2021, tức là thêm 2021 file từ F0 đến F2020

Số cluster lúc này = 2 (bắt đầu) + 2 (F0) + 1 (F1) + 2 (F2) + 1 (F3) + … + 1 (F2019) + 2 (F2020) = 2 + 2.(1011) + 1.(1010) = 3034

2.4. Xoá toàn bộ các file F<K>.DAT (K chẵn)

2.4.1. Xoá toàn bộ các file F<K>.DAT (K chẵn)

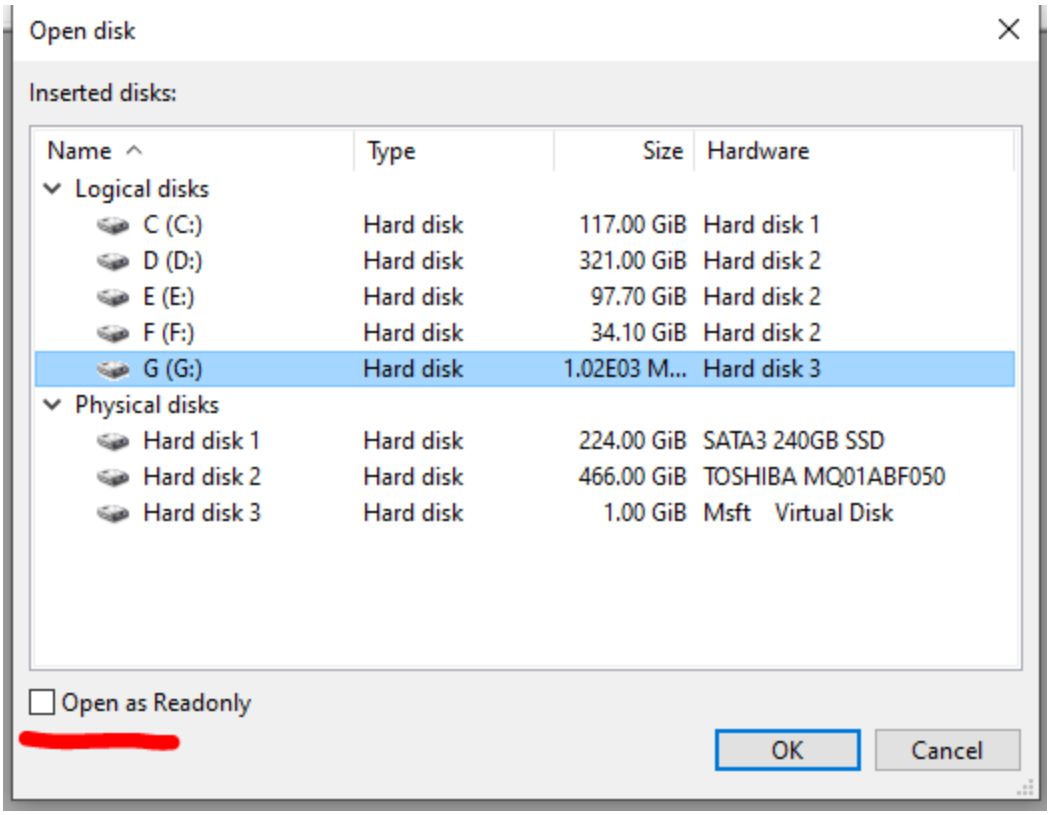
- Tạo 1 file .bat, trong file ta thực hiện đếm các file hiện có

- Dùng vòng for chạy từ 0 tới i <số lượng file> step = 2 (chương trình sẽ xóa hết tất cả file F<k> chẵn vì số lượng F<k> chẵn nhỏ hơn số lượng file, ta không cần nhập N vào file .bat. Khi chạy chỉ cần nhấn đúp chuột vào file .bat là chương trình sẽ chạy, ta không cần mở cmd rồi truyền tham số vào)

```
@echo off
echo Truoc khi xoa
dir
pause
@REM dem so luong file
set cnt=0
for %%A in (*) do set /a cnt+=1
@REM xoa cac file chan
for /l %%n in (0,2,%cnt%) do (
  del f%%n.dat
)
echo Sau khi xoa
dir
pause
```

2.4.2. Cứu F0.DAT (HexEdit)

- Mở ổ đĩa ảo lên bằng HexEdit và bỏ chọn chế độ readonly



- Từ bootsector của ổ đĩa, ta lần lượt lấy các thông số bootsector (S_B), FAT (S_F), số lượng FAT (N_F), số sector trên 1 cluster (S_C), cluster bắt đầu RDET (clus_RDET)

```
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 08 1E 10  ëX.MSDOS5.0.....
00000010  02 00 00 00 00 F8 00 00 3F 00 FF 00 80 00 00 00  .....ø..?.ÿ.€...
00000020  00 E0 1F 00 F1 07 00 00 00 00 00 00 02 00 00 00  .à..ñ...........
```

Thông Số	Vị trí	Byte-BE	Byte-LE	Kích thước(sector)
S_B	0xE-0xF	0x1E10	0x101E	4126
S_F	0x24-0x227	0xF1070000	0x7F1	2033
N_F	0x10	0x2	0x2	2
S_C	0xD	0x8	0x8	8
clus_RDET	0x2C-0x2F	0x2000000	0x2	2

- Vị trí cluster thứ 2 là

Clus_2 = (S_B+S_F*N_F+(2-2)*S_C) = 8192 (sec)

=> Vị trị bảng RDET nằm ở sector thứ 8192

- Di chuyển tới bản RDET và quan sát các entry của file

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

00400100  E5 30 20 20 20 20 20 20 44 41 54 20 18 77 F9 BC  å0      DAT .wù¼
00400110  76 53 79 53 00 00 63 4D 77 53 08 00 50 19 00 00  vSyS..cMwS..P...
00400120  E5 6D 00 65 00 6E 00 74 00 2E 00 0F 00 9F 74 00  åm.e.n.t.....Ÿt.
00400130  78 00 74 00 00 00 FF FF FF FF 00 00 FF FF FF FF  x.t...ÿÿÿÿ..ÿÿÿÿ
00400140  E5 4E 00 65 00 77 00 20 00 54 00 0F 00 9F 65 00  åN.e.w. .T...Ÿe.
00400150  78 00 74 00 20 00 44 00 6F 00 00 00 63 00 75 00  x.t. .D.o...c.u.
```

- Ở trên là 1 entry của RDET

+ 2 dòng đầu là entry chính (BYTE thứ 11 khác 0xF)

+ 2 dòng tiếp theo là entry phụ (BYTE thứ 11 là 0xF)

+ 2 dòng cuối là entry phụ (BYTE thứ 11 là 0xF)

- Quan sát các tên tập tin của entry này

+ 8 byte đầu (tên chính): ảo

+ 3 byte tiếp theo (phần mở rộng): DAT

+ Cluster chứa nội dung file (byte thứ 20,21,26,27) 0x00008000: 8 (đảo byte lại vì volume lưu theo little-endian)

- Ta di chuyển đến cluster thứ 8 để kiểm tra nội dung

Clus_8 == (S_B+S_F*N_F+(8-2)*S_C) = 8240 (sec)

=> Vị trị bảng RDET nằm ở sector thứ 8240

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

00406000  30 0D 0A 30 0D 0A 30 0D 0A 30 0D 0A 30 0D 0A 30  0..0..0..0..0..0
00406010  0D 0A 30 0D 0A 30 0D 0A 30 0D 0A 30 0D 0A 30 0D  ..0..0..0..0..0.
00406020  0A 30 0D 0A 30 0D 0A 30 0D 0A 30 0D 0A 30 0D 0A  .0..0..0..0..0..
00406030  30 0D 0A 30 0D 0A 30 0D 0A 30 0D 0A 30 0D 0A 30  0..0..0..0..0..0
00406040  0D 0A 30 0D 0A 30 0D 0A 30 0D 0A 30 0D 0A 30 0D  ..0..0..0..0..0.
```

