PROJECT 1

QUẢN LÝ HỆ THỐNG TẬP TIN TRÊN WINDOWS

Lớp: 20\_1

Môn: Hệ điều hành

GVHD: ThS. Lê Viết Long

TP. HỒ CHÍ MINH, THÁNG NĂM 20

Mục lục

Giới thiệu

**Thông tin nhóm**..……………………………………………………1

**Phân công công việc**..……………………………………………….2

**Mức độ hoàn thành đồ án**.………………………………………….3

**Hệ thống tập tin trên windows**.…………………………………….4

**Các ý tưởng giải quyết**

* **FAT32** ..………………………………………………………6
* **NTFS** ..………………………………………………………..7

**Xây dựng và tổ chức code**

* **FAT32** ….....………………………………………………...11
* **NTFS** .....…………………………………………………….14

**Tham khảo** …………………………………………………………17

Giới thiệu

1. **Tổng quan:**

Xây dựng chương trình máy tính (sử dụng ngôn ngữ tùy ý) nhằm đọc các thông tin trên phân vùng FAT32 và NTFS.

1. **Nhiệm vụ đề tài:**

* **Đọc các thông tin chi tiết của một phân vùng**

**\_ FAT32, NTFS**

* **Hiển thị thông tin cây thư mục của phân vùng**

\_ Chương trình hiển thị cây thư mục gốc gồm tên tập tin / thư mục, trạng thái, kích thước (nếu có), chỉ số sector lưu trữ trên đĩa cứng

\_ Khi truy xuất thông tin trên cây thư mục: chương trình hiển thị thông tin là nội dung tập tin đối với tập tin có phần mở rộng là txt, các loại tập tin khác hiển thị thông báo dùng phần mềm tương thích để đọc nội dung. Trường hợp đối tượng là thư mục, chương trình cho phép hiển thị cây thư mục con (thông tin hiển thị tương tự như cây thư mục gốc)

Thông tin nhóm

|  |  |  |
| --- | --- | --- |
| **MSSV** | **Họ tên** | **Email** |
| 20120524 | Lương Trọng Khôi | 20120524@student.hcmus.edu.vn |
| 20120392 | Phạm Thụy Bích Truyền | 20120392@student.hcmus.edu.vn |
| 20120427 | Lê Nhựt Anh | 20120427@student.hcmus.edu.vn |
| 20120429 | Nguyễn Quốc Anh | 20120429@student.hcmus.edu.vn |
| 20120603 | Trần Minh Trí | 20120603@student.hcmus.edu.vn |

Bảng phân công công việc

|  |  |  |  |
| --- | --- | --- | --- |
| **MSSV** | **Họ và tên** | **Công việc** | **Tiến độ** |
| 20120514 | Lương Trọng Khôi | **Đọc thông tin chi tiết của một phân vùng:**  + Đọc Boot Sector (FAT32).  + Đọc Partition Boot Sector (NTFS). | **100%** |
| 20120603 | Trần Minh Trí | **Đọc và hiển thị cây thư mục gốc - RDET (FAT32, NTFS):**  + Tên tập tin/Thư mục.  + Trạng thái.  + Kích thước (nếu có). | **100%** |
| 20120427 | Phạm Thụy Bích Truyền | **Truy xuất thông tin trên cây thư mục (FAT32):**  + Đối với tập tin: đọc được tập tin .txt, các tập tin khác thông báo phần mềm tương thích để đọc.  + Đối với thư mục: cho phép hiển thị giống cây thư mục gốc. | **100%** |
| 20120392 | Lê Nhựt Anh | **Truy xuất thông tin trên cây thư mục (NTFS):**  + Đối với tập tin: đọc được tập tin .txt, các tập tin khác thông báo phần mềm tương thích để đọc.  + Đối với thư mục: cho phép hiển thị giống cây thư mục gốc. | **100%** |
| 20120429 | Nguyễn Quốc Anh | **Tổng hợp nội dung:**  + Tổng hợp lại các file code, fix bug.  + Viết menu giao diện console.  Viết báo cáo. | **100%** |

Mức độ hoàn thành đồ án

|  |  |  |
| --- | --- | --- |
| STT | Tên công việc | Mức độ hoàn thành |
| 1 | Đọc các thông tin chi tiết của phân vùng FAT32 | 100% |
| 2 | Đọc các thông tin chi tiết của phân vùng NTFS | 100% |
| 3 | Hiển thị cây thư mục phân vùng FAT32 | 100% |
| 4 | Hiển thị cây thư mục phân vùng NTFS | 100% |
| Toàn bộ project: 100% | | |

