



fit@hcmus

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN**

BÁO CÁO BÀI THỰC HÀNH GIỮA KỲ

Các thành viên trong nhóm:

- | | |
|--------------------|----------------|
| 1. Lê Trọng Anh Tú | MSSV: 20127091 |
| 2. Phan Tuấn Khải | MSSV: 20127524 |
| 3. Lê Đăng Khoa | MSSV: 20127533 |

Giảng viên hướng dẫn: Ths. Thái Hùng Văn

Mục lục

1	Giới thiệu	3
2	Kết quả đạt được	3
2.1	Một số lưu ý	3
2.1.1	Hướng dẫn biên dịch	3
3	Phần trả lời câu hỏi	3
3.1	Câu 1	3
3.1.1	Mã nguồn chương trình	3
3.1.2	Demo	5
3.2	Câu 2	9
3.2.1	Thiết kế kiến trúc tổ chức cho hệ thống tập tin	9
3.2.2	Kiến trúc của volume được thiết kế trong chương trình	14
3.2.3	Viết chương trình thực hiện các chức năng	15
3.2.3.1	Chức năng tạo và định dạng volume	15
3.2.3.2	Chức năng xác minh Volume cho các lần truy xuất tiếp theo	16
3.2.3.3	Chức năng thiết lập, thay đổi, kiểm tra mật khẩu truy xuất volume:	17
3.2.3.4	Chức năng liệt kê thông tin volume và danh sách tập tin bên trong:	18
3.2.3.5	Chức năng sao chép tập tin từ bên ngoài vào trong Volume::	20
3.2.3.6	Chức năng đặt, đổi mật khẩu cho tập tin trong volume:	21
3.2.3.7	Chức năng chép một tập tin trong Volume ra ngoài:	23
3.2.3.8	Xoá 1 tập tin trong Volume:	24
	Tài liệu	26

1 Giới thiệu

Bài thực hành giữa kỳ môn Hệ điều hành được thực hiện bởi ba thành viên đề cập ở trang bìa. Nội dung của bài tập lần này xoay quanh chương *Hệ thống quản lý tập tin* đã được học ở trên lớp.

Toàn bộ mã nguồn trong thư mục *Source* và mã nguồn được trình bày trong các phần tiếp theo được nhóm sử dụng ngôn ngữ C/C++ để viết.

Dưới đây sẽ là phần phân công công việc, các phần đã làm được/chưa làm được và trình bày chi tiết câu trả lời của nhóm.

2 Kết quả đạt được

Trong thời gian thực hiện (từ 10/03/2022 đến 22/03/2022), nhóm đã hoàn thành 100% các yêu cầu được đưa ra trong bài tập. Dưới đây là bảng phân công công việc của nhóm.

STT	Sinh viên thực hiện	Nội dung	Mức độ hoàn thành
1	Phan Tuấn Khải	Viết 4 hàm đọc/ghi dãy sector và đọc/ghi 1 số nguyên trong sector	100%
2	Lê Trọng Anh Tú	Thiết kế kiến trúc tổ chức cho hệ thống tập tin (mô hình FATscript)	100%
3	Lê Đăng Khoa	Viết chương trình thực hiện 7 chức năng cho hệ thống tập tin	100%

2.1 Một số lưu ý

2.1.1 Hướng dẫn biên dịch

- Câu lệnh biên dịch: `g++ -std=c++17 *.cpp -o *.exe && *.exe`
- Lưu ý: Phải chạy với quyền administrator

3 Phần trả lời câu hỏi

3.1 Câu 1

Viết 04 hàm đọc/ghi dãy sector, đọc/ghi 1 số nguyên trong sector theo như mô tả ở các bài ôn tập và bài KT LTGK.

3.1.1 Mã nguồn chương trình

Để đọc và ghi dãy sector, ta cần sử dụng một số hàm có sẵn trong thư viện hỗ trợ (*fileapi.h* và *windows.h*) được liệt kê dưới đây.

```
1 // Create file to read data
2 HANDLE CreateFile(dsk, GENERIC_READ, FILE_SHARE_VALID_FLAGS, 0, OPEN_EXISTING, 0, 0);
3
4 // Set file pointer to n (numSect * 512)
5 void SetFilePointer(hDisk, numSect * 512, 0, FILE_BEGIN);
6
7 // Read data from sector
```

```

8     void ReadFile(hDisk, buf, size, &dwRead, 0);
9
10    // Write to from buf to sector
11    void WriteFile(hDisk, buf, strlen(buf), &dwWrite, 0);
12

```

Tiếp theo là phần mã nguồn của 2 hàm đọc/ghi 1 dãy sector. Bắt đầu từ đây người đọc có thể xem chi tiết mã nguồn tại *cau1.cpp* trong thư mục *Source* đi kèm.

```

1    // Read <nS> sectors from <dsk> at sector <numSect> to <buf>
2    bool ReadSector(const char* dsk, char *& buf, int nS, unsigned int numSect){
3        DWORD dwRead;
4        HANDLE hDisk = CreateFile(dsk, GENERIC_READ, FILE_SHARE_VALID_FLAGS, 0, OPEN_EXISTING
, 0, 0);
5        if(hDisk == INVALID_HANDLE_VALUE){//Check create file success or not
6            CloseHandle(hDisk);
7            return 0;
8        }
9        SetFilePointer(hDisk, numSect * 512, 0, FILE_BEGIN);//Move file pointer to sector n =
numSect * 512
10
11        ReadFile(hDisk, buf, nS * 512, &dwRead, 0);//Read nS * 512 byte at sector n = numSect
* 512
12        SetFilePointer(hDisk, 0, 0, FILE_BEGIN);
13        CloseHandle(hDisk);
14        return 1;
15    }
16
17    // Write <nS> sectors in <dsk> at sector <numSect> from <buf>
18    bool WriteSector(char* dsk, const char* buf, int nS, unsigned int numSect){
19        DWORD dwWrite;
20        HANDLE hDisk = CreateFile(dsk, GENERIC_WRITE, FILE_SHARE_VALID_FLAGS, 0,
OPEN_EXISTING, 0, 0);
21        if(hDisk == INVALID_HANDLE_VALUE){//Check create file success or not
22            CloseHandle(hDisk);
23            return 0;
24        }
25        SetFilePointer(hDisk, numSect * 512, 0, FILE_BEGIN);
26        WriteFile(hDisk, buf, 512 * nS, &dwWrite, 0);//Write 512 * bS byte to sector
27        SetFilePointer(hDisk, 0, 0, FILE_BEGIN);
28        CloseHandle(hDisk);
29        return 1;
30    }
31

```

Sau khi đã có được hai hàm đọc/ghi dãy sector, ta tiếp tục thực hiện việc cài đặt hai hàm đọc/ghi một số nguyên vào offset 0 của sector n. Nhóm có cài đặt thêm năm hàm phụ phục vụ cho công việc trên, phần mã nguồn chi tiết cài đặt từng hàm hỗ trợ này sẽ nằm trong file đã đề cập trước đó.

```

1    // Function to convert integer to hex
2    char* toHex(int t);
3
4    // Function to convert back hex to integer
5    int toInt(char* buf);
6
7    // Function to save data at offset o
8    void toOffset(char*& buf, char* t, int offset);

