

Vietnam National University – Ho Chi Minh City
University of Science – Information Technology (High Quality Program)



FILE MANAGEMENT SYSTEM

INTRODUCTION TO OPERATING SYSTEM
PROJECT 01
19CLC6

LECTURERS:
Mr. Nguyen Van Giang
Mr. Le Viet Long

First word

Đây là report về đề án Quản lý hệ thống tập tin trên Windows – Xây dựng chương trình máy tính đọc các thông tin trên phân vùng FAT32 và NTFS do các thành viên nhóm thực hiện.

Đề án này được nhóm chia ra làm 4 phần:

1. Đọc các thông tin chi tiết của phân vùng FAT32
2. Đọc các thông tin chi tiết của phân vùng NTFS
3. Hiển thị cây thư mục của phân vùng FAT32
4. Hiển thị cây thư mục của phân vùng NTFS

Đây là danh sách các thành viên trong nhóm - bảng phân chia công việc – tỉ lệ phần trăm hoàn thành và bảng đánh giá mức độ hoàn thành trên từng yêu cầu và toàn bộ project:

Number	Mission	Unfinished	Attribution
1	Đọc các thông tin chi tiết của FAT32	No	100%
2	Đọc các thông tin chi tiết của NTFS	No	100%
3	Hiển thị cây thư mục của FAT32	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)	90%
4	Hiển thị cây thư mục của NTFS	Yes	0%
Progress:			72.5%

Student's ID	Name	Mission	Attribution
19127095	Ngô Huy Anh	Report + Testing	100%
19127399	Huỳnh Cao Nhật Hiếu	Report + 1	100%
19127444	Ngô Đăng Khoa	1 + 2	100%
19127449	Phùng Anh Khoa	2 + 3	100%
Progress:		Completed	100%