Hệ thống tập tin trên Windows

* **Các hệ thống tập tin của Windows**
* Windows sử dụng các hệ thống tập tin FAT, [NTFS](https://vi.wikipedia.org/wiki/NTFS), [exFAT](https://vi.wikipedia.org/wiki/ExFAT), Live File System và ReFS (những thứ cuối cùng chỉ được hỗ trợ và sử dụng được trong [Windows Server 2012](https://vi.wikipedia.org/wiki/Windows_Server_2012), [Windows Server 2016](https://vi.wikipedia.org/wiki/Windows_Server_2016), [Windows 8](https://vi.wikipedia.org/wiki/Windows_8), [Windows 8.1](https://vi.wikipedia.org/wiki/Windows_8.1) và [Windows 10](https://vi.wikipedia.org/wiki/Windows_10); Windows không thể khởi động từ nó).
* Các hệ thống file trong windows gồm 2 hệ thống chính là FAT và NTFS, trong đó NTFS là hệ thống file với nhiều đặc tính hiện đại mà hệ thống FAT không có.Sau đây chúng ta sẽ tìm hiểu về 2 hệ thống file này.  
  Khái niệm về FAT và NTFS
* **FAT**
* **FAT32**

\_ Được giới thiệu trong phiên bản Windows 95 Service Pack 2 (OSR 2), được xem là phiên bản mở rộng của FAT16. Do sử dụng không gian địa chỉ 32 bit nên FAT32 hỗ trợ nhiều cluster trên một partition hơn, do vậy không gian đĩa cứng được tận dụng nhiều hơn. Ngoài ra với khả năng hỗ trợ kích thước của phân vùng từ 2GB lên 2TB và chiều dài tối đa của tên tập tin được mở rộng đến 255 ký tự đã làm cho FAT16 nhanh chóng bị lãng quên. Tuy nhiên, nhược điểm của FAT32 là tính bảo mật và khả năng chịu lỗi (Fault Tolerance) không cao.

* **NTFS (New Technology File System)**

\_ Được giới thiệu cùng với phiên bản Windows NT đầu tiên (phiên bản này cũng hỗ trợ FAT32). Với không gian địa chỉ 64 bit, khả năng thay đổi kích thước của cluster độc lập với dung lượng đĩa cứng, NTFS hầu như đã loại trừ được những hạn chế về số cluster, kích thước tối đa của tập tin trên một phân vùng đĩa cứng.  
  
\_ NTFS sử dụng bảng quản lý tập tin MFT (Master File Table) thay cho bảng FAT quen thuộc nhằm tăng cường khả năng lưu trữ, tính bảo mật cho tập tin và thư mục, khả năng mã hóa dữ liệu đến từng tập tin. Ngoài ra, NTFS có khả năng chịu lỗi cao, cho phép người dùng đóng một ứng dụng “chết” (not responding) mà không làm ảnh hưởng đến những ứng dụng khác. Tuy nhiên, NTFS lại không thích hợp với những ổ đĩa có dung lượng thấp (dưới 400 MB) và không sử dụng được trên đĩa mềm.

* **So sánh giữa FAT32 và NTFS**

|  |  |
| --- | --- |
| FAT32 là định dạng file system khá cũ | NTFS là hệ thống file hiện đại |
| FAT32 không hỗ trợ các tính năng bảo mật như phần quyền quản lý, mã hoá | NTFS có hỗ trợ các tính năng bảo mật như phần quyền quản lý, mã hoá |
| FAT32 có khả năng phục hồi và chịu lỗi rất kém | NTFS là hệ thống file có khả năng ghi lại được các hoạt động mà hệ điều hành đã và đang thao tác trên dữ liệu |
| Khi mà mất điện đột ngột thì Windows 98, 2000, XP… đều phải quét lại đĩa khi khởi động lại nếu đĩa đó được format bằng chuẩn FAT32 | Trong khi format đĩa cứng bằng NTFS thì lại hoàn toàn không cần quét đĩa lại |

\_ Tuy thế, FAT32 vẫn còn tỏ ra hữu dụng trên các máy tính cấu hình quá yếu ớt, chỉ có thể chạy được Windows 98. FAT16 và FAT32 vẫn được dùng để định dạng cho các loại thẻ nhớ, vì các thiết bị chấp nhận thẻ nhớ như máy ảnh số, máy nghe nhạc vẫn chưa thấy loại nào tương thích với NTFS cả. FAT16 luôn là lựa chọn hàng đầu khi bạn muốn copy dữ liệu của mình từ một máy tính chạy Windows sang máy chạy hệ điều hành khác như Mac chẳng hạn. Hầu hết các máy Mac hiện nay đều không thể nhận dạng các thẻ nhớ USB được định dạng bằng FAT 32.