```

```

9
10 // Function to get data at offset o
11 void getOffset(char*& buf, char* t, int offset);
12

```

Và dưới đây là mã nguồn của hai hàm đọc/ghi 1 số nguyên.

```

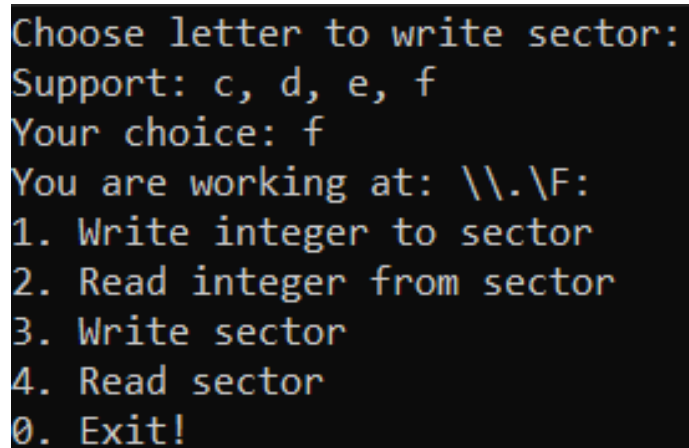
1 // Read 1 integer in sector
2 pair<bool, int> readInt(char* dsk, int numS, int offset){
3     if(offset < 0 || offset > 508)
4         return make_pair(false, -1);
5     int res = 0;
6     char* buf = new char[512];
7     bool ok = ReadSector(dsk, buf, 1, numS);
8     char* r = new char[4];
9     getOffset(r, buf, offset);
10    res = toInt(r);
11    delete[] buf;
12    delete[] r;
13    return make_pair(ok, res);
14 }
15
16 // Write 1 integer into sector
17 bool writeInt(char* dsk, int numS, int num, int offset){
18     char* buf = new char[512];
19     char* t = toHex(num);
20     if(offset < 0 || offset > 508)
21         return 0;
22     bool ok1 = ReadSector(dsk, buf, 1, numS);
23     toOffset(buf, t, offset);
24     bool ok = WriteSector(dsk, buf, 1, numS);
25     delete[] buf;
26     delete[] t;
27     return ok & ok1;
28 }
29

```

3.1.2 Demo

– Demo chương trình:

Sau khi biên dịch xong và chạy, chương trình hiện ra menu như hình dưới



```

Choose letter to write sector:
Support: c, d, e, f
Your choice: f
You are working at: \\.\F:
1. Write integer to sector
2. Read integer from sector
3. Write sector
4. Read sector
0. Exit!

```

Tới đây thì người dùng sẽ nhập 1 ký tự đại diện cho ổ đĩa mình muốn thực hiện và tiến hành đọc/ghi sector. Để thuận tiện cho demo chương trình nhóm chọn ký tự **f** (tương ứng với ổ đĩa F). Sau khi nhập vào **f** và nhấn **Enter**, chương trình sẽ có 5 lựa chọn cho người dùng:

- **1**: Ghi số nguyên vào sector.
- **2**: Đọc số nguyên từ sector.
- **3**: Ghi sector.
- **4**: Đọc sector.
- **0**: Thoát chương trình.

Khi người dùng nhập vào **1** thì chương trình yêu cầu nhập vào ordinal của sector, offset trên sector và số nguyên muốn ghi như hình sau:

```
Input your choice: 1
Input ordinal numbers of sector: 20
Input offset: 0
Input integer: 123456
Write to sector success!
```

+ Nếu ghi thành công thì chương trình sẽ thông báo:

Write to sector success!

+ Nếu không thành công chương trình sẽ thông báo:

ERROR! Can not write to sector!

Khi người dùng nhập vào **2**, chương trình yêu cầu người dùng nhập vào ordinal của sector và vị trí offset trên sector để đọc số nguyên như hình sau:

```
Input your choice: 2
Input ordinal numbers of sector: 20
Input offset: 0
Read success!, Result: 123456
```

+ Nếu đọc thành công chương trình sẽ thông báo: **Read success!, Result: ...**

+ Nếu không thành công chương trình sẽ thông báo: **ERROR! Can not read integer from sector!**

Khi người dùng nhập **3** chương trình sẽ yêu cầu người dùng nhập số sector muốn ghi, vị trí sector muốn ghi và dữ liệu ghi vào sector như sau:

```
Input your choice: 3
How many sector want to write: 2
Index sector to write: 17
Input text: OS_20CLC07
Write success!
```

Khi người dùng nhập **4** chương trình sẽ yêu cầu người dùng nhập vào số sector muốn đọc và vị trí của sector đó như sau:


```
Input your choice: 0
Exit program!!!
```

Cuối cùng nhóm thực hiện việc đọc/ghi nhiều số nguyên cùng một lúc.

```
Input your choice: 1
Input ordinal of sector: 20
Input offset: 0
Input integer: 123
Write to sector success!
1. Write integer to sector
2. Read integer from sector
3. Write sector
4. Read sector
0. Exit!
Input your choice: 1
Input ordinal of sector: 20
Input offset: 4
Input integer: 456
Write to sector success!
```



```

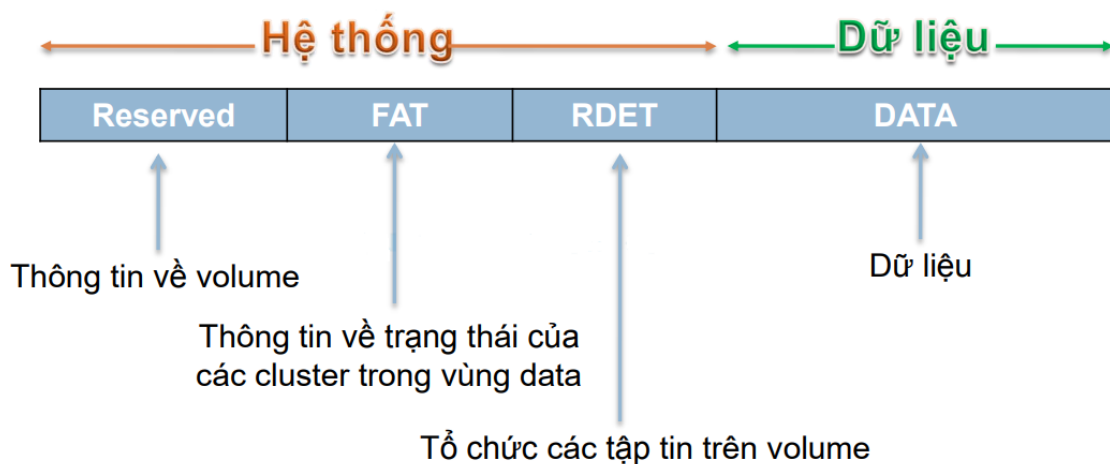
Input your choice: 2
Input ordinal of sector: 20
Input offset: 0
Read success!, Result: 123
1. Write integer to sector
2. Read integer from sector
3. Write sector
4. Read sector
0. Exit!
Input your choice: 2
Input ordinal of sector: 20
Input offset: 4
Read success!, Result: 456

```

3.2 Câu 2

3.2.1 Thiết kế kiến trúc tổ chức cho hệ thống tập tin

Trước khi bước vào thiết kế, ta cùng tìm hiểu khái quát hệ thống tập tin FAT – File Allocation Table.