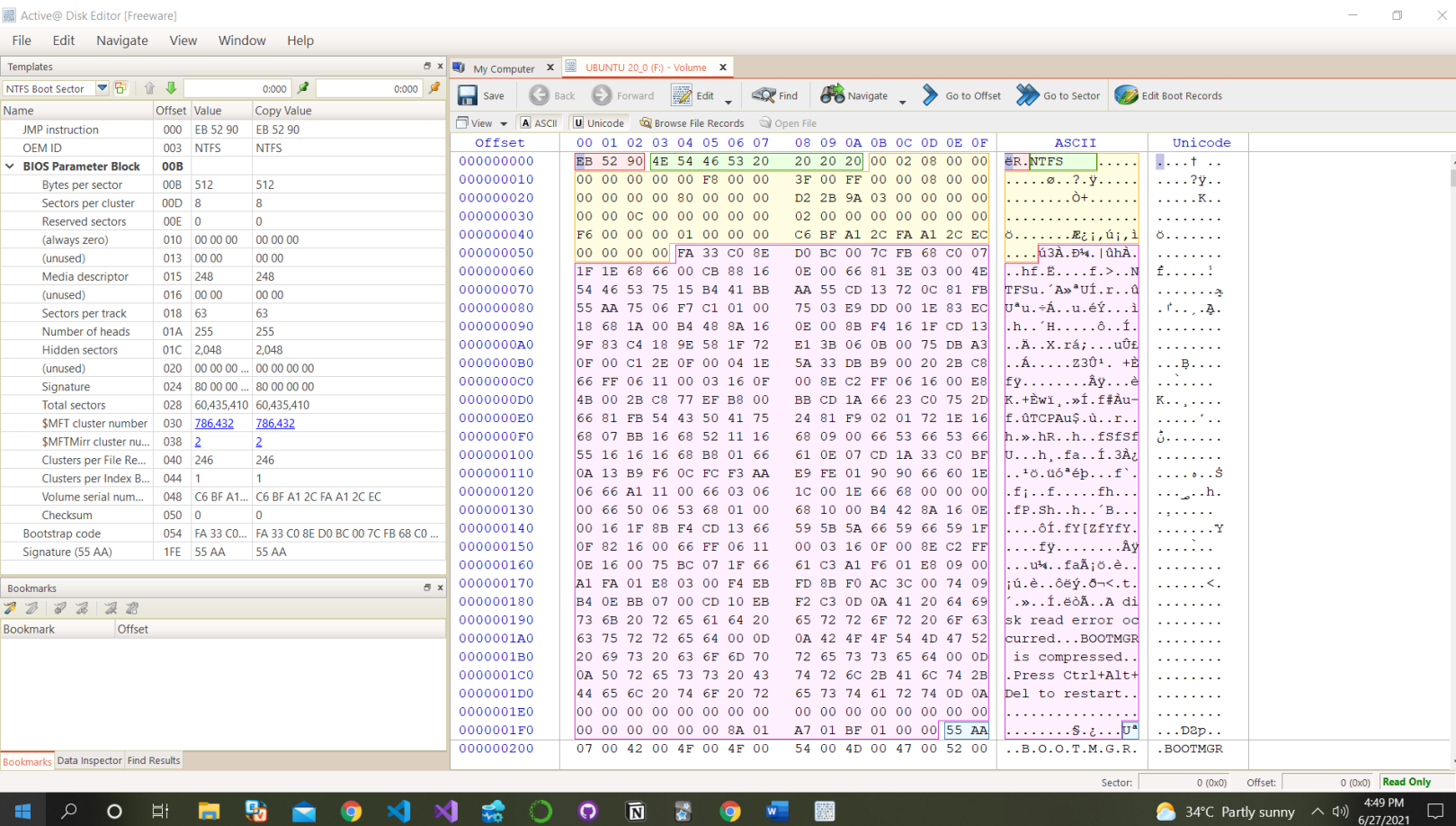
Contents

I. CÁC PHẦN MỀM VÀ CÔNG CỤ BỔ TRỢ	4
II. ĐỌC THÔNG TIN CHI TIẾT PHÂN VÙNG FAT32	5
1. Mô tả.....	5
2. Demo.....	6
III. HIỂN THỊ CÂY THƯ MỤC CỦA FAT32	9
1. Mô tả.....	9
2. Demo	16
IV. ĐỌC THÔNG TIN CHI TIẾT PHÂN VÙNG NTFS	18
1. Mô tả.....	18
2. Demo	18
V. CÁC NGUỒN THAM KHẢO	20
1. Các nguồn tham khảo từ Giáo Trình – Sách – Tài Liệu.....	20
2. Các nguồn tham khảo từ trên Internet.....	20

I. CÁC PHẦN MỀM VÀ CÔNG CỤ BỔ TRỢ

- ✓ Microsoft Visual Studio
- ✓ Disk Editor

Trước tiên, vì muốn kiểm tra xem thông tin của một ổ đĩa đã được đọc đúng chưa nên nhóm em sử dụng phần mềm hỗ trợ: [Disk Editor](#).



Hình ảnh của phần mềm khi đọc thông tin của phân vùng NTFS

II. ĐỌC THÔNG TIN CHI TIẾT PHÂN VÙNG FAT32

1. Mô tả

Offset	Số byte	Nội dung
0	3	Jump_Code: lệnh nhảy qua vùng thông số (như FAT)
3	8	OEM_ID: nơi sản xuất – version, thường là “MSWIN4.1”
B	2	Số byte trên Sector, thường là 512 (như FAT)
D	1	S_C: số sector trên cluster (như FAT)
E	2	S_B: số sector thuộc vùng Bootsector (như FAT)
10	1	N_F: số bảng FAT, thường là 2 (như FAT)
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 - như FAT)
16	2	Không dùng, thường là 0 (số sector của bảng FAT – với FAT)
18	2	Số sector của track (như FAT)
1A	2	Số lượng đầu đọc (như FAT)
1C	4	Khoảng cách từ nơi mô tả vol đến đầu vol (như FAT)
20	4	S_V: Kích thước volume (như FAT)
24	4	S_F: 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 chứa 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)

Để đọc các thông tin chi tiết về bảng FAT32, đầu tiên em xây dựng struct các thông số cần hiển thị của FAT32 dựa trên thành phần và các kiểu dữ liệu của các thông số của FAT32, cũng như số byte mà thông số đó chiếm.