**Các ý tưởng giải quyết**

1. Ngôn ngữ lập trình và IDE:

* Ngôn ngữ lập trình: C++.
* Phần mềm thực hiện: Visual Studio, Visual Studio Code.
* Phần mềm hỗ trợ: Disk Editor

1. Đọc thông tin chi tiết của một phân vùng

* Đọc Boot Sector (FAT32).

\_ Đối với phân vùng FAT32 đọc các thông tin được mô tả trong Boot Sector.

|  |  |  |
| --- | --- | --- |
| Offset | Số byte | Nội dung |
| 0 | 3 | Jump\_Code: lệnh nhảy qua vùng thông số |
| 3 | 8 | OEM\_ID: nơi sản xuất-version, thường là “MSWIN4.1” |
| B | 2 | Số byte trên Sector, thường là 512 |
| D | 1 | Sc: số sector trên cluster |
| E | 2 | Sb: số sector thuộc vùng Boostsector |
| 10 | 1 | Nf: số bảng FAT, thường là 2 |
| 11 | 2 | Không dùng, thường là 0 (số entry của RDET-với FAT) |
| 13 | 2 | Không dùng, thường là 0 (số sector của vol-với FAT) |
| 15 | 1 | Loại thiết bị(F8h nếu là đĩa cứng) |
| 16 | 2 | Không dùng, thường là 0 (số sector của bảng FAT-với FAT) |
| 18 | 2 | Số sector của track |
| 1A | 2 | Số lượng đầu đọc |
| 1C | 4 | Khoảng cách từ nơi mô tả vol đến đầu vol |
| 20 | 4 | Sv: Kích thước volume |
| 24 | 4 | Sf: Kích thước mỗi bảng FAT |
| 28 | 2 | Bit 8 bật: chỉ ghi vào bảng FAT active(có chỉ số là 4 bit đầu) |
| 2A | 2 | Version của FAT32 trên vol này |
| 2C | 4 | Cluster bắt đầu của RDET |
| 30 | 2 | Sector chứa thông tin phụ(về cluster trống), thường là 1 |
| 32 | 2 | Sector chứa bản lưu của Boot Sector |
| 34 | C | Dành riêng(cho các phiên bản sau) |
| 40 | 1 | Kí hiệu vật lý của đĩa chứ vol(0: mềm, 80h cứng) |
| 41 | 1 | Dành riêng |
| 42 | 1 | Kí hiệu nhận diện HĐH |
| 43 | 4 | SerialNumber của Volume |
| 47 | B | Volume Label |
| 52 | 8 | Loại FAT, là chuỗi “FAT32” |
| 5A | 1A4 | Đoạn chương trình khởi tạo & nạp HĐH khi khởi động máy |
| 1FE | 2 | Dấu hiệu kết thúc BootSector/Master Boot(luôn là AA55h) |

\_ USB Drive (E): Đây sẽ là ổ đĩa để thực hiện demo và đánh giá kết quả cho đồ án 1 này. Ổ đĩa đã được Format thành định dạng FAT32, đây là một số thông tin cơ bản của ổ đĩa E này:

Graphical user interface, text, application, email

Description automatically generated

\_ Boot Sector của FAT32 chứ rất nhiều trường dữ liệu. Tuy nhiên, các thông số quan trọng sẽ được sử dụng lại nhiều lần trong đồ án là:

\_ Số byte trên Sector

\_ Số sector trên cluster (SC)

\_ Số sector thuộc vùng Boot Sector (SB)

\_ Số bảng FAT (NF)

\_ Số entry của RDET

\_ Kích thước volume (SV)

\_ Kích thước mỗi bảng FAT (SF)

\_ Cluster bắt đầu của RDET

* Mô tả BPB của NTFS.