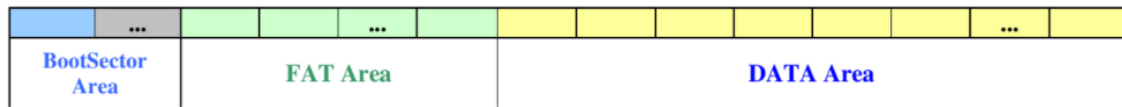


Hình 1: Cấu trúc của FAT

FAT có 3 phiên bản khác nhau: FAT12, FAT16 (2 FAT này thường được gọi là FAT bình thường) và FAT32 (hay FAT mở rộng). Chữ số gắn với FAT là số lượng bit của mỗi phần tử, ví dụ với FAT12 mỗi phần tử đều có

12 bit. Ở đây chúng ta sẽ tập trung vào kiến trúc FAT32 để từ đó xây dựng mô hình và thiết kế kiến trúc tổ chức cho một hệ thống tập tin.

Đối với FAT32, phần RDET sẽ không nằm trong vùng hệ thống mà nằm luôn trên phần DATA. Như vậy có thể xem kiến trúc FAT32 gồm 3 thành phần Boot Sector – FAT – DATA.



Hình 2: Kiến trúc FAT32

• Vùng Boot Sector (Vùng Reserved)

- Gồm các sector đầu tiên trong volume.
- Thường chứa thông tin mô tả về volume hoặc 1 đoạn chương trình để boot hệ điều hành.
- Thông tin trên vùng này rất nhiều nhưng nhóm chỉ liệt kê các thông tin được xem là quan trọng nhất cho phần thiết kế kiến trúc tập tin sau này (Người đọc có thể xem tổ chức thông tin chi tiết trong FAT32 tại [đây](#)).

Offset	Số byte	Nội dung
B	2	Số byte trên sector (thường 1 sector có 512 byte)
D	1	S_C : số sector trên cluster
E	2	S_B : số sector thuộc vùng Boot Sector
10	1	N_F : số bảng FAT (thường có 2 bảng FAT)
20	4	S_V : kích thước volume
24	4	S_F : kích thước 1 bảng FAT
2C	4	Cluster bắt đầu của RDET
30	2	Sector chứa thông tin phụ về cluster trống
32	2	Sector chứa bản lưu của Boot Sector
52	8	Loại FAT, là chuỗi "FAT32"

• Vùng FAT

- Được tổ chức theo hình thức danh sách liên kết kết hợp chỉ mục.
- Vì tính chất quan trọng của bảng quản lý cluster nên thường có N_F bảng để phòng tránh việc hư hỏng.
- Do nằm kế sau vùng Boot Sector nên vị trí bắt đầu của FAT là kích thước của Boot Sector (S_B).
- Với mỗi bảng FAT ta có các thông tin về trạng thái của các cluster trong vùng dữ liệu. (Cluster bị hư – BAD, Cluster chưa được sử dụng – FREE, Cluster chứa dữ liệu - EOF hay chỉ số cluster nào đó liền sau).

• Bảng thư mục

- Là một dãy phần tử entry chứa thông tin của một tập tin trên volume (hoặc là phần tử trống - chưa thuộc 1 tập tin nào).
- Gồm bảng thư mục gốc (RDET) và bảng thư mục con (SDET).
- Mỗi entry chiếm 32 byte, được tổ chức như hình dưới đây.
- Khi xóa 1 Entry, chỉ có byte đầu tiên được chuyển thành E5 chứ không phải là 0.
- **RDET:** Do mỗi entry có 32 byte nên mỗi sector sẽ chứa đúng 16 entry. Vì vậy số entry của RDET thường là bội của 16.

OFFSET	ĐỘ DÀI (byte)	NỘI DUNG
0h (0)	8	Tên chính của tập tin
8h (8)	3	Tên mở rộng
Bh (11)	1	Thuộc tính (0-0-A-D-V-S-H-R) Nếu có giá trị là 0x0F thì entry này sử dụng cho LFNs
Ch (12)	10	Không dùng
16h (22)	2	Giờ cập nhật tập tin
18h (24)	2	Ngày cập nhật tập tin
1Ah (26)	2	Cluster bắt đầu
1Ch (28)	4	Kích thước tập tin

Hình 3: Cấu trúc của mỗi Entry

Phía trên là toàn bộ kiến thức khái quát về kiến trúc FAT32. Từ đây nhóm sẽ tự thiết kế kiến trúc tổ chức cho hệ thống tập tin của riêng mình.

Lấy ý tưởng từ kiến trúc FAT32 cùng với việc bỏ đi một số thông tin không cần thiết cho việc đơn giản hoá mô hình, nhóm chia thiết kế ra thành 4 vùng liên nhau như hình dưới đây:



Hình 4: Thiết kế mô hình tập tin tự tạo

i. Vùng *Information about volume*

Đây là vùng chứa những thông tin quan trọng của volume. Vùng này thường được đặt ở đầu volume trong đa số các thiết kế (và thiết kế này cũng vậy). Khi mở volume lên, chương trình sẽ đọc nội dung phần này trước để lấy thông tin cơ bản của volume. Về thông tin chi tiết của vùng này, người đọc có thể tham khảo dưới đây.

Offset	Số byte	Nội dung
0	16	Tên volume (bao gồm cả phần mở rộng)
10	4	Mật khẩu cho volume (có thể có hoặc không tùy người sử dụng)
14	2	Số byte trên mỗi sector
16	1	S_C : số sector trên cluster
17	2	S_B : số sector thuộc vùng Information about volume
19	1	N_F : số bảng FAT
1A	4	S_V : kích thước volume
1E	4	S_F : kích thước 1 bảng FAT
22	4	Vị trí bắt đầu của vùng FAT
26	4	Vị trí bắt đầu của vùng Entry table
2A	4	Kích thước của vùng Entry table
2E	2	Sector chứa bản lưu của Boot Sector
30	8	Loại kiến trúc tổ chức tập tin đang sử dụng

- *Tên volume:*

- Ngoài phần tên chính ra thì kiến trúc này lưu thêm cả phần mở rộng. Tên được lưu trong mô hình theo định dạng sau đây.