2. Demo

```

1 struct FAT32
2 {
3     uint8_t bootstrapJump[3]; // JumpCode ( 0 - 3 )
4     uint8_t oem[8]; // OEM name/version (3 - 8)
5     uint8_t bytePerSector[2]; // Byte per sector (B - 2)
6     uint8_t sectorPerCluster; // Sector per cluster (D - 1)
7     uint8_t reservedSector[2]; // Number of reserved sectors (E - 2)
8     uint8_t fatCopy; // Number of FAT copies (10 - 1)
9     uint8_t rdetEntry[2]; // Number of root directory entries (11 - 2)
10    uint8_t sector_of_Volume[2]; // Total number of sector of Volume (13 - 2)
11    uint8_t mediaType; // Media descriptor type (15 - 1)
12    uint8_t sectorPerFAT[2]; // Sector per FAT, 0 for FAT32 ( 16 - 2 )
13    uint8_t sectorPerTrack[2]; // Sector per track ( 18 - 2 )
14    uint8_t head[2]; // Number of heads ( 1A - 2 )
15    uint8_t hiddenSector[4]; // Number of hidden sectors ( 1C - 4 )
16    uint8_t Total_sector[4]; // Volume size ( 20 - 4 ) Sv
17    uint8_t SectorperFAT[4]; // Sector per FAT ( 24 - 4 ) Sf
18    uint8_t Extended_flag[2]; // (28-2)
19    uint8_t Version[2]; // Version of FAT32 (2A - 2)
20    uint8_t Root_cluster[4]; // Root Cluster (2C - 4)
21    uint8_t System_inf_sector[2]; // Empty cluster inf (30 - 2)
22    uint8_t Backup_boot_sec[2]; // Backup Boot Sector (32-2)
23    uint8_t S6[12]; // (34 - C )
24    uint8_t Physical_drive; // Physical drive(40 - 1)
25    uint8_t Reversed; // (41 - 1)
26    uint8_t Extended_signature; // (42 - 1)
27    uint8_t Serial[4]; // (43 - 4)
28    uint8_t Volume_label[11]; // (47 - B)
29    uint8_t Fat_name[8]; // (52 - 8)
30    uint8_t BootstrapCode[420]; // (5A - 1A4)
31    uint8_t signatrue[2]; // (1FE - 2)
32 };

```

FAT32 lưu trữ dữ liệu theo kiểu Little endian, vì vậy cho nên đối với các dữ liệu kiểu số, FAT32 sẽ đọc ngược từ byte cuối cùng. Từ đó em xây dựng hàm **reverseByte** để đảo thứ tự các byte lại giúp đọc đúng thông tin.

```

3 uint64_t reverseByte(uint8_t* byte, unsigned int count)
4 {
5     uint64_t result = 0;
6     int i;
7     for (i = count - 1; i ≥ 0; i--)
8         result = (result << 8) | byte[i];
9
10    return result;
11 }

```

Tiếp theo là hàm quan trọng nhất của chương trình để đọc sector.

int ReadSector(LPCWSTR drive, int readPoint, BYTE sector[512], long posSector)

```

1  int ReadSector(LPCWSTR drive, int readPoint, BYTE sector[512], long posSector)
2  {
3      int retCode = 0;
4      DWORD bytesRead;
5      HANDLE device = NULL;
6
7      device = CreateFile(drive,          // Drive to open
8                          GENERIC_READ,  // Access mode
9                          FILE_SHARE_READ | FILE_SHARE_WRITE, // Share Mode
10                         NULL,           // Security Descriptor
11                         OPEN_EXISTING,  // How to create
12                         0,              // File attributes
13                         NULL);          // Handle to template
14
15     if (device == INVALID_HANDLE_VALUE) // Open Error
16     {
17         printf("CreateFile: %u\n", GetLastError());
18         return 1;
19     }
20
21     SetFilePointer(device, readPoint + 512 * posSector, NULL, FILE_BEGIN); // Set a Point to Read
22
23     if (!ReadFile(device, sector, 512, &bytesRead, NULL))
24     {
25         printf("ReadFile: %u\n", GetLastError());
26     }
27 }

```

Từ đó, chúng ta đã có thể đọc phân vùng của 1 ổ cứng, chỉ cần truyền đường dẫn ổ cứng đó vào hàm.

```
ReadSector(L"\\\\.\\F:", 0, sector, 0);
```

Ta cũng sử dụng hàm memcpy để sao chép num byte từ vị trí mà sector trả tới vị trí của FAT32. do khi thao tác với FAT, dữ liệu sẽ được truy cập như là mảng byte 1 chiều

```
memcpy(&fat32, sector, 512);
```


III. HIỂN THỊ CÂY THƯ MỤC CỦA FAT32

1. Mô tả

Để có thể làm phần này ta cần biết được sector đầu của RDET để làm. Ta có thể kiểm được sector đầu tiên của RDET bằng các thông tin kiểm được ở phần 1 (Đọc thông tin chi tiết phân vùng FAT32).