- Quan sát thấy nội dung là của f0.dat, ta quay lại bảng RDET, đến entry của file f0.dat (entry lưu thông tin cluster 8) và sửa byte đầu thành 0x46 (kí tự F ở mã ASCII)

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

00400100  46 30 20 20 20 20 20 20 44 41 54 20 18 77 F9 BC  F0      DAT .wù¼
00400110  76 53 79 53 00 00 63 4D 77 53 08 00 50 19 00 00  vSyS..cMwS..P...
```

- Xem kích thước file (từ byte thứ 28-byte thứ 31): 0x50190000 => 0x1950 (đảo byte lại vì volume lưu theo little endian) => Kích thước file= 6480B => chiếm 2 cluster (8*512<6480<=2*8*512)

- Di chuyển đến bản FATvà cập nhật lại cluster thứ 8 và 9 (2 cluster của file f0.dat)

- Vị trí bảng FAT thứ nhất là nằm sau bootsector => ở sector 4126

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

00203C00  F8 FF FF 0F FF FF FF 7F FF FF FF 0F FF FF FF 0F  øÿÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.
00203C10  FF FF FF 0F FF FF FF 0F FF FF FF 0F FF FF FF 0F  ÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.
00203C20  00 00 00 00 00 00 00 00 FF FF FF 0F FF FF FF 0F  ........ÿÿÿ.ÿÿÿ.
```

- 0x0FFFFFF8 là byte bắt đầu của bảng FAT

- Volume theo định dạng FAT32 nên 4 byte sẽ đánh dấu là 1 cluster

- Quan sát thấy vị trí 0x203C20-0x203C27 là vị trí lưu cluster 8 9 của file f0.dat

- Ta cập nhật bằng cách lưu vị trí cluster tiếp theo (9) vào vị trí hiện tại (8) theo Little endian, cluster thứ 9 sẽ lưu giá trị EOF(0x0FFFFFFF)

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

00203C00  F8 FF FF 0F FF FF FF 7F FF FF FF 0F FF FF FF 0F  øÿÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.
00203C10  FF FF FF 0F FF FF FF 0F FF FF FF 0F FF FF FF 0F  ÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.
00203C20  09 00 00 00 FF FF FF 0F FF FF FF 0F FF FF FF 0F  ....ÿÿÿ.ÿÿÿ.ÿÿÿ.
```

- Ấn save và ta đã khôi phục được file f0.dat

2.5. Khôi phục các tập tin bị xoá

Khi ta đã xoá file, thì nội dung file vẫn còn trên volume

Các bước khôi phục như sau:

- Thay đổi offset đầu tiên của entry lưu thông tin file ở bảng RDET hành 1 byte khác 0xE5(entry có 1 entry chính và 2 entry phụ(cách nhận biết entry phụ là ở byte thứ B sẽ có giá trị là 0F))

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

00400100  E5 30 20 20 20 20 20 20 44 41 54 20 18 77 F9 BC  å0      DAT .wù¼
00400110  76 53 79 53 00 00 63 4D 77 53 08 00 50 19 00 00  vSyS..cMwS..P...
00400120  E5 6D 00 65 00 6E 00 74 00 2E 00 0F 00 9F 74 00  åm.e.n.t.....Ÿt.
00400130  78 00 74 00 00 00 FF FF FF FF 00 00 FF FF FF FF  x.t...ÿÿÿÿ..ÿÿÿÿ
00400140  E5 4E 00 65 00 77 00 20 00 54 00 0F 00 9F 65 00  åN.e.w. .T...Ÿe.
00400150  78 00 74 00 20 00 44 00 6F 00 00 00 63 00 75 00  x.t. .D.o...c.u.
```