<tên volume>.<phần mở rộng của volume>

- Chiều dài tối đa cho tên volume (bao gồm cả phần mở rộng) là 128 kí tự. Không có chiều dài tối thiểu của tên.
- Không phân biệt chữ hoa và chữ thường.
- Không gõ có dấu, cho phép có khoảng trắng trong tên.
- Nếu tên file đã tồn tại trước đó → Xoá file cũ đi và tạo lại file mới cùng tên với các thuộc tính mới.
- *Mật khẩu cho volume*: các thuộc tính giống như tên của volume nhưng độ dài tối đa của mật khẩu là 32 kí tự.
 - Khi người dùng đã đặt mật khẩu cho volume thì nội dung trong volume này rất quan trọng và cần hệ điều hành bảo vệ kĩ lưỡng. Như vậy nếu ta cứ lưu mật khẩu xuống bộ nhớ mà không thực hiện mã hoá thì rất nguy hiểm, những kẻ xấu có thể lấy được mật khẩu dễ dàng và bán dữ liệu đó làm người dùng phá sản.
 - Cách đơn giản để mã hoá mật khẩu mà nhóm sử dụng ở đây đó là chuyển từ chuỗi mật khẩu đó thành một số nguyên. Công thức chuyển mà nhóm dùng như sau:
$$\text{hash}(\text{password}) = \sum_{i=0}^{n-1} (\text{password}[i] \cdot (i + 1))$$
 - Như vậy cho dù kẻ xấu có nắm được kiến trúc tập tin và truy cập vào để xem nội dung thì vẫn không biết mật khẩu của ta là gì.
- Số byte trên mỗi sector được mặc định là 512 byte.
- Số sector trên 1 cluster (S_C) được mặc định là 4.
- Với S_B là số sector thuộc vùng Information about volume, ta có được kích thước vùng này = $S_B * 512$ (byte).
- Ở đây nhóm sẽ thiết kế 1 bảng FAT cho mô hình này ($N_F = 1$).
- Kích thước volume được lưu bằng số nguyên không dấu 4 byte dưới dạng MB (ví dụ 4 byte đó lưu số 1024 → volume có kích thước là 1024MB).
- Tại offset 24 được mô tả ở bảng trên, dùng để làm dấu hiệu nhận biết volume có kiến trúc gì, loại kiến trúc tổ chức tập tin mà nhóm đang thiết kế có chuỗi là "FATscript".

ii. Vùng *FAT*

Đây là phần chứa bảng quản lý cluster theo hình thức danh sách liên kết kết hợp với chỉ mục với quy định:

- Nếu phần tử i trên bảng quản lý có giá trị là FREE → Vùng dữ liệu tại vị trí i trống.
- Nếu phần tử i trên bảng quản lý có giá trị là BAD → Vùng dữ liệu tại vị trí i đã bị hư.
- Nếu giá trị khác BAD và FREE → Phần DATA tương ứng tại vị trí đó đang chứa dữ liệu tập tin. Nếu phần tử i có giá trị là K nào đó thì phần kế tiếp của nội dung tập tin tại i chính là nội dung tại K , còn nếu giá trị tại phần tử i là EOF thì báo hiệu i chính là phần nội dung cuối cùng của một tập tin.

Do chứa toàn bộ thông tin trạng thái của vùng dữ liệu nên ta cần cho vùng FAT này 1 mảng một chiều có số lượng cluster bằng với số lượng cluster tại vùng DATA.

Với các trạng thái tại từng ô, ta có thể giả sử FREE có giá trị là 0, BAD có giá trị là -1, EOF có giá trị là 1, các giá trị khác sẽ phụ thuộc theo vùng dữ liệu tiếp theo tại vị trí nào. Lí do ta có thể gán 0 vào FREE còn 1 vào EOF bởi ta sẽ bắt đầu vùng dữ liệu bằng cluster 2, như vậy cluster 0 và 1 không được để ý đến (2 cluster đó lúc này có thể chứa rác).

iii. Vùng *Entry table*

Đây là vùng dùng để quản lý từng tập tin trong vùng DATA – mỗi entry quản lý 1 tập tin hay 1 thư mục con. Nội dung về cấu trúc của 1 Entry được trình bày trong bảng sau.

Offset	Số byte	Nội dung
0	32	Tên tập tin (bao gồm cả phần mở rộng)
20	4	Mật khẩu cho tập tin (có thể có hoặc không tùy người sử dụng)
24	4	Kích thước của tập tin
28	4	Thời điểm (ngày tháng năm + giờ phút giây) tạo tập tin
2C	4	Thời điểm (ngày tháng năm + giờ phút giây) chỉnh sửa tập tin lần cuối
30	4	Thời điểm (ngày tháng năm + giờ phút giây) mở tập tin lần cuối
34	2	Cluster bắt đầu của tập tin
36	1	Các thuộc tính của tập tin

- Tên tập tin tương tự như phần tên volume đã trình bày ở trên. Chỉ khác ở việc tên tập tin có độ dài tối đa là 256 kí tự.
- Mật khẩu cho tập tin cũng có các thuộc tính như phần mật khẩu cho volume.
- Kích thước của tập tin được lưu dưới dạng số nguyên không dấu 4 byte theo đơn vị byte.
- Với mỗi Entry quản lý 1 tập tin hay 1 thư mục con thì việc nhận biết được vị trí bắt đầu của dữ liệu này ở đâu là rất quan trọng. Vì vậy ta dành 2 byte cho việc lưu Cluster bắt đầu của tập tin do Entry quản lý chúng.
- Về cách lưu ngày tháng năm và giờ phút giây của 1 tập tin nhất định:
 - Ta sử dụng 2 byte đầu cho việc lưu ngày tháng năm và 2 byte sau đó cho việc lưu giờ phút giây.
 - Với 2 byte ở ngày tháng năm, ta cho 5 bit thấp nhất để lưu trữ ngày, 4 bit ở giữa để lưu trữ tháng và 7 bit cao nhất để lưu trữ năm (đối với năm thì ta sẽ lưu con số = năm – 1980). Ví dụ: hệ thống của chúng ta cần lưu ngày 18/10/2006.
 - * Với ngày 18, ta lưu 5 bit: $18 = 1'0010_2$
 - * Với tháng 10, ta lưu 4 bit: $10 = 1010_2$
 - * Với năm 2006, ta lưu số $2006 - 1980 = 26$ dưới dạng 7 bit như sau: $26 = 001'1010_2$
 - * Vậy 2 byte của ta sẽ là $0011'0101'0101'0010_2 = 3552_{16}$
 - * Về cách lưu như thế này, ta sẽ luôn có đủ bit để lưu 31 ngày và 12 tháng trong năm. Đối với năm thì ta sẽ lưu được trong khoảng $[1980, 2107]$ ($2107 = 1980 + 2^7 - 1$). Tới năm đó ta đã có thể sử dụng những mô hình tiên tiến hơn và có thể lưu chính năm mình đang sống mà không còn trừ đi cho 1 số nào đó nữa.