Khi đó:

- Reserved sector = 2238 = SB
- Number of FATS = 2 = NF
- Sectors per FAT = 15265 = SF

⇒ Sector đầu tiên của RDET = SB + NF*SF = 32768

Có sector này giờ ta đơn giản là tìm cách chạy đến vị trí sector này để đọc thông tin của RDET

Name	Offset	Value	Copy Value
Bytes per sector	011	8	512
Sectors per cluster	013	0	8
Reserved sectors	014	0	2,238
Number of FATS	016	0	2
(unused)	017	00 00	00 00
(unused)	019	00 00	00 00
Media descriptor	021	0x00	0xF8
(unused)	022	C4 A1	00 00
Sectors per track	024	21,208	63
Number of heads	026	0	255
Hidden sectors	028	0	63
Total sectors	032	1,224,745,026	15,663,312
Sectors per FAT	036	1,711,304,192	15,265
Extended flags	040	28,416	0
Version	042	3,840	0
Root cluster	044	7,500,288	2
System Information sector	048	109	1
Backup Boot sector	050	97	6
(reserved)	052	74 00 69 00 ...	00 00 00 00 00 00 ...

Field	Offset	Value
Reserved sectors	014	0, 2,238
Number of FATS	016	0, 2
Sectors per FAT	036	1,711,304,192, 15,265

FIRST SECTOR OF RDET = SB + NF*SF

Những thông tin kiểm được bằng phần mềm

```

Bytes per sector: 512
Sector per cluster: 8
Reserved sectors (Sb): 2238
FAT copies: 2
Total sector: 15663312
Sector per FAT: 15265
Root directory entries: 0
first sector of FAT1: 2238
first sector of RDET: 32768
first sector of DATA: 32768

```

Những thông tin kiểm được bằng Code

Lúc này, ta được sector đầu tiên của RDET = 32768

Bây giờ ta cần máy có thể đọc được thông tin ở vị trí sector 32768

Em tùy chỉnh chút về code readsector mà thầy đã gửi trước đó, em thêm biến posSector để nó chạy tới sector em muốn đọc

```

int ReadSector(LPCWSTR drive, int readPoint, BYTE sector[512], long posSector)
{
    int retCode = 0;
    DWORD bytesRead;
    HANDLE device = NULL;

    device = CreateFile(drive, // Drive to open
        GENERIC_READ, // Access mode
        FILE_SHARE_READ | FILE_SHARE_WRITE,
        NULL, // Security Descriptor
        OPEN_EXISTING, // How to create
        0, // File attributes
        NULL); // Handle to template

    if (device == INVALID_HANDLE_VALUE) // Open Error
    {
        printf("CreateFile: %u\n", GetLastError());
        return 1;
    }

    SetFilePointer(device, readPoint + 512 * posSector, NULL, FILE_BEGIN);
    if (!ReadFile(device, sector, 512, &bytesRead, NULL))
    {
        printf("ReadFile: %u\n", GetLastError());
    }
}

```

Bây giờ em có thể đọc thông tin ở sector chỉ định

```
ReadSector(L"\\\\.\\H:", 0, entry, size);
```

Ở đây size của em có giá trị = với sector đầu tiên của RDET

Mà trong RDET có 2 loại sector: entry chính và entry phụ

Trong RDET có 2 loại sector: entry chính và entry phụ

Name	Offset	Value	Offset	Value	ASCII
Long entry (f)	000		00 01 02 03 04 05 06 07	08 09 10 11 12 13 14 15	
Long entry (lcalAgents.pdf)	032	ENTRY CHÍNH	0016777404	08 52 D8 52 00 00 D1 A6 B1 52 06 00 D5 E2 18 00	SRSR..R:R..ôâ..
Long entry (lecture07-Log)	064		0016777440	43 66 00 00 00 FF FF FF FF FF FF 0F 00 EB FF FF	SE...yyyyyy..yyy
Short entry (LECTUR-2.PDF)	096	ENTRY PHỤ	0016777456	FF FF FF FF FF FF FF FF FF FF 00 00 FF FF FF FF	yyyyyyyyyy..yyy
			0016777472	02 69 00 63 00 61 00 6C 00 41 00 0F 00 EB 67 00	.i.c.a.l.A...eg.
			0016777488	65 00 6E 00 74 00 73 00 2E 00 00 00 70 00 64 00	e.n.t.s....p.d.
			0016777504	01 4C 00 65 00 63 00 74 00 75 00 0F 00 EB 72 00	.L.e.c.t.u...er.
			0016777520	65 00 30 00 37 00 2D 00 4C 00 00 00 6F 00 67 00	e.o.7.-.L...o.g.
			0016777536	4C 45 43 54 55 52 7E 32 50 44 46 20 00 08 C9 A1	LECTUR-2PDF..E
			0016777552	D8 52 D8 52 00 00 D0 A6 B1 52 95 01 F0 4A 2B 00	SRSR..D:R..ôJ+.

Cứ có n entry phụ thì sẽ có 1 entry chính của các entry đó

Nhiệm vụ của entry chính là lưu giữ các thông tin tên, thời gian, size, tình trạng của file, ...