Entry sau khi thay đổi

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

00400100  46 30 20 20 20 20 20 20 44 41 54 20 18 77 F9 BC  F0      DAT .wù¼
00400110  76 53 79 53 00 00 63 4D 77 53 08 00 50 19 00 00  vSyS..cMwS..P...
00400120  E5 6D 00 65 00 6E 00 74 00 2E 00 0F 00 9F 74 00  åm.e.n.t.....Ÿt.
00400130  78 00 74 00 00 00 FF FF FF FF 00 00 FF FF FF FF  x.t...ÿÿÿÿ..ÿÿÿÿ
00400140  E5 4E 00 65 00 77 00 20 00 54 00 0F 00 9F 65 00  åN.e.w. .T...Ÿe.
00400150  78 00 74 00 20 00 44 00 6F 00 00 00 63 00 75 00  x.t. .D.o...c.u.
```

- Từ entry vừa rồi ra sẽ kiếm được cluster chứa nội dung file (như ở trên thì cluster chúa nội dung file là 0x0008 (theo little endian) di chuyển tới bảng quản lý cluster FAT và gán giá trị USED cho cluster chứa nội dung mà ta cần khôi phục.

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

00203C00  F8 FF FF 0F FF FF FF 7F FF FF FF 0F FF FF FF 0F  øÿÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.
00203C10  FF FF FF 0F FF FF FF 0F FF FF FF 0F FF FF FF 0F  ÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.
00203C20  00 00 00 00 00 00 00 00 FF FF FF 0F FF FF FF 0F  ........ÿÿÿ.ÿÿÿ.
00203C30  00 00 00 00 FF FF FF 0F 00 00 00 00 00 00 00 00  ....ÿÿÿ.........
00203C40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Vi dụ cluster chứa nội dung file cần khôi phục là 8, kích thước file chiếm 2 cluster 8, 9 (ở trên ta thấy offset từ 0x203C20 tới 0x203C27 có giá trị 0) thì ta ghi vào offset từ 0x203C20 tới 0x203C23 dãy byte 0x00000009 theo little endian nhằm mục đích cluser 09 là của file, ta ghi vào offset từ 0x203C24 tới 0x203C27 dãy byte 0x0FFFFFFF (EOF) - dấu hiệu kết thúc file. Sau khi ghi xong sẽ có giá trị sau

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

00203C00  F8 FF FF 0F FF FF FF 7F FF FF FF 0F FF FF FF 0F  øÿÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.
00203C10  FF FF FF 0F FF FF FF 0F FF FF FF 0F FF FF FF 0F  ÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.
00203C20  09 00 00 00 FF FF FF 0F FF FF FF 0F FF FF FF 0F  ........ÿÿÿ.ÿÿÿ.
00203C30  00 00 00 00 FF FF FF 0F 00 00 00 00 00 00 00 00  ....ÿÿÿ.........
00203C40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Cluster chứa nội dung file cần khôi phục là 12, kích thước file chiếm 1 cluster 12 thì ta ghi vào offset offset từ 0x203C30 tới 0x203C33 dãy byte 0x0FFFFFFF (EOF) - dấu hiệu kết thúc file

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

00203C00  F8 FF FF 0F FF FF FF 7F FF FF FF 0F FF FF FF 0F  øÿÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.
00203C10  FF FF FF 0F FF FF FF 0F FF FF FF 0F FF FF FF 0F  ÿÿÿ.ÿÿÿ.ÿÿÿ.ÿÿÿ.
```

```
00203C20  09 00 00 00 FF FF FF 0F FF FF FF 0F FF FF FF 0F  ........ÿÿÿ.ÿÿÿ.
00203C30  FF FF FF 0F FF FF FF 0F 00 00 00 00 00 00 00 00  ....ÿÿÿ.........
00203C40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

- Khi thực hiện code thì sẽ có vấn đề phát sinh là 1 số hệ điều hành sẽ không cho chỉnh sửa volume ở bảng quản lí cluster FAT và bảng thư mục gốc RDET khi nó không bị lỗi. Vì thế ta gần làm hỏng volume trước rồi tiến hành chỉnh sửa các offset bên trong volume

- Làm hỏng volume bằng cách:

+ Đọc bootsector vào trong 1 buffer riêng (không được sử dụng buffer này cho tới khi muốn khôi phục lại volume)

+ Ghi sector rác vào bootsector (việc ghi vào bootsector vẫn có thể thực hiện được)

+ Khi volume bị hỏng thì nó như 1 file lỗi và khi đó ta có thể sửa bảng quản lí cluster FAT với bảng thư mục gốc RDET

+ Sau khi chỉnh sửa các offset như đã nói ở trên, ta tiến hành khôi phục bootsector bằng việc ghi đè buffer chứa nội dung bootsector ban đầu ta đã backup

3. Bài tập 2: Tổ chức hệ thống thông tin

3.1. Xây dựng mô hình

3.1.1. Các tiêu chí

- Việc bảo mật thông tin (tránh bị lộ nội dung 1 số tập tin quan trọng) được xem là thiết yếu nhất

-Việc an toàn dữ liệu (tránh hư hỏng /mất mát nội dung các tập tin quan trọng) cũng rất cần thiết

-Tốc độ truy xuất tập tin cũng cần đạt mức không thấp hơn quá nhiều so với tốc độ truy xuất các tập tin bên ngoài của HĐH (và cũng ưu tiên cho việc đọc ra hơn là ghi vào)

-Số tập tin cần tổ chức trong MyFS.Dat có thể đủ nhiều /đa dạng để phải tổ chức hệ thống thư mục phân cấp

-Kích thước tập tin trong MyFS không quá lớn (<4GB) và các tập tin kích thước lớn (>100MB) sẽ là không quan trọng và không đòi hỏi phải đáp ứng tốt 3 tiêu chí đầu

-Nội dung các byte bên trong file MyFS.Dat cần được truy xuất theo từng cụm 512 byte giống như sector trên các volume của HĐH hoặc các đĩa của máy tính. (tức cần thiết kế & xây dựng các hàm ReadBlock / WriteBlock giống các hàm ReadSector / WriteSector của HĐH & máy tính)

3.1.2. Xây dựng mô hình

Giả sử hệ thống tập tin được chứa trong 1 volume có kích thước 10000 sector, khi đó xây dựng mô hình thiết kế tổ chức như sau:

Phần 1: SYSTEM AREA (chiếm 200 sector)

Bootsector: Sector đầu tiên của volume (có thể bao gồm các sector dự trữ và 1 bảng backup bootsector)

Bảng Cluster: Quản lí các cluster thông qua danh sách liên kết kết hợp chỉ mục, có lưu 1 bản sao

Phần 2: DATA AREA (chiếm 9800 sector)

RDET: Chiếm 1cluster và sẽ mở rộng sau (như SDET)

SDET: Chiếm 1 cluster và sẽ mở rộng sau

Cluster: 2450 cluster, mỗi cluster có 4 sector, trong đó, có 1 cluster dùng để lưu trữ mật khẩu của hệ thống dưới dạng hashing

Giải thích chức năng:

- Hệ thống có thiết lập mật khẩu và mã hoá mật khẩu bằng cách sử dụng hàm băm (hashing) nên nếu có bị xâm nhập thì cũng không thể đăng nhập để lấy cắp dữ liệu (tiêu chí 1)

- Việc quản lí cluster bằng danh sách liên kết kết hợp chỉ mục sẽ có hạn chế là có thể bị sai bảng quản lí cluster, vì vậy trong vùng SYSTEM AREA có lưu thêm 1 bản sao của bảng cluster để dễ dàng lấy lại bảng trong trường hợp 1 trong 2 bảng bị hỏng dữ liệu quản lí (tiêu chí 2)

- Quản lí bằng danh sách kết hợp chỉ mục sẽ dễ tìm kiếm nội dung các file so với các cách quản lí khác như danh sách thuần hay là cluster liên tiếp (chỉ tra cứu bảng quản lí là lấy được file), hơn nữa vì các cluster không liên tiếp nên việc đọc cũng dễ dàng hơn so với ghi (tiêu chí 3)

3.2. Viết chương trình thực hiện chức năng

*Thiết kế ,tổ chức mô hình tập tin ở bài tập 2B

-Vùng hệ thống

+Bootsector chiếm 2 sector

+ có 1 FAT với kích thước được tính bằng công thức (sẽ được chứng minh ở dưới)

S_F = (1024*1024*n-1024)/ 532992

-Vùng dữ liệu

+cluster thứ 2 sẽ được dùng để lưu password volume

+cluster thứ 3 sẽ lưu bảng thư mục gốc RDET

+ Nội dung tập tin sẽ được lưu từ cluster thứ 4

+nếu RDET hết sẽ kiếm 1 cluster trống để tiếp tục lưu RDET(ở entry cuối RDET trước đó sẽ không lưu thông tin file mà sẽ lưu thông tin RDET tiếp theo)

3.2.1. Tạo/ định dạng volume MyFS.Dat (tối ưu: 80%)

-Tạo 1 file chưa nội dung bootsector đọc từ 1 volume định dạng FAT32 khác

+ Chỉnh sửa các thông số như là: kích thước bootsector, kích thước FAT, số lượng bảng FAT, …(đề yêu cầu tạo 1 thiết kế đơn giản nên không quá trú trọng vào các thông số khác của bootsector như version, tên, nơi sản xuất, …)

+ Ta đọc 1 cấu trúc bootsector từ file tạo sẵn vào. Để không phải gán các offsets ở bootsector không sử dụng bằng 1 giá trị giống nhau (gán 1 byte bất kì cho tất cả các offset không sử dụng), hoặc random các giá trị rác để lưu vào các offsets thì ta sẽ đọc bootsector từ 1 volume khác vào, các offsets không dùng đến ở câu 2B sẽ có giá trị khác 0 và xem nó như file rác