|  |  |  |  |
| --- | --- | --- | --- |
| **Offset** | **Số byte** | | **Nội dung** |
| 0Bh | 2 | Kích thước một sector | |
| 0Dh | 1 | Số sector trong một cluster | |
| 0Eh | 2 | Chưa sử dụng | |
| 10h | 1 | Với hệ thống NTFS luôn mang giá trị 0 | |
| 11h | 2 | Với hệ thống NTFS luôn mang giá trị 0 | |
| 13h | 2 | Luôn mang giá trị 0, hệ thống NTFS không sử dụng tới trường hợp này. | |
| 15h | 1 | Mã xác định loại đĩa. | |
| 16h | 2 | Với hệ thống NTFS luôn mang giá trị 0 | |
| 18h | 2 | Số sector/track | |
| 1Ah | 2 | Số mặt đĩa (head hay side) | |
| 1Ch | 4 | Sector bắt đầu của ổ đĩa login | |
| 20h | 4 | Luôn mang giá trị 0, hệ thống NTFS không sử dụng tới trường hợp này | |
| 24h | 4 | Hệ thống NTFS luôn thiết lập giá trị này là “80008000” | |
| 28h | 8 | Số sector của ổ đĩa logic | |
| 30h | 8 | Cluster bắt đầu của MFT | |
| 38h | 8 | Cluster bắt đầu của MFT dự phòng (MFTMirror) | |
| 40h | 1 | Kích thước của một bản ghi trong MFT (MFT entry), đơn vị tính là byte | |
| 41h | 3 | Luôn mang giá trị 0, hệ thống NTFS không sử dụng tới trường này | |
| 44h | 1 | Số cluster của Index Buffer | |
| 45h | 3 | Luôn mang giá trị 0, hệ thống NTFS không sử dụng tới trường này. | |
| 48h | 8 | Số seri của ổ đĩa (volume serial number) | |
| 50h | 4 | Không được sử dụng bởi NTFS | |

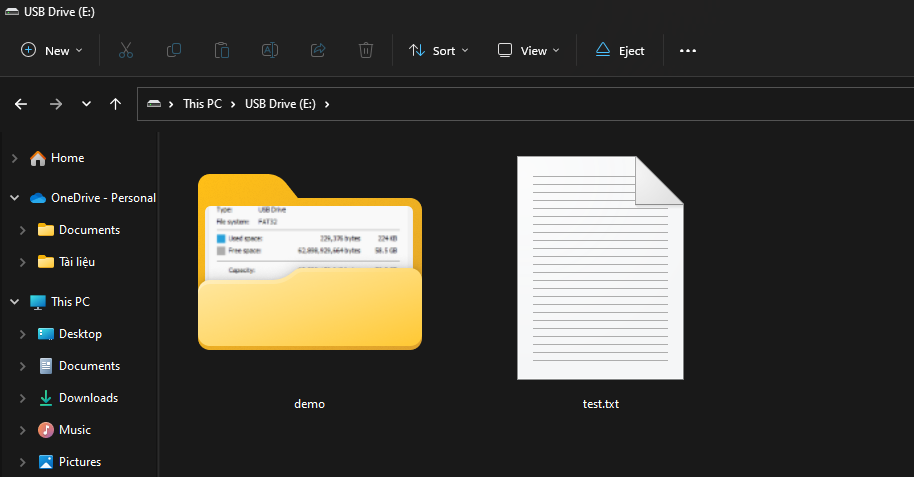
\_ USB Drive (E): Đây sẽ là ổ đĩa để thực hiện demo và đánh giá kết quả cho đồ án 1 này. Ổ đĩa đã được Format thành định dạng NTFS, đây là một số thông tin cơ bản của ổ đĩa E này:

Graphical user interface, text, application, email

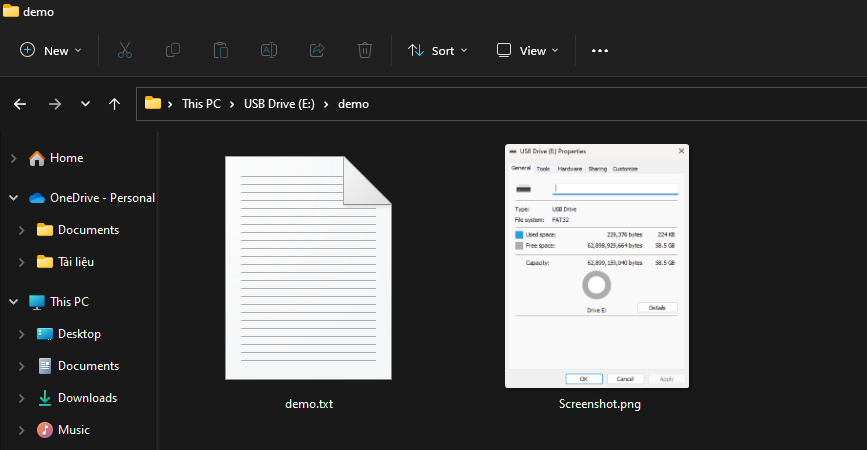
Description automatically generated

1. **Hiển thị cây thư mục phân vùng**

* Trong ổ đĩa E có các thư mục và file được tổ chức như sau:



*Hình 1:*Thư mục và file trong ổ E



*Hình 2:* file trong thư mục demo

Xây dựng và tổ chức code

1. Đọc thông tin và hiển thị cây thư mục phân vùng FAT32
2. Đọc thông tin phân vùng FAT32

* Để đọc thông tin Boot Sector thuận tiện dễ dàng hiểu biết để code thì xây dựng một cấu trúc FAT32 có các trường dữ liệu là các thông tin quan trọng cần chú ý đến đã được nêu ở trên thì ta xây được cấu trúc sau đây:

//Cấu trúc BootSector

struct BOOTSECTORFAT32

{

BYTE JUMP[3];

BYTE OEM[8];

WORD BytePerSector; // Số byte của một Sector (B - 2)

BYTE SectorPerCluster; // Số Sector của một Cluster (D - 1) SC

WORD ReservedSector; // Số Sector của BootSector (E - 2) SB

BYTE FatNum; // Số bảng FAT (10 - 1) NF

WORD EntryRDET; // Số Entry của RDET (11 - 2) SRDET

WORD LowNumberSectors;

BYTE DeviceType;

WORD SectorPerFat16;

WORD SectorPerTrack; // Số Sector của một Track (18 - 2)

WORD HeadPerDisk;

DWORD NumberHiddenSectors;

DWORD HighNumberSectors;

DWORD SectorPerFat32; // Số Sector của một bảng FAT (24 - 4) Sf

WORD Bit8Flag;

WORD FAT32Ver;

DWORD FirstRDETCluster; // Cluster bắt đầu của RDET (2C - 4)

WORD AddiInfoSector;

WORD BackupSector;

BYTE LaterVerReserved[12];

BYTE PhysicDisk;

BYTE Reserved;

BYTE Signature;

DWORD VolumeSerial; //Số sector trong Volume

BYTE VolumeLabel[11];

BYTE FATID[8];

BYTE BootProgram[420];

WORD EndSignature;

};

* Khi xây dựng được cấu trúc và hàm đọc ngược thì bước tiếp theo là đọc một sector trên phân vùng, ta sẽ dùng hàm int ReadBootSectorFAT32(LPCWSTR drive, int readPoint, BYTE sector[512]) được thầy cung cấp trước. Hàm này đọc trên một phân vùng drive (ổ đĩa cần đọc) một sector(512 Byte) từ vị trí readPoint.
* Tiếp theo dùng hàm void covertSectorToBS32(BYTE sector[512], BOOTSECTORFAT32& bs32), để lấy thông tin đã được ghi lại từ sector và truyền vào biến bs32 có kiểu dữ liệu là BOOTSECTORFAT32 để lưu trữ thông tin cần thiết.
* Ta dùng thêm 2 hàm để chuyển kiểu dữ liệu của các dữ liệu trong cấu trúc BOOTSECTORFAT32 thành unsigned int để xuất ra màn hình.

unsigned int reversedWORD(WORD buffer) {

unsigned int result = 0;

BYTE arr[2];

memcpy(arr, &buffer, 2);

result = (result << 8) | arr[1];

result = (result << 8) | arr[0];

return result;

}

unsigned int reversedDWORD(DWORD buffer) {

unsigned int result = 0;

BYTE arr[4];

memcpy(arr, &buffer, 4);

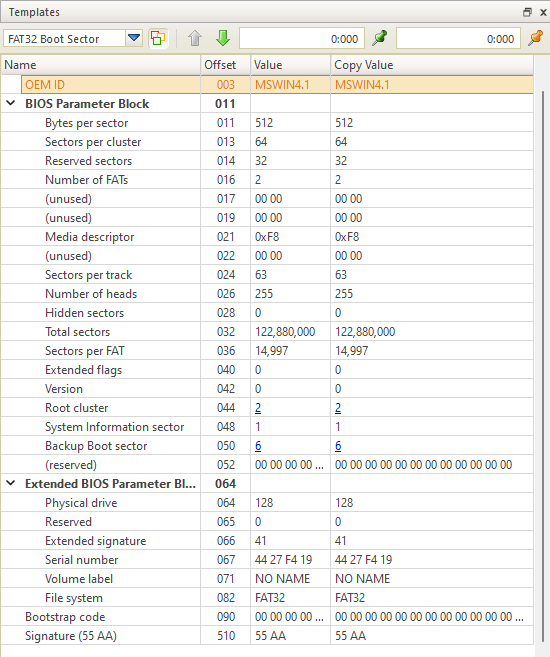
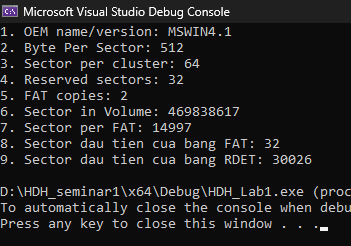
for (int i = 3; i >= 0; i--)

result = (result << 8) | arr[i];

return result;