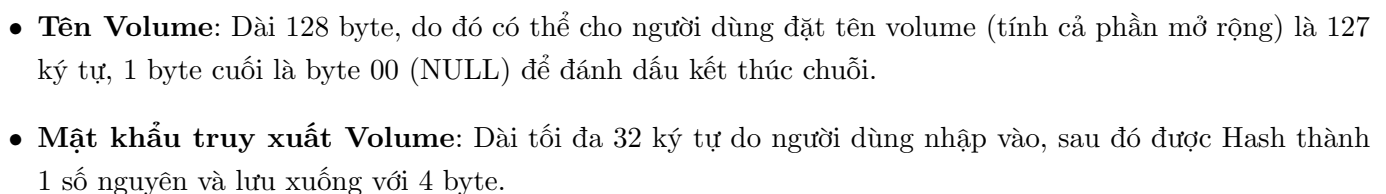
- * Giờ chiếm 5 bit đầu tiên, phút chiếm 6 bit tiếp theo và giây chiếm 5 bit cuối cùng (Đối với giây ta lưu phần nguyên của giây chia cho 2).
- * Với cách lưu này, ta sẽ không thể lưu chính xác cả 3 giá trị được vì giây lúc này sẽ gây ra sai số. Nhưng thường sai số ở giây (cụ thể là 1 – 2 giây) sẽ không quá đáng kể mà còn tối ưu về bộ nhớ (nếu ta lưu đầy đủ thì sẽ mất ít nhất 24 bit – tương ứng với 3 byte).

- **Archive:** thuộc tính lưu trữ, cho biết tập tin đã được backup hay chưa bằng lệnh backup của hệ điều hành.
- **Directory:** thuộc tính thư mục, nếu entry có thuộc tính này thì tập tin tương ứng không phải là một tập tin bình thường mà là một thư mục.
- **System:** thuộc tính hệ thống, cho biết tập tin có phải thuộc hệ điều hành không.
- **Hidden:** thuộc tính ẩn, ở trạng thái mặc định hệ thống sẽ không hiển thị tên của các tập tin này khi liệt kê danh sách tập tin.
- **ReadOnly:** thuộc tính chỉ đọc, khi tập tin có thuộc tính này hệ thống sẽ không cho phép chỉnh sửa hoặc xóa.

Dùng để chứa nội dung của các tập tin trong volume. Ban đầu các dữ liệu để được đưa vào tuần tự theo thứ tự file được import vào. Tuy nhiên sau khi xóa lại gây ra từng chỗ trống đơn lẻ, lúc này bảng FAT sử dụng danh sách liên kết kết hợp chỉ mục sẽ giúp ích tương đối lớn cho việc khai thác tối ưu bộ nhớ volume.

Để thuận tiện cho việc viết chương trình thực hiện các chức năng trong volume, nhóm quyết định sử dụng kiến trúc được miêu tả dưới đây.

1. Phần quản lý (System Area)



- **Kích thước volume:** Dùng 4 byte số nguyên không dấu để lưu kích thước volume dưới dạng byte.
- **Số lượng tập tin mà Volume quản lý:** dùng 4 byte số nguyên không dấu để lưu trữ.
⇒ Tổng độ dài của phần này là 140 byte.

Với mỗi file được thêm vào Volume, phần Data Area sẽ ghi 144 byte tuần tự gồm:

- **Tên tập tin:** Có độ dài tương tự như tên Volume.
- **Mật khẩu truy xuất tập tin:** Tương tự như mật khẩu truy xuất Volume.
- **Kích thước tập tin:** Tương tự cách lưu kích thước Volume.
- **Chỉ mục của sector bắt đầu:** dùng 4 byte số nguyên không dấu để lưu trữ chỉ mục (index) của sector đầu tiên chứa dữ liệu của tập tin.
- **Số sector được cấp phát:** dùng 4 byte số nguyên không dấu để lưu trữ số sector được cấp phát cho dữ liệu đó (Vì mỗi sector chỉ nên chứa dữ liệu của đúng 1 tập tin, do đó nếu tập tin có kích thước chưa đủ 1 sector thì các byte của sector đó sẽ ghi NULL và không thuộc về tập tin nào khác).

2. Phần dữ liệu (Data Area): Bắt đầu từ sector thứ 10 (byte thứ 5120) trở về sau là phần chứa dữ liệu của tập tin (Data Area).

3.2.3 Viết chương trình thực hiện các chức năng

Bảng menu chính của chương trình:

```
===== FILE SYSTEM MENU ===== (Verified with myFS.dat)
[1]. Verify your volume to access (if exist)
[2]. Create and Format new volume
[3]. File listing and display information volume
[4]. Change password of Volume
[5]. Import new file to volume
[6]. Export file from volume
[7]. Change password of file
[8]. Delete file in volume
[0]. Quit
```

3.2.3.1 Chức năng tạo và định dạng volume

Mã nguồn:

```
1 bool createFormatVolume() {
2     int sizeInMegabyte = 0;
3     cout << "+ Name for volume (with extensions): ";
4     cin.getline(VolumeName, sizeFileName);
5     cout << "+ Enter size of volume (in MB): ";
6     cin >> sizeInMegabyte;
7     cin.ignore(1);
8     if (sizeInMegabyte <= 0) {
9         cout << "\t==> Size is too small to create" << endl; return false;
10    }
11    int numOfFiles = 0;
12    sizeVolume = 1048576 * sizeInMegabyte;
```

```

13  ofstream out;
14  out.open(VolumeName, ios::out | ios::binary);
15  if (!out) {
16      cout << "\t==> Error when create Volume" << VolumeName << endl;    return false;
17  }
18  isVerify = true;
19  out.write(VolumeName, sizeFileName);
20  setPassword(out);
21  out.write((char*)&numOfFiles, 4);
22  out.seekp(sizeVolume - 1, ios::beg);
23  int nul = 0x0;
24  out.write((char*)&nul, 1);
25  out.close();
26  cout << "\t==> " << VolumeName << " was created and formated successfully" << endl;
27  return true;
28 }
29

```

Chương trình sẽ cho người dùng nhập tên của Volume (VD: MyFS.dat), kích thước Volume tính theo MB và cuối cùng là nhập mật khẩu để truy cập Volume (enter nếu không muốn đặt).

Kết quả chương trình:

```

==> Choose your option: 2
+ Name for volume (with extensions): myFS.dat
+ Enter size of volume (in MB): 15
+ Input password (enter if not set): 123456
==> myFS.dat was created and formated successfully

```

3.2.3.2 Chức năng xác minh Volume cho các lần truy xuất tiếp theo

Nếu Volume đã được tạo từ trước đó, sau khi chương trình tắt đi mở lại, toàn bộ chức năng liên quan đến việc truy xuất thông tin Volume sẽ không hoạt động nếu như người dùng không xác minh volume thông qua việc nhập tên và mật khẩu của hệ thống tập tin.