```
int S_F = (1024 * 1024 * n - 1024) / 532992;    //chung minh cong thuc o p
BYTE* byte_sf = dec_to_hex_with_little_endian(S_F, 4);
for (int i = 0; i < 4; i++)
    bootSector.BPB_FATSz32[i] = byte_sf[i];
bootSector.BPB_RsvdSecCnt[0] = BYTE(2);         //size of bootsector area
bootSector.BPB_RsvdSecCnt[1] = BYTE(0);
bootSector.BPB_RootClus[0] = BYTE(3);   //RDET start =3
bootSector.BPB_ClusSavePass[0] = BYTE(2);   //cluster save password =2
bootSector.BPB_NumFATs[0] = BYTE(1);    //number FAT =1
```

- Chứng minh công thức S_F = (1024*1024*n-1024)/ 532992

Gọi: n là kích thước người dùng nhập (chọn): đơn vị MB

S_F là kích thước FAT(sector)

d là kích thước vùng data

Ta mặc định dùng 2 sector để lưu bootsector: S_B=1024B

Kích thước người dùng nhập: 1024*1024*n(B)

Kích thước FAT: 512*S_F(B)

Mỗi volume có 1 vùng hệ thống và 1 vòng dữ liệu

Vùng hệ thống gồm bootsector và FAT

Nên ta có công thức S_B+512*S_F+d = 1024*1024*n (1)

Ta có

$$\begin{cases} 1 \text{ sector của FAT sẽ lưu được } 128 \text{ cluster} \\ 1 \text{ entry có } 64 \text{ byte} \\ 1\ cluster = 8\ sector \end{cases}$$

$$\Rightarrow \begin{cases} \text{có } 128 * 64 \text{ B để lưu thông tin file} \\ 128 * 8 * 512B \text{ để } lưu\ nội\ dung\ file \end{cases}$$

=> 128*8*512*S_F+128*64*S_F=d (2)

Thay (2) vào (1) ta được

S_B+ 512*S_F+ 128* (512*8+64)* S_F= 1024*1024*n

=> 532992*S_F=1024*1024*n - S_B

=> S_F=(1024*1024*n-S_B) /532992

=> S_F=(1024*1024*n-1024) /532992

- Như hình dưới đây,ta chỉ quan tâm tới các offset:

0x0D: số sector trên 1 cluster;

0x0E-0x0F: kích thước bootsector;

0x10: số bảng FAT;

0x24-0x27: kích thước bảng FAT;

0x2C-0x2F: cluster bắt đầu của RDET.

Điều chú ý ở đây là thay vì offset 0x11-0x12 sẽ có giá trị là 0 vì FAT32 không dùng tới, ta tận dụng 2 byte này để lưu cluster chứa password

```
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 08 02 00  ëX.MSDOS5.0.....
00000010  01 02 00 00 00 F8 00 00 3F 00 32 00 80 00 00 00  .....ø..?.2.€...
00000020  00 E0 1F 00 F1 07 00 00 00 00 00 00 03 00 00 00  .à..ñ...........
00000030  01 00 06 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000040  80 00 29 AB 2F 7D F2 4E 4F 20 4E 41 4D 45 20 20  €.)«/}òNO NAME
00000050  20 20 46 41 54 33 32 20 20 20 33 C9 8E D1 BC F4    FAT32   3ÉŽÑ¼ô
00000060  7B 8E C1 8E D9 BD 00 7C 88 56 40 88 4E 02 8A 56  {ŽÁŽÙ½.|ˆV@ˆN.ŠV
00000070  40 B4 41 BB AA 55 00 13 72 10 81 FB 55 AA 75 0A  @´A»ªU..r..ûUªu.
00000080  F6 C1 01 74 05 FE 46 02 EB 2D 8A 56 40 B4 08 00  öÁ.t.þF.ë-ŠV@´..
```

```
00000090  13 73 05 B9 FF FF 8A F1 66 0F B6 C6 40 66 0F B6  .s.¹ÿÿŠñf.¶Æ@f.¶
000000A0  D1 80 E2 3F F7 E2 86 00 C0 ED 06 41 66 0F B7 C9  Ñ€â?÷â†.Àí.Af.·É
000000B0  66 F7 E1 66 89 46 F8 83 7E 16 00 75 39 83 7E 2A  f÷áf‰Føƒ~..u9ƒ~*
000000C0  00 77 33 66 8B 46 1C 66 83 C0 0C BB 00 80 B9 01  .w3f‹F.fƒÀ.».€¹.
000000D0  00 E8 2C 00 E9 A8 03 A1 F8 7D 80 C4 7C 8B F0 AC  .è,.é¨.¡ø}€Ä|‹ð¬
000000E0  84 C0 74 17 3C 32 74 09 B4 0E BB 07 00 00 10 EB  „Àt.<2t.´.».....ë
000000F0  EE A1 FA 7D EB E4 A1 7D 80 EB DF 98 00 16 00 19  î¡ú}ëä¡}€ëß˜....
00000100  66 60 80 7E 02 DF 1A 61 29 00 7C 25 6D BB 91 E6  f`€~.ß.a).|%m»'æ
00000110  79 D6 B5 08 DD DC DD FD 9F 9A 46 50 86 BA A0 CE  yÖµ.ÝÜÝýŸšFP†° Î
00000120  BB 83 73 8D 6F 1E 7C 64 BF EE BB E6 9B 25 AF DE  »ƒs.o.|d¿î»æ›%¯Þ
00000130  24 36 F7 B5 EE 03 81 D2 6A 93 C5 BB 2A 1A EB 23  $6÷µî..Òj"Å»*.ë#
00000140  83 17 78 7F 51 66 DD D7 2A 9A 76 12 86 D6 8A C5  ƒ.x.QfÝ×*šv.†ÖŠÅ
00000150  40 47 25 0D 94 A5 8B CC 65 FE 17 6C 1A BB 7F ED  @G%.”¥‹Ìeþ.l.».í
00000160  15 74 E3 BC 34 6D 02 6E 40 49 75 07 C3 8F 82 82  .tã¼4m.n@Iu.Ã.,,
00000170  D4 C3 D0 52 FD F7 35 9D 09 DD 1A E8 C9 00 1C 3D  ÔÃÐRý÷5..Ý.èÉ..=
00000180  F7 6D 00 08 00 00 00 93 00 00 00 00 C0 96 C9 00  ÷m.....“....À–É.
00000190  DD 6B 15 BD 09 DD 0A FA CF 00 1C 3D F7 6D 00 08  Ýk.½.Ý.úÏ..=÷m..
000001A0  00 00 00 91 00 00 00 00 58 D4 CF 00 D0 71 51 B8  ...'....XÔÏ.ÐqQ¸
000001B0  7A B6 36 89 E5 72 6F 72 FF 0D 0A 58 72 65 73 E4  z¶6‰årorÿ..Xresä
000001C0  20 99 50 79 20 6B 65 79 FD 13 7A 01 7B B8 65 94   ™Py keyý.z.{¸e”
000001D0  C3 AF B5 ED 20 B0 DD DD DD DD DD DE DD 21 48 DD  Ã¯µí °ÝÝÝÝÝÞÝ!HÝ
000001E0  85 09 12 DD DD 6F 15 F5 09 DD 2A E2 97 00 1C 3D  …..ÝÝo.õ.Ý*â—..=
000001F0  F7 6D 00 08 00 00 00 91 AC CC 74 CC 80 8E 55 AA  ÷m.....'¬ÌtÌ€ŽUª
```

*Hạn chế:

- Cần phải có sẵn 1 file bootsector

3.2.2. Thiết lập /Đổi /Kiểm tra mật khẩu truy xuất MyFS (tối ưu: 90%)

Thao tác 1: Thiết lập

- Lấy thông tin về kích thước bootsector (S_B), FAT(S_F), số lượng FAT(N_F), số sector trên 1 cluster (S_C), cluster lưu pass (clus_pass)