}

* Để xuất ra thông tin cũng như dữ liệu mà ta cần biết của một phân vùng thì ta xây dựng hàm void InforBootSector(BOOTSECTORFAT32 bs32). Hàm này được truyền vào biến bs32 có kiểu dữ liệu BootSector\_FAT32\* đã được mang thông tin hay chứa dữ liệu đã được đọc. Sau đó việc cần làm chỉ là xuất ra thông tin quan trọng mà bs32 đang chứa.
* Để kết thúc ta dùng hàm main để thực thi các hàm bên trên:
* **Kết quả**: Đối chiếu với kết quả khi dùng phần mềm Disk Editor cho phân vùng FAT32 của ổ đĩa E thì ta thấy kết quả chính xác.



1. Hiển thị cây thư mục phân vùng FAT32

* Đầu tiên ta tạo một cấu trúc tên là DIRECTORY để lưu trữ các thông tin của thư mục, tập tin như tên thư mục/tập tin, thuộc tính, Cluster bắt đầu, kích thước của một sector, và con trỏ dùng để trỏ đến thư mục/tập tin tiếp theo, và một con trỏ khác dùng để trỏ đến thư mục con.

struct DIRECTORY {

char Name[256]; // Tên thư mục/ tập tin

int Attr; // Thuộc tính (thường là thư mục/ tập tin)

int StartCluster; // Cluster bắt đầu

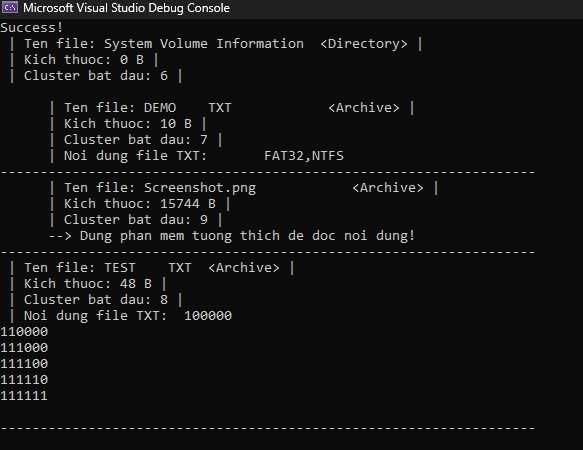
int FileSize; // Kích cỡ (tính theo byte)

DIRECTORY\* next; // Trỏ đến thư mục/ tập tin tiếp theo

DIRECTORY\* dir; // Trỏ đến thư mục con

};