Mã nguồn:

```

1  bool verify() {
2      char* name = new char[128];
3      char* pass = new char[32];
4      cout << "+ Enter your volume name: ";
5      cin.getline(name, sizeFileName);
6      ifstream in;
7      in.open(name, ios::in | ios::binary);
8      if (!in) {
9          cout << "\t==> " << name << " is not exists" << endl;
10         return false;
11     }
12     cout << "+ Enter your password's volume (enter if not set): ";
13     cin.getline(pass, 32);
14     int hashValue = hashPass(pass);
15     in.seekg(sizeFileName, ios::beg);
16     char* passReal = new char[4];
17     in.read(passReal, 4);
18     if (hashValue == *(int*)passReal) {
19         strcpy(VolumeName, name);

```



```

20     isVerify = true;
21     sizeVolume = filesize(VolumeName);
22     cout << "\t==> Verified, now you can fully access with " << VolumeName << endl;
23 } else
24     cout << "\t==> Wrong password" << endl;
25 return true;
26 }
27

```

Kết quả chạy chương trình:

```

==> Choose your option: 1
+ Enter your volume name: myFS.dat
+ Enter your password's volume (enter if not set): 123456
==> Verified, now you can fully access with myFS.dat

```

3.2.3.3 Chức năng thiết lập, thay đổi, kiểm tra mật khẩu truy xuất volume:

Tính năng thiết lập mật khẩu đi kèm với lúc tạo và định dạng volume để thuận tiện cho việc ghi vào volume. Tính năng kiểm tra mật khẩu được sử dụng trong chức năng xác minh Volume. Tính năng thay đổi mật khẩu chỉ được hoạt động khi người dùng đã xác minh Volume và nhập đúng mật khẩu trong chức năng này yêu cầu

Mã nguồn:

```

1 bool changePasswordFile()
2 {
3     if (!isVerify) {
4         cout << "\t==> Volume is not created or verified!" << endl;
5         return false;
6     }
7     int step = 0;
8     char* nameFile = new char[128];
9     cout << "+ Enter file name to be changed password: ";
10    cin.getline(nameFile, 128);
11    char* oldPass = new char[32];
12    char* newPass = new char[32];
13    char* repeatNewPass = new char[32];
14    cout << "+ Input old password: ";
15    cin.getline(oldPass, 32);
16    int OldPValue = hashPass(oldPass);
17    ifstream in;
18    in.open(VolumeName, ios::in | ios::binary);
19    in.seekg(132, ios::beg);
20    step += 132;
21    char* numFi = new char[4];
22    in.read(numFi, 4);
23    step += 4;
24    int numOfFile = *(int*)numFi;
25    bool flag = false;
26    char* file = new char[128];
27    char* temp = new char[12];
28    for (int i = 0; i < numOfFile; i++) {
29        in.read(file, 128);
30        step += 128;
31        in.read(numFi, 4);

```

```

32     step += 4;
33     if (_stricmp(file, nameFile) == 0) {
34         if (OldPValue != *(int*)numFi) {
35             cout << "\t==> Wrong password" << endl;
36             in.close();
37             return false;
38         }
39         flag = true;
40         break;
41     }
42     in.read(temp, 12);
43     step += 12;
44 }
45 if (flag == false) {
46     cout << "\t==> " << nameFile << " does not exists" << endl;
47     return false;
48 }
49 in.close();
50 cout << "+ Enter new password: ";
51 cin.getline(newPass, 32);
52 cout << "+ ReEnter password: ";
53 cin.getline(repeatNewPass, 32);
54 if (strcmp(newPass, repeatNewPass) != 0) {
55     cout << "\t==> No match" << endl;
56     return false;
57 }
58 ofstream out;
59 int newPassVal = hashPass(newPass);
60 out.open(VolumeName, ios::binary | ios::in | ios::out);
61 out.seekp(step - 4, ios::beg);
62 out.write((char*)&newPassVal, sizeof(int));
63 cout << "\t==> Password has changed" << endl;
64 return true;
65 }
66

```

Kết quả chạy chương trình:

```

==> Choose your option: 4
+ Input old password: 123456
+ Enter new password: 123789
+ Re-Enter password: 123789
    ==> Password has changed

```

```

==> Choose your option: 4
+ Input old password: 123789
+ Enter new password: 123456
+ Re-Enter password: abcdef
    ==> No match

```

```

==> Choose your option: 4
+ Input old password: 123abc
    ==> Wrong password!

```

3.2.3.4 Chức năng liệt kê thông tin volume và danh sách tập tin bên trong:

Các thông tin được liệt kê gồm: Trường tên volume, kích thước volume tính theo kB và Mb, số lượng tập tin nó đang chứa bên trong. Với mỗi tập tin, chương trình cũng hiển thị các thông tin như tên, kích thước, chỉ mục của sector đầu tiên trong phần Data Area và số sector được cấp phát để lưu dữ liệu của tập tin đó.

Mã nguồn:

```

1 bool infoVolume()
2 {
3     if (!isVerify) {
4         cout << "Volume is not created or verified!" << endl;
5         return false;
6     }
7     cout << "Volume name: " << VolumeName << endl;
8     cout << "Size: " << sizeVolume / 1024 / 1024 << " MB (" << sizeVolume << " bytes)" << endl;
9     ifstream in;
10    in.open(VolumeName, ios::in | ios::binary);
11    in.seekg(132, ios::beg);
12    char* data = new char[4];
13    in.read(data, 4);
14    int numOfFile = *(int*)data;
15    cout << "Number of file(s): " << numOfFile << endl;
16    for (int i = 0; i < numOfFile; i++) {
17        int number = 0;
18        char* name = new char[128];
19        in.read(name, 128);
20        cout << "\t File name: " << name << endl;
21        in.read(data, 4);
22        in.read(data, 4);
23        number = *(int*)data;
24        cout << "\t Size: " << number << " bytes" << endl;
25        in.read(data, 4);
26        number = *(int*)data;
27        cout << "\t Started sector: " << number << endl;
28        in.read(data, 4);
29        number = *(int*)data;
30        cout << "\t Sector allocated: " << number << endl;
31        cout << endl;
32    }
33    return true;
34 }
35

```

Kết quả chạy chương trình:

```

==> Choose your option: 3
Volume name: myFS.dat
Size: 15 MB (15728640 bytes)
Number of file(s): 3
    File name: lect06-CSP.pdf
    Size: 1076854 bytes
    Started sector: 10
    Sector allocated: 2104

    File name: a.pdf
    Size: 558080 bytes
    Started sector: 2114
    Sector allocated: 1090

    File name: Snake(team3).exe
    Size: 193024 bytes
    Started sector: 3204
    Sector allocated: 377

```

3.2.3.5 Chức năng sao chép tập tin từ bên ngoài vào trong Volume::

Hệ thống sẽ yêu cầu người dùng nhập tên của tập tin cần sao chép bao gồm cả đường dẫn đến tập tin đó, nếu tồn tại, hệ thống sẽ yêu cầu người dùng nhập mật khẩu để truy xuất tập tin. Sau đó tập tin đã được sao chép vào Volume