```
S_B = reverseByte(bootSector.BPB_RsvdSecCnt, 2);
S_F = reverseByte(bootSector.BPB_FATSz32, 4);
N_F = reverseByte(bootSector.BPB_NumFATs, 1);
Clus_Begin_RDET = reverseByte(bootSector.BPB_RootClus, 4);
S_C = reverseByte(bootSector.BPB_SecPerClus, 1);
```

- Chuyển password thành hệ hexa với mỗi kí tự tương ứng 1 byte

- Tạo 1 sector có giá trị bằng 0 và lưu password vào đầu sector

- Di chuyển tới cluster dùng để lưu password và ghi đè vào

- Đối với trường hợp xóa mật khẩu thì chỉ cần truyền vào tham số password="", khi đó cluster lưu password sẽ có giá trị = 0

```
get_info(bootSector, S_B, S_F, N_F, Clus_Begin_RDET, S_C);
flag = savePassword_volume(password, (S_B + S_F * N_F) * BYTE_PER_SEC, fileName);
```

Thao tác 2: Kiểm tra

- Di chuyển tới cluster lưu password và đổi về chuỗi

- Nếu chuỗi = "" thì volume không có xét password

```
BYTE* buffer;
buffer = readBlock(offset, fileName);
string pass_sec = hex_sector_to_string(buffer, 0, 32);
return pass_sec;
```

Như bootsector ở trên thì ta sẽ có các thông tin sau

S_B= 2; S_F= 2033 ; N_F= 1; S_C= 8; clus_pass= 2

=> Cluster lưu pass nằm ở vị trị (S_B+ N_F* S_F+ (clus_pass - 2) * S_C)* 512

= (2+ 1* 2033+ (2-2) *8) *512 = 1041920 (ở offset 0xFE600)

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

000FE600  64 61 6E 67 31 32 33 34 35 36 00 00 00 00 00 00  dang123456......

000FE610  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Ta sẽ thấy được password ở trên là dang123456

Khi xóa password ta chỉ cần chuyển các byte từ offset 0xFE600 về 0 là được

*Hạn chế

-Mật khẩu xor thẳng vào bootsector và không qua hash table

3.2.3. Liệt kê danh sách các tập tin trong MyFS (tối ưu 90%)

- Lấy thông tin về kích thước bootsector (S_B), FAT (S_F), số lượng FAT(N_F), số sector trên 1 cluster (S_C), cluster bắt đầu RDET (clus_begin_RDET)

- Tiến hành đọc từng sector, trong sector sẽ duyệt từng entry, mỗi entry chiếm 64Byte,nếu Byte bắt đầu có giá trị khác 0xE5 thì sẽ được in ra màn hình

- Những file có thuộc tính (mode=6) là file đã khóa (đã xét password) thì sẽ không in ra kích thước file (ta cần mở khóa các file đó để hiện thị kích thước file)

```
for (int i = 0; i < BYTE_PER_SEC; i += 64) {
    if (buffer[i] == BYTE(0))
        break;
    if (buffer[i] == BYTE(229))     //file da xoa 0xE5
        continue;
    if (buffer[i + 26] != BYTE(0)) {
        for (int j = i; j < i + 8; j++) {
            if (buffer[j] == BYTE(32))
                break;
            cout << buffer[j];
        }
        if (buffer[i + 8] == BYTE(32)) {
            break;
        }
        cout << ".";
        for (int j = i + 8; j < i + 11; j++) {
            cout << buffer[j];
        }
        if (buffer[i + 11] == BYTE(6)) {
            cout << endl;
            continue;
        }
```

- Như bootsector ở trên thì ta sẽ có các thông tin sau

S_B= 2; S_F= 2033; N_F= 1; S_C= 8; clus_begin_REDET= 3

=> cluster bắt đầu RDET nằm ở vị trí

(S_B+ N_F* S_F+ (clus_begin_REDET -2)* S_C)* 512

= (2+ 1* 2033+(3-2)* 8)* 512 = 1046016 (ở offset 0xFF600)

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

```
000FF600  E5 30 20 20 20 20 20 20 64 61 74 02 02 00 00 00  å0      dat.....
000FF610  00 00 00 00 00 00 00 00 00 00 04 00 96 00 00 00  ............-...
000FF620  E5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  å...............
000FF630  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000FF640  66 31 20 20 20 20 20 20 64 61 74 00 02 00 00 00  f1      dat.....
000FF650  00 00 00 00 00 00 00 00 00 00 04 00 8C 0A 00 00  ............Œ...
000FF660  E5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  å...............
000FF670  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Khi liệt kê, kết quả chỉ có f1.dat 2700B

```
chon cac yeu cau
0: thoat
1: Tao, dinh dang volume MyFS.dat
2: Thiet lap, doi, kiem tra mat khau truy xuat MyFS
3: liet ke danh sach cac tap tin trong MyFS
4: dat, doi mat khau truy xuat cho 1 tap tin trong MyFS
5: chep 1 tap tin tu ben ngoai vao MyFS
6: chep 1 tap tin tu MyFS ra ngoai
7: xoa 1 tap tin trong myFS
3
cac tap tin trong volume
f1.dat          2700 B
```