* Dùng hàm void initFAT(int\*& FAT, BOOTSECTORFAT32 fat32, LPCWSTR drive1) để phân tích bảng FAT, tìm và đọc từ ổ đĩa nhờ hàm seekg( ) và read( ) bắt đầu từ chỉ số Sector của BootSector \* Kích thước của một sector, và truyền vào mảng động a được khai báo với số lượng phần tử là kích cỡ theo byte của số sector của bảng FAT, sau đó gán lại cho tham số đầu vào fat32.
* Ta cần biết được sector đầu của RDET để làm. Ta có thể kiếm được sector đầu tiên của RDET bằng các thông tin kiếm được sau khi đã đọc BootSector ở trên là: 30026. Sau khi đã có được sector đầu tiên của RDET điều ta cần làm là xây dựng hàm để có thể đọc được thông tin ở vị trí sector 30026.
* Ta đọc cây thư mục bằng hàm DIRECTORY\* readDirectory(int firstEntryIndex, int clusIndex, int\* FAT, BOOTSECTORFAT32 fat32, LPCWSTR drive1, string space). Khai báo các biến cần thiết như số entry, kích thước của cluster với kích thước của mảng các phần tử cần được đọc vào từ mảng động ở hàm void initFAT(int\*& FAT, BOOTSECTORFAT32 fat32, LPCWSTR drive1). Sau đó ta sẽ tìm kiếm từ sector đầu tiên của tham số clusIndex truyền vào từ hàm trong ổ đĩa bằng hàm seekg( ) và đọc thông tin này vào mảng a với số lượng clusSize nhờ hàm read( ) để có được một mảng các chỉ số hex của thư mục. Ép kiểu mảng a thành mảng động readBytes có kiểu dữ liệu BYTE. Sau đó ta sẽ duyệt bằng vòng lặp được trỏ đến từng byte để xét các thông tin của thư mục. Lúc đọc ta cần kiểm tra xem nó có phải là entry chính, phụ.
* Với entry chính là lưu giữ các thông tin tên, thời gian, size, tình trạng của file, … .Entry phụ là để lưu giữ tên của file nếu như tên file dài quá mức giữ được của entry chính. Như vậy ta xây dựng các vòng lặp để đọc từng kí tự và Push vào Stack.
* Ta xét xem trạng thái của 1 thư mục/ tập tin và thực hiện hiển thị thông tin của nó lên màn hình console. Ta khai báo con trỏ đây được xem là con trỏ dirTempNode tượng trưng cho cây thư mục với kiểu dữ liệu là DIRECTORY. Thực hiện khởi tạo các giá trị của thuộc tính của thư mục/ tập tin này nhờ mảng readBytes.
* Sau khi biết được các thông tin của một thư mục, ta thực hiện in cây thư mục. Nếu trạng thái hiện tại đọc được tại 0B là 0x10 thì đây là thư mục và cần phải hiển thị cây thư mục con bên trong nên ta sẽ gọi hàm readDriectory( ) với tham số là cluster bắt đầu của thư mục hiện tại. Nếu trạng thái đọc được tại 0B là 0x20 thì đây là tập tin và ta thực hiện xem xét phần mở rộng của tập tin này có phải là “txt” hoặc “TXT” hay không. Nếu không thì ta phải dùng phần mềm tương thích đọc nội dung, ngược lại thì ta sẽ hiển thị nội dung của tập tin “txt” qua hàm readContentOfFile( ).
* Kết quả: Kiểm tra ổ đĩa E ở ý 3 (Các ý tưởng giải quyết) ở trên ta nhận thấy kết quả chính xác.



1. Đọc thông tin và hiển thị cây thư mục phân vùng NTFS
2. Đọc thông tin của phân vùng NTFS

* Không như FAT32, nhóm sử dụng class NTFS để lưu các thông tin quan trọng của NTFS

class NTFS

{

private:

HANDLE device;

vector<vector<string>> BootSector;

int bytes\_per\_sector; // Số bytes trên mỗi sector

int sectors\_per\_cluster; // Sc : Số bytes trên mỗi cluster

int begin\_MFT; // cluster bắt đầu MFT

vector<MFT\_ENTRY> MFTEntries;

public:

NTFS(HANDLE disk, vector<vector<string>> sector);

void readBoot\_Sector();

void read\_MFT(vector<vector<string>>, NTFS);

int get\_first\_sector\_MFT();

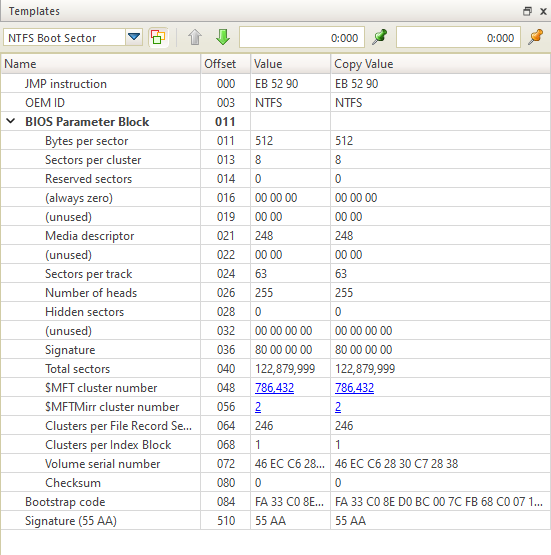
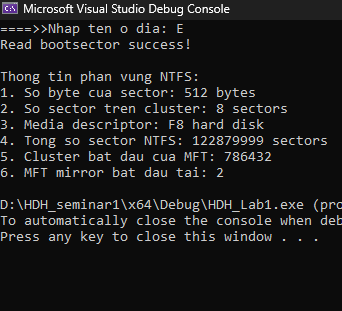
string get\_type\_file(string);

void pushToMFTEntries(MFT\_ENTRY entry);

void findSubDirectory(int parentID = 5, int numTab = -1);