Mã nguồn:

```
1 bool importFile() {
2     if (!isVerify) {
3         cout << "\t==> Volume is not created or verified!" << endl;
4         return false;
5     }
6     char* fileImport = new char[256];
7     cout << "+ Enter file name (with path Ex:\"D:\\folder\\filename.pdf\"): ";
8     cin.getline(fileImport, 256);
9     string tempStr(fileImport);
10    tempStr = tempStr.substr(tempStr.find_last_of('\\') + 1);
11    const char* fileName = tempStr.c_str();
12
13    ifstream in;
14    in.open(fileImport, ios::in | ios::binary);
15    if (!in) {
16        cout << "\t==> " << fileImport << " does not exist" << endl;
17        return false;
18    }
19    int size = filesize(fileImport);
20    char* dataFile = new char[size];
21    in.read(dataFile, size);
22    in.close();
23
24    in.open(VolumeName, ios::in | ios::binary);
25    in.seekg(sizeFileName + 4);
26    char* num = new char[4];
27    in.read(num, 4);
28    int numOfFiles = *(int*)num;
29    for (int i = 0; i < numOfFiles; i++) {
30        char* name = new char[128];
31        in.read(name, 128);
32        if (_stricmp(name, fileName) == 0) {
33            cout << "\t==> File existed" << endl;
34            return false;
35        }
36        in.read(name, 16);
37    }
38    int lastBlock = 0;
39    int numOfBlock = 0;
40    if (numOfFiles == 0) {
41        lastBlock = 10;
42        numOfBlock = 0;
43    }
44    else {
45        int step = 136 + numOfFiles * 144 - 8;
46        in.seekg(step, ios::beg);
47        in.read(num, 4);
48        lastBlock = *(int*)num;
```

```

49     in.read(num, 4);
50     numOfBlock = *(int*)num;
51 }
52 in.close();
53 int startNewBlock = lastBlock + numOfBlock;
54 int numOfNewBlock = ceil((double)size / sizeSector);
55 if (sizeSector * (startNewBlock + numOfNewBlock) > sizeVolume) {
56     cout << "\t==> File's size is more than Volume's size" << endl;
57     return false;
58 }
59 ofstream out;
60 out.open(VolumeName, ios::binary | ios::in | ios::out);
61 out.seekp(sizeFileName + 4, ios::beg);
62 numOfFiles++;
63 out.write((char*)&numOfFiles, 4);
64 int step = 136 + (numOfFiles - 1) * 144;
65 out.seekp(step, ios::beg);
66 out.write(fileName, 128);
67 setPassword(out);
68 out.write((char*)&size, 4);
69 out.write((char*)&startNewBlock, 4);
70 out.write((char*)&numOfNewBlock, 4);
71 out.seekp(sizeSector * startNewBlock, ios::beg);
72 out.write(dataFile, size);
73 out.seekp(sizeSector * (startNewBlock + numOfNewBlock) - 1, ios::beg);
74 int nul = 0x0;
75 out.write((char*)&nul, 1);
76 out.close();
77 cout << "\t==> Import file " << fileName << " successfully" << endl;
78 return true;
79 }
80

```

Kết quả chạy chương trình:

```

==> Choose your option: 5
+ Enter file name (with path Ex:"D:\folder\filename.pdf"): D:\OS\FileSystem\FileSystem\document.pdf
+ Input password (enter if not set): password
==> Import file document.pdf successfully

```

```

==> Choose your option: 5
+ Enter file name (with path Ex:"D:\folder\filename.pdf"): C:\123.txt
==> C:\123.txt does not exist

```

```

==> Choose your option: 5
+ Enter file name (with path Ex:"D:\folder\filename.pdf"): D:\CNXHKKH.pdf
==> File's size is more than Volume's size

```

3.2.3.6 Chức năng đặt, đổi mật khẩu cho tập tin trong volume:

Với mỗi tập tin được thêm vào hệ thống đều sẽ được yêu cầu đặt mật khẩu. Chức năng thay đổi mật khẩu sẽ được đưa ra làm chức năng riêng, nó chỉ hoạt động nếu người dùng xác minh volume và nhập đúng mật khẩu khi chức năng yêu cầu

Mã nguồn:

```

1 bool changePasswordFile()
2 {
3     if (!isVerify) {
4         cout << "\t==> Volume is not created or verified!" << endl;

```

```

5     return false;
6 }
7 int step = 0;
8 char* nameFile = new char[128];
9 cout << "+ Enter file name to be changed password: ";
10 cin.getline(nameFile, 128);
11 char* oldPass = new char[32];
12 char* newPass = new char[32];
13 char* repeatNewPass = new char[32];
14 cout << "+ Input old password: ";
15 cin.getline(oldPass, 32);
16 int OldPValue = hashPass(oldPass);
17 ifstream in;
18 in.open(VolumeName, ios::in | ios::binary);
19 in.seekg(132, ios::beg);
20 step += 132;
21 char* numFi = new char[4];
22 in.read(numFi, 4);
23 step += 4;
24 int numOfFile = *(int*)numFi;
25 bool flag = false;
26 char* file = new char[128];
27 char* temp = new char[12];
28 for (int i = 0; i < numOfFile; i++) {
29     in.read(file, 128);
30     step += 128;
31     in.read(numFi, 4);
32     step += 4;
33     if (_stricmp(file, nameFile) == 0) {
34         if (OldPValue != *(int*)numFi) {
35             cout << "\t==> Wrong password" << endl;
36             in.close();
37             return false;
38         }
39         flag = true;
40         break;
41     }
42     in.read(temp, 12);
43     step += 12;
44 }
45 if (flag == false) {
46     cout << "\t==> " << nameFile << " does not exists" << endl;
47     return false;
48 }
49 in.close();
50 cout << "+ Enter new password: ";
51 cin.getline(newPass, 32);
52 cout << "+ ReEnter password: ";
53 cin.getline(repeatNewPass, 32);
54 if (strcmp(newPass, repeatNewPass) != 0) {
55     cout << "\t==> No match" << endl;
56     return false;
57 }
58 ofstream out;
59 int newPassVal = hashPass(newPass);

```

```

60 out.open(VolumeName, ios::binary | ios::in | ios::out);
61 out.seekp(step - 4, ios::beg);
62 out.write((char*)&newPassVal, sizeof(int));
63 cout << "\t==> Password has changed" << endl;
64 return true;
65 }
66

```

Kết quả chạy chương trình:

```

==> Choose your option: 7
+ Enter file name to be changed password: a.pdf
+ Input old password: 2
+ Enter new password: 123456
+ ReEnter password: 123456
  ==> Password has changed

```

```

==> Choose your option: 7
+ Enter file name to be changed password: c.pdf
+ Input old password: 123
  ==> c.pdf does not exists

```

```

==> Choose your option: 7
+ Enter file name to be changed password: a.pdf
+ Input old password: 123456
+ Enter new password: 123789
+ ReEnter password: abcdef
  ==> No match

```

3.2.3.7 Chức năng chép một tập tin trong Volume ra ngoài:

Chương trình yêu cầu người dùng nhập tên tập tin, yêu cầu người dùng nhập mật khẩu. Nếu tập tin không tồn tại trong volume hoặc mật khẩu nhập sai, chương trình sẽ không thực hiện việc chép file, ngược lại file sẽ được sao chép ra ngoài.