* Hạn chế

- Chỉ tên file liệt kê được kích thước file, chưa thể hiện ngày giờ chỉnh sửa

- chưa thực hiện được trên thư mục con

3.2.4. Đặt /đổi mật khẩu truy xuất cho 1 tập tin trong MyFS (tối ưu 80%)

- Nhập tên file cần đổi

- Kiểm tra xem file có xét password không, nếu có thì yêu cầu nhập password để mở khóa rồi thực hiện bước tiếp theo

- Thiết lập, đổi mật khẩu

+ Nhập password

+ Lấy thông tin sector chứa file, vị trí bắt đầu của entry

+ Đọc sector vào buffer vào gán password vào vị trí bắt đầu của entry+ 33 (byte thứ 1 của entry phụ, vì byte thứ 0 chưa giá trị 0xE5)

+ Ghi lại buffer vào sector vừa đọc

+ Nếu pasword nhập vào =”“ thì entry phụ sẽ chứa giá trị 0, khi đó file không có password nữa

```
chon cac yeu cau
0: thoat
1: Tao, dinh dang volume MyFS.dat
2: Thiet lap, doi, kiem tra mat khau truy xuat MyFS
3: liet ke danh sach cac tap tin trong MyFS
4: dat, doi mat khau truy xuat cho 1 tap tin trong MyFS
5: chep 1 tap tin tu ben ngoai vao MyFS
6: chep 1 tap tin tu MyFS ra ngoai
7: xoa 1 tap tin trong myFS
4
nhap ten file can thiet lap
vi du: f0.dat
ten file: f1.dat
0: Tro ve menu
1: Thiet lap, doi mat khau
2: Kiem tra mat khau
3: xoa mat khau
1
nhap mat khau moimat khau toi da 31 ki tu!!
mat khau: dang123
cap nhat mat khau thanh cong !!!
lan truy cap file tiep theo can nhap mat khau moi
```

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

000FF640  66 31 20 20 20 20 20 20 64 61 74 00 02 00 00 00  f1      dat.....

000FF650  00 00 00 00 00 00 00 00 00 00 04 00 8C 0A 00 00  ............Œ...

000FF660  E5 64 61 6E 67 31 32 33 00 00 00 00 00 00 00 00  ådang123........

000FF670  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Kiểm tra password

+ Di chuyển tới offset chưa password (vị trí bắt đầu cảu entry+ 33) rồi lấy password

+Nếu chuỗi =”“ thì file không có xét password

Như entry ở trên thì password sẽ là dang123

```
0: Tro ve menu
1: Thiet lap, doi mat khau
2: Kiem tra mat khau
3: xoa mat khau
2
mat khau file la: dang123
```

*Hạn chế

- Mật khẩu xor thẳng vào entry và không qua hash table

-chưa thực hiện được cho thư mục con

3.2.5. Chép (Import) 1 tập tin từ bên ngoài vào MyFS (tối ưu 90%)

- Nhập đường dẫn file từ bên ngoài (ví dụ C:\f0.dat)

- Lấy thông tin kích thước bootsector, FAT, số lượng FAT, vị trí bắt đầu RDET số sector mỗi cluster

- Tính số cluster cần để lưu file (BYTE (205) là kí tự kết thúc file)

```
get_info(bootSector, S_B, S_F, N_F, Clus_Begin_RDET, S_C);
int sec_of_file = calculate_sector_of_file(fileSource);//s
int clus_of_file = (sec_of_file / S_C) + 1;//so cluster ma
int index = -1;
int offset = 0;
```

- Lấy offset chưa RDET còn trống (nếu RDET còn đúng 1 entry trống thì kiếm 1 cluster trống khác làm RDET)

```
//get offset RDET
offset_RDET =get_and_update_clus_RDET(S_B, S_F, N_F, Clus_Begin_RDET, S_C, fileDes);
//update FAT
```

- Ý tưởng để mở rộng RDET

+ Trên FAT lấy 1 cluster trống và dùng nó làm RDET

+ Ở entry cuối cùng của RDET ghi cluster index dùng để làm RDET mới vào

+ Trả về offset chứa RDET mới

- Di chuyển tới bảng FAT ,tìm kiếm các cluster liên tiếp sao cho không nhỏ hơn số cluster chứa file để lưu file

```
for (int i = 0; i < S_F; i++) {
    int n;
    buffer = readBlock(S_B * BYTE_PER_SEC+ offset, fileDes);
    for (int j = 0; j < BYTE_PER_SEC; j += 4) {
        if (buffer[j] == BYTE(0)) {
            // tim khu vuc trong(EOF) du lon de chua file
            n= count_empty_cluster(buffer, j);
            if (clus_i_begin == -1) {
                clus_i_begin = j;           //vi tri bat dau cap nhat FAT
                offset_i_begin = offset;    //sector chua clus_i_begin
            }
            cluster_empty += n;
            if (cluster_empty >= clus_of_file) {
                index = clus_i_begin;
                break;
            }
            j += n * 4;
            if (j != 512)
                clus_i_begin = -1;//reset lai vi khong du cluster
        }
    }
    if (index != -1)
        break;
    offset += BYTE_PER_SEC;
}
```

- Tiến hành cập nhật lại giá trị USED ở bảng FAT

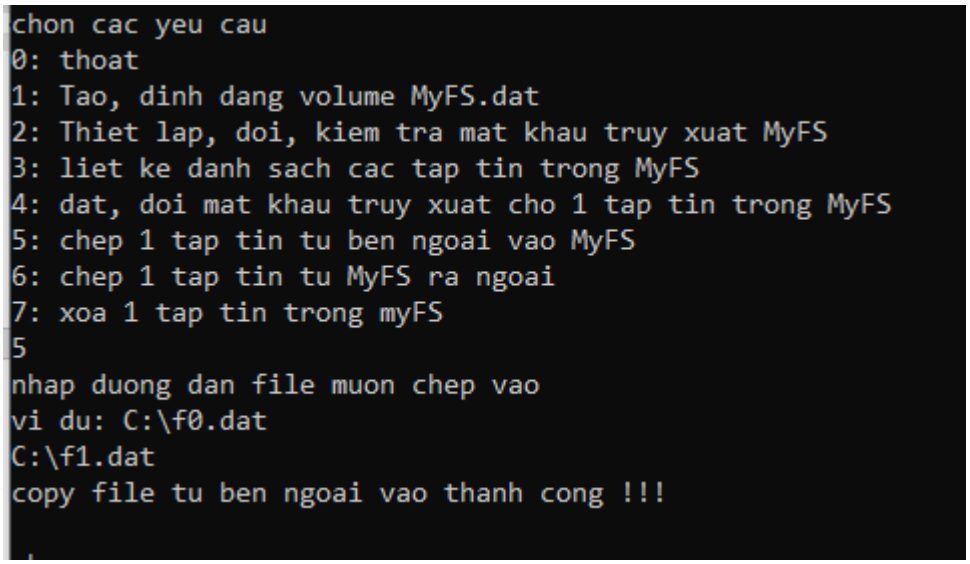
- Tạo entry chính lưu tên file,kích thước và số thứ tự cluster chứa nội dung