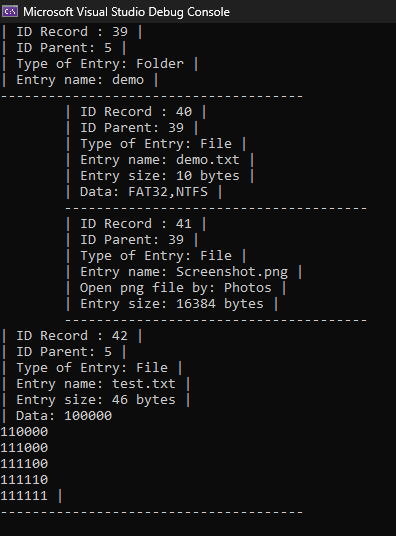
};

* Sau đó ta dùng hàm int ReadBootSectorFAT32(LPCWSTR drive, int readPoint, BYTE sector[512]). Hàm này đọc trên một phân vùng drive (ổ đĩa cần đọc) một sector(512 Byte) từ vị trí readPoint.
* Sau đó ta viết hàm void NTFS::readBoot\_Sector() để xuất ra thông tin cần thiết trong class NTFS.Đã được xử lý bằng các hàm ở trên.
* Để kết thúc ta dùng hàm main để thực thi các hàm bên trên:
* Kết quả: Đối chiếu với kết quả khi dùng phần mềm Disk Editor cho phân vùng FAT32 của ổ đĩa E thì ta thấy kết quả chính xác.



1. Hiển thị cây thư mục của phân vùng NTFS

* Sau khi đã có được các dữ liệu cần thiết của NTFS. Ta dùng hàm read\_MFT để lấy thông tin entry chính và entry phụ. Với entry chính là lưu giữ các thông tin tên, thời gian, size, tình trạng của file, … .Entry phụ là để lưu giữ tên của file nếu như tên file dài quá mức giữ được của entry chính. Như vậy ta xây dựng các vòng lặp để đọc từng kí tự và Push vào Stack (ở trong hàm là sector). Lưu lại những thông tin cần thiết để xử lý như: id\_record, start\_offset\_$standard\_info, size\_$standard\_info, start\_offset\_data\_$standard\_info, start\_offset\_$file\_name, size\_$file\_name, start\_offset\_data\_$file\_name, parent\_id
* Sau khi biết được các thông tin của một thư mục, ta thực hiện in cây thư mục. Nếu trạng thái hiện tại đọc được tại 0B là 0x10 thì đây là thư mục và cần phải hiển thị cây thư mục con bên trong. Nếu trạng thái đọc được tại 0B là 0x20 thì đây là tập tin và ta thực hiện xem xét phần mở rộng của tập tin này có phải là “txt” hoặc “TXT” hay không.
* void MFT\_ENTRY::printEntry(int numTab) để xuất thông tin hay dữ liệu của từng file và folder có trong ổ đĩa.
* Cuối cùng dùng hàm void NTFS::findSubDirectory(int parentID, int numTab) để xuất thông tin và hiển thị cây thư mục.
* Kết quả: Kiểm tra ổ đĩa E ở ý 3 (Các ý tưởng giải quyết) ở trên ta nhận thấy kết quả chính xác.



Tham Khảo

Tham khảo về NTFS: http://ntfs.com/ntfs\_basics.htm

Giáo trình Hệ Điều Hành – Khoa Công Nghệ Thông Tin – Trường Đại Học Khoa Học Tự Nhiên, ĐHQG TPHCM (2019).

Tài liệu và video hướng dẫn của Thầy Lê Viết Long

<https://text.123docz.net/document/5154233-xay-dung-chuong-trinh-doc-fat-cua-dia-cung-voi-dinh-dang-ntfs-va-fat32-tim-hieu-giao-thuc-ftp-xay-dung-ung-dung-truyen-file-tren-mang-theo-mo-hinh-clientserver.htm>

https://www.academia.edu/14578381/%C4%90\_I\_H\_C\_%C4%90%C3%80\_N\_NG\_B%C3%81O\_C%C3%81O\_%C4%90\_%C3%81N\_1E\_C6\_1E\_C0?fbclid=IwAR1ajcgplKk25d4\_NBF9dzzh\_TmW-osCukF09dlUdVkjHCTGq23da7dSclM

https://www.youtube.com/watch?v=4IS1vd8lwOU&t=27s