Mã nguồn:

```

1 bool exportFile() {
2     if (!isVerify) {
3         cout << "\t==> Volume is not created or verified!" << endl;
4         return false;
5     }
6     ifstream in;
7     in.open(VolumeName, ios::in | ios::binary);
8     char* data = new char[128];
9     char* num = new char[4];
10    in.seekg(132, ios::beg);
11    in.read(num, 4);
12    int numOfFile = *(int*)num;
13    bool check = false;
14    int size = 0;
15    int blockStart = 0;
16    int numOfBlock = 0;
17
18    char* fileName = new char[128];
19    cout << "+ Enter file name: ";
20    cin.getline(fileName, 128);
21    for (int i = 0; i < numOfFile; i++) {
22        in.read(data, 128);
23        if (_stricmp(data, fileName) == 0) {
24            char* pass = new char[32];
25            cout << "+ Enter password: ";
26            cin.getline(pass, 32);
27            in.read(num, 4);
28            int passVal = *(int*)num;
29            if (hashPass(pass) != passVal) {

```

```

30     cout << "\t==> Wrong password" << endl;
31     return false;
32 }
33 in.read(num, 4);
34 size = *(int*)num;
35 in.read(num, 4);
36 blockStart = *(int*)num;
37 in.read(num, 4);
38 numOfBlock = *(int*)num;
39 check = true;
40 break;
41 }
42 else {
43     in.read(data, 16);
44 }
45 }
46 if (check == false) {
47     cout << "\t==> " << fileName << " is not exist" << endl;
48     return false;
49 }
50 in.seekg(blockStart * sizeSector, ios::beg);
51 char* dataFile = new char[size];
52 in.read(dataFile, size);
53 in.close();
54
55 ofstream out;
56 out.open(fileName, ios::out | ios::binary);
57 out.write(dataFile, size);
58 out.close();
59 cout << "\t==> " << fileName << " was exported successfully" << endl;
60 return true;
61 }
62

```

Kết quả chạy chương trình:

```

==> Choose your option: 6
+ Enter file name: Snake(team3).exe
+ Enter password: 3
    ==> Snake(team3).exe was exported successfully

```

```

==> Choose your option: 6
+ Enter file name: 123.exe
    ==> 123.exe is not exist

```

```

==> Choose your option: 6
+ Enter file name: a.pdf
+ Enter password: 123456789
    ==> Wrong password

```

3.2.3.8 Xóa 1 tập tin trong Volume:

Hệ thống yêu cầu người dùng nhập tên tập tin cần xóa, nhập mật khẩu. Nếu cả 2 chính xác, chương trình sẽ tiến hành xóa tập tin, và tiến hành dồn sector để tối ưu không gian lưu trữ cho Volume

Mã nguồn:

```

1 bool deleteFile()
2 {
3     if (!isVerify) {
4         cout << "\t==> Volume is not created or verified!" << endl;
5         return false;
6     }

```



```

7  char* fileNameDelete = new char[128];
8  cout << "+ Enter file name to be deleted: ";
9  cin.getline(fileNameDelete, 128);
10 fstream handle;
11 handle.open(VolumeName, ios::binary | ios::in | ios::out);
12 handle.seekg(132);
13 char* num = new char[4];
14 handle.read(num, 4);
15 int numFiles = *(int*)num;
16 char* nameFile = new char[128];
17 int seekgPosition = 136;
18 int blockFileDelete = 0;
19 int numBlockDelete = 0;
20 bool flag = false;
21 for (int i = 1; i <= numFiles; i++) {
22     handle.read(nameFile, 128);
23     if (_stricmp(nameFile, fileNameDelete) == 0) {
24         flag = true;
25         char* pass = new char[32];
26         cout << "+ Enter password of file: ";
27         cin.getline(pass, 32);
28         handle.read(num, 4);
29         if (hashPass(pass) != *(int*)num) {
30             cout << "\t==> Wrong password" << endl;
31             return false;
32         }
33         handle.read(nameFile, 4); //ignore 4 next byte;
34         handle.read(num, 4);
35         blockFileDelete = *(int*)num;
36         handle.read(num, 4);
37         numBlockDelete = *(int*)num;
38         numFiles--;
39         handle.seekp(132, ios::beg);
40         handle.write((char*)&numFiles, 4);
41         int sizeOverwrite = 10 * 512 - 136 - 144 * i;
42         char* overrideData = new char[sizeOverwrite];
43         handle.seekg(seekgPosition + 144, ios::beg);
44         handle.read(overrideData, sizeOverwrite);
45         handle.seekp(seekgPosition, ios::beg);
46         handle.write(overrideData, sizeOverwrite);
47         if (i == numFiles + 1) {
48             cout << "\t==> Delete " << fileNameDelete << " successfully" << endl;
49             break;
50         }
51         for (int j = 0; j < numFiles + 1 - i; j++){
52             seekgPosition += 136;
53             handle.seekg(seekgPosition, ios::beg);
54             handle.read(num, 4);
55             int temp = *(int*)num - numBlockDelete;
56             handle.seekp(seekgPosition, ios::beg);
57             handle.write((char*)&temp, 4);
58             seekgPosition += 8;
59
60             handle.read(num, 4);
61             int temp2 = *(int*)num;

```

```

62
63     handle.seekg((temp + numBlockDelete) * sizeSector, ios::beg);
64     char* dataMove = new char[sizeSector * temp2];
65     handle.read(dataMove, sizeSector * temp2);
66     handle.seekp(temp * sizeSector, ios::beg);
67     handle.write(dataMove, sizeSector * temp2);
68 }
69 cout << "\t==> Delete " << fileNameDelete << " successfully" << endl;
70 break;
71 }
72 else {
73     seekgPosition += 144;
74     handle.read(nameFile, 16);
75 }
76 }
77 if (flag == false) {
78     cout << "\t==> " << fileNameDelete << " does not exist" << endl;
79 }
80 return true;
81 }
82

```

Kết quả chạy chương trình:

```

==> Choose your option: 8
+ Enter file name to be deleted: Snake(team3).exe
+ Enter password of file: 3
    ==> Delete Snake(team3).exe successfully

```

```

==> Choose your option: 8
+ Enter file name to be deleted: a.cpp
+ Enter password of file: 456
    ==> Wrong password

```

```

==> Choose your option: 8
+ Enter file name to be deleted: file.exe
    ==> file.exe does not exist

```

Tài liệu

- [1] Tham khảo cách đọc và ghi sector trong C/C++:
 - <http://www.cplusplus.com/forum/windows/18019/>
 - <https://stackoverflow.com/questions/29219674/read-specific-sector-on-hard-drive-using-c-language-on-windows>
- [2] Cách chuyển đổi cơ số: <https://www.geeksforgeeks.org/convert-decimal-to-hexa-decimal-including-negative-numbers/>
- [3] Tham khảo cho việc thiết kế mô hình cho hệ thống tập tin: <https://drive.google.com/file/d/1ynsToAL2IxmoLfsU-9kMHJurmXlTv0V2/view?usp=sharing> (Slide thầy Thái Hùng Văn)