- Ở entry phụ chỉ cần byte đầu là 0xE5 là được

```
int fileSize = get_fileSize(fileSource);//lay kick thuoc file
BYTE* main_entry = create_main_entry_file(name, 2, clus_index_in_RDET, fileSize);
BYTE* entry = create_offsets_zero(64);//gan entry =0
for (int i = 0; i < 32; i++)
    entry[i] = main_entry[i];
entry[32] = BYTE(229);              // E5
```

- Di chuyển tới cluster chưa nội dung vào copy nội dung file từ fileSource bên ngoài vào bằng cách đọc từng block (như đọc sector) từ file Source rồi lưu vào sector bên trong volume

```
offset = 0;
for (int i = 0; i < fileSize / BYTE_PER_SEC; i++) {
    buffer = readFile_normal(offset, fileSource);   //doc noi dung file source
    flag = writeBlock((S_B + S_F * N_F + (clus_index_in_RDET - 2) * S_C) * BYTE_PER_SEC + offset,
        buffer,
        fileDes);
    if (!flag)
        return false;
    offset += BYTE_PER_SEC;
}
buffer = readFile_normal(offset, fileSource);   //doc noi dung file source
flag = writeBlock_tail((S_B + S_F * N_F + (clus_index_in_RDET - 2) * S_C) * BYTE_PER_SEC + offset,
    buffer,
    fileDes);
```



* Hạn chế

- Đuôi mở rộng file nhiều hơn 3 kí tự thì chỉ ghi 3 kí tự ,vidu .docx=> .doc

- Chỉ lưu tên chính tối đa 8 kí tự, copy file có tên chính nhiều hơn 8 kí tự sẽ bị lỗi

-chưa thực hiện được khi import vào thư mục con trong volume MyFS

3.2.6. Chép (Outport) 1 tập tin trong MyFS ra ngoài (tối ưu 90%)

- Nhập tên file cần chép ra ngoài (vidu : MyFS.dat\f0.dat)

- Nhập đường dẫn tới file cần chép ra ngoài (vidu: C:\f1.dat)

- Di chuyển đến bang RDET lấy cluster index và file size của file

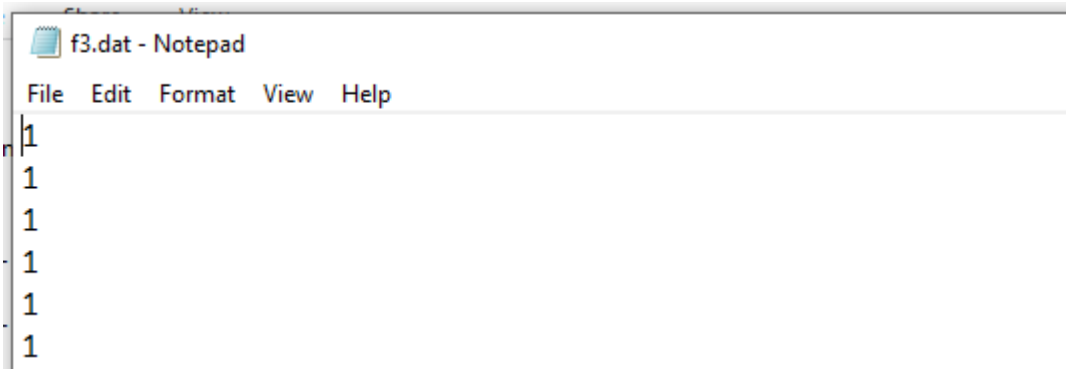
```
while (1) {
    buffer = readFile_normal((S_B + S_F * N_F + (Clus_Begin_RDET - 2) * S_C) * BYTE_PER_SEC + offset,
        fileName_volume);
    if (buffer[0] == BYTE(0))
        return false;
    clus_index = get_clus_i_in_sector(buffer, list_path.back());
    fileSize = get_fileSize_in_volume(buffer, list_path.back());
    if (clus_index > 0 && fileSize>0)
        break;
    offset += BYTE_PER_SEC;
}
```

- Di chuyển đến cluster index ,đọc từng block (sector) rồi ghi vào đường dẫn file bên ngoài đã nhập ở trên

```
offset = 0;
for (int i = 0; i < fileSize / BYTE_PER_SEC; i++) {
    //file name la kieu string nen dung ham doc file binh thuong(FILE*f)
    buffer = readFile_normal((S_B + S_F * N_F + (clus_index - 2) * S_C) * BYTE_PER_SEC + offset,
        fileName_volume);
    flag = writeFile_normal(offset, buffer, fileDes);
    offset += BYTE_PER_SEC;
}
buffer = readFile_normal((S_B + S_F * N_F + (clus_index - 2) * S_C) * BYTE_PER_SEC + offset,
    fileName_volume);
flag = write_tail_file(offset, buffer, fileDes);
```

```
nhap ten file can copy ra ngoai
vidu: MyFS.dat\f0.dat
MyFS.dat\f1.dat
nhap duong dan vi tri file muon copy den(bao gom ten file)
vidu: C:\f1.dat (copy noi dung tu file vao f1.dat)
C:\f3.dat
copy file ra ben ngoai vao thanh cong !!!
```

File f1.dat sau khi chép ra file C:\f3.dat



*Hạn chế

-Tên file chính copy ra ngoài tối đa 8 kí tự

-Chưa thực hiện được cho thư mục con

3.2.7. Xóa 1 tập tin trong MyFS (tối ưu 95%)

-Thực hiện xóa file như trong FAT32 (cho đầu entry =0xE5,cho bằng 0 ở các byte lưu cluster của file trong FAT)

- Nhập đường dẫn file cần xóa trong volume (vidu MyFS.dat\f0.dat)

- Di chuyển đến enrty của file ở bảng RDET và chuyển byte đầu của entry thành 0xE5

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

000FF6C0  E5 30 20 20 20 20 20 20 64 61 74 00 02 00 00 00  å0      dat.....
000FF6D0  00 00 00 00 00 00 00 00 00 00 06 00 96 00 00 00  ..........–...
000FF6E0  E5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  å...............
000FF6F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

- Lấy cluster index của file (ở entry trên cluster index là 6)

- Di chuyển đến bảng quản lí cluster FAT và chuyển các byte từ cluster_index*4 thành 0 cho tới hết khối dữ liệu cuối của tập tin (0x0FFFFFFF)

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

 00000400  F8 FF FF 0F FF FF FF FF FF FF FF 0F FF FF FF 0F  øÿÿ.ÿÿÿÿÿÿÿÿ.ÿÿÿ.
 00000410  FF FF FF 0F FF FF FF 0F 00 00 00 00 00 00 00 00  ÿÿÿ.ÿÿÿ.........
```

* Khi thoát chương trình sẽ tiến hành khóa lại các file có xét password,khóa volume nếu volume có xét password

```
lock_files(file_used,                                          //khoa cac file
    (S_B + N_F * S_F + (Clus_Begin_RDET - 2) * S_C)* BYTE_PER_SEC,
    fileName);
password = getPassword_volume((S_B + S_F * N_F) * BYTE_PER_SEC, fileName);
lock_or_unlock_sector(password, 0, fileName, 0, BYTE_PER_SEC);  //khoa volume
```

Bootsector sau khi mã hóa (XOR với password)

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

00000000  8F 39 FE 2A 62 76 7C 67 00 18 54 61 6C 6F 33 32  .9þ*bv|g..Talo32
00000010  32 36 35 36 64 99 6E 67 0E 32 01 34 B5 36 64 61  2656d™ng.2.4µ6da
00000020  6E 87 2E 32 C2 33 35 36 64 61 6E 67 32 32 33 34  n‡.2Â356dang2234
00000030  34 36 62 61 6E 67 31 32 33 34 35 36 64 61 6E 67  46bang123456dang
00000040  B1 32 1A 9F 1A 4B 96 2F 21 47 7F 73 7E 71 15 16  ±2.Ÿ.K–/!G.s~q..
00000050  44 41 28 26 65 01 01 14 15 16 57 A8 E0 B6 8D C6  DA(&e.....W¨à¶.Æ
00000060  48 BA F4 B8 BD DC 6E 1B B9 64 73 BC 7B 34 EE 37  H°ô¸½Ün.¹ds¼{4î7
00000070  2E D3 70 89 99 61 35 25 16 71 EF 9C 64 98 46 3E  .Óp‰™a5%.qïœd˜F>
```

```
00000080  C3 F7 65 15 6B 99 77 30 D8 19 BF 60 24 D5 66 67  Ã÷e.k™w0Ø.¿`$Õfg
00000090  22 41 36 8D CA C9 EE 90 08 68 87 F4 73 52 3A 80  "A6.ÊÉî..h‡ôsR:€
000000A0  B5 E1 8C 58 C6 D0 B5 34 F5 DB 62 20 08 68 86 FB  µáŒXÆÐµ4õÛb .h†û
000000B0  55 C3 D4 50 ED 27 96 E4 4F 24 33 41 0C B5 1A 4B  UÃÔPí'–äO$3A.µ.K
000000C0  6E 10 02 54 B8 72 29 50 E7 A1 62 DC 31 B2 8A 35  n..T¸r)Pç¡bÜ1²Š5
000000D0  35 DE 48 61 87 CF 32 93 CB 49 B5 F2 18 EA 9E CB  5ÞHa‡Ï2"ËIµò.êžË
000000E0  B5 F2 47 23 09 04 10 68 DA 69 8A 35 33 34 25 DD  µòG#...hÚiŠ534%Ý
000000F0  8A C0 94 1A DA D6 92 49 B5 DD BB F9 6E 71 31 2B  ŠÀ".ÚÖ'IµÝ»ùnq1+
00000100  55 54 B5 48 66 BE 74 06 18 32 4F 11 58 8D F5 87  UTµHf¾t..2O.X.õ‡
00000110  17 B1 84 3A EE E8 E8 CB FB FB 28 37 B7 88 93 FA  .±„:îèèËûû(7·ˆ"ú
00000120  8E B5 17 EC 01 79 4D 56 8C DA 8E D0 FF 44 C1 B9  Žµ.ì.yMVŒÚŽÐÿDÁ¹
00000130  15 04 C4 81 DB 35 E5 B3 04 F4 F4 89 19 2E DE 15  ..Ä.Û5å³.ôô‰..Þ.
00000140  E7 76 16 18 60 54 EE E3 1F AC 12 73 E8 B1 BB F7  çv..`Tîã.¬.sè±»÷
00000150  73 73 10 3B F0 C4 E5 AB 54 CC 24 58 2F 8D 1B 8C  ss.;ðÄå«TÌ$X/..Œ
00000160  7B 13 D2 8E 07 59 37 58 24 28 1B 60 F2 BD B1 B6  {.ÒŽ.Y7X$(.`ò½±¶
00000170  E1 F5 B4 33 93 90 04 AF 3A E9 2F DE AD 61 72 5A  áõ´3"..¯:é/Þ.arZ
00000180  C6 5F 33 3C 35 36 64 F2 6E 67 31 32 F3 A2 FC 36  Æ_3<56dòng12ó¢ü6
00000190  B9 0A 7B DA 38 EF 39 CE FA 36 78 5C 99 0A 31 3A  ¹.{Ú8ï9Îú6x\™.1:
000001A0  33 34 35 A7 64 61 6E 67 69 E6 FC 34 E5 47 35 D9  345§dangiæü4åG5Ù
000001B0  14 D1 07 BB D6 46 5A 44 9B 6C 64 3F 43 57 40 D0  .Ñ.»ÖFZD›ld?CW@Ð
000001C0  15 AF 34 18 4E 0C 54 4B CE 27 4F 37 1F D9 0B F3  .¯4.N.TKÎ'O7.Ù.ó
000001D0  F2 9D 86 D9 15 86 B9 BC B3 BA EC EC EE 15 7D EB  ò.†Ù.†¹¼³°ììî.}ë
000001E0  E1 68 7C BA EC 5D 26 C1 3C EB 4E 83 F9 67 2D 0F  áh|°ì]&Á<ëNƒùg-.
000001F0  C4 59 35 3E 64 61 6E F6 9D FE 47 F8 B5 B8 31 CB  ÄY5>danö.þGøµ¸1Ë
```

Các file có password sẽ bị mã hóa(XOR với password file đó)

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
000FF640  66 31 20 20 20 20 20 20 64 61 74 06 02 00 00 00  f1      dat.....
000FF650  64 61 6E 67 31 32 33 64 61 6E 63 31 BE 39 64 61  dang123danc1¾9da
000FF660  8B 03 50 5C 54 55 53 5D 67 31 32 33 64 61 6E 67  ‹.P\TUS]g123dang
000FF670  31 32 33 64 61 6E 67 31 32 33 64 61 6E 67 31 32  123dang123dang12
```

*Hạn chế

-Chưa thực hiện được khi xóa file bên trong thư mục con

4. Tài liệu tham khảo

Cấu trúc FAT32

http://elm-chan.org/docs/fat_e.html

Sách giáo trình Trần Trung Dũng-Pham Tuấn Sơn-Hệ Điều Hành-Trường Đại học Khoa Học Tự Nhiện-ĐHQG TpHCM-Chương 7 Quản lí tập tin

Đọc ghi file cho volume

https://docs.microsoft.com/en-us/windows/win32/api/fileapi/

5. Link demo

*Link video demo câu 2B https://www.youtube.com/watch?v=OByrTvgNhTQ