Còn nhiệm vụ của entry phụ là để lưu giữ tên của file nếu như tên file dài quá mức giữ được của entry chính

Ta có tính chất của 1 file sẽ gồm 1 entry chính và 0 hoặc nhiều entry phụ

Mà nó đọc entry phụ trước rồi mới đọc tới entry chính

Mỗi entry chiếm 32 byte trong sector với cấu trúc khác nhau tùy vào loại chính hay phụ, ta có thể tìm hiểu trong [FAT](#)

Bằng cách đó em có thể xài 1 hàm struct để đọc ngược thông tin, em sẽ đọc các thông tin của entry phụ và cộng dồn tên nó lại theo chiều ngược lại đọc trước sẽ được cộng sau và nếu gặp entry chính thì tên đó sẽ là của entry đó

```

File Attribute : Long File Name
sequenceNo: 67
fileName_Part1: h.pdf
fileName_Part2:
fileName_Part3:
FULL: h.pdf

File Attribute : Long File Name
sequenceNo: 2
fileName_Part1: ersar
fileName_Part2: ialSea
fileName_Part3: rc
FULL: ersarialSearc

File Attribute : Long File Name
sequenceNo: 1
fileName_Part1: Lectu
fileName_Part2: re05-A
fileName_Part3: dv
FULL: Lecture05-Adv
    
```

NAME OF FILE

```

////////// INFOR OF ENTRY //////////
name: Lecture05-AdversarialSearch.pdf
file status: Exist
file Size: 2933353
sector Index: 32769
    
```

Còn các thông tin khác em dùng struct và memcpy để đọc thông tin

```
memcpy(&rootEntry, pEntry, 32);
```

- 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

```

typedef struct DirectoryEntry
{
    uint8_t name[8];
    uint8_t fileType[3];
    uint8_t attrib;
    uint8_t userattrib;
    uint8_t Millisecondtime;

    uint16_t createtime;
    uint16_t createdate;
    uint16_t accessdate;
    uint16_t clusterhigh;

    uint16_t modifiedtime;
    uint16_t modifieddate;
    uint16_t clusterlow;
    uint8_t filesize[4];

    string fullname;
} mainEntry;
    
```

structure cho entry chính

```

typedef struct ShortFileName
{
    uint8_t sequenceNo;
    uint8_t fileName_Part1[10];
    uint8_t fileattribute;

    uint8_t reserved_1;
    uint8_t checksum;

    uint8_t fileName_Part2[12];
    uint8_t NOTHING[2];
    uint8_t fileName_Part3[4];
} extraEntry;
    
```

structure cho entry phụ

```

struct inforEntry
{
    string name;
    string fileStatus;
    int fileSize;
    int sectorIndex;
};
    
```

Thông tin xuất ra của file đó

Để đọc 1 entry ta đọc 32 byte từ sector

Và kiểm tra xem nó là entry chính, phụ, xóa hay không có

```
if (rootEntry[11] == 0x0F)
{
```

=> Entry phụ

```
if (rootEntry[0] == 0xE5)
{
```

=> Entry bị xóa

```
if (rootEntry[0] == 0x00)
```

=> Không có entry nào hết

Còn lại là entry chính

Bây giờ ta đã xong được ý đầu của FAT32

Sang phần này, ta có được tên file rồi, ta xài hàm đọc file bình thường với đường dẫn

```
printf("forOfMainEntry(entry[i]);
string a = "H:/" + entry[i].fullname;
```

Ta đọc thông tin của file nếu file có tên mở rộng là txt bằng thư viện fstream với đường dẫn trên

```
cout << endl << "INFOR: " << endl;
ifstream ifs;
ifs.open(a);
if (ifs.fail())
    cout << "cant open file" << endl;

string docfile = "", dong = "";
while (!ifs.eof()) {
    getline(ifs,dong);
    docfile += dong + "\n";
}

cout << docfile << endl;
```

Cách đọc thông tin của tập tin hoặc đọc các thư mục con của thư mục gốc ở đây ta có cluster dẫn tới vị trí đọc file của FOLDER HDH là 6

Biết 1 cluster có 8 sector

- DDC tummy cluster used with several other items.

Nếu biết giá trị cluster (kí hiệu: N) bất kì, có thể tính được sector bắt đầu (sector tương đối) của cluster đó bằng công thức sau:

- FirstSectorOfCluster: sector bắt đầu của cluster
- SecPerClus: số sector/cluster
- FirstDataSector: sector đầu tiên của cluster số 2 trong vùng Data

Ví dụ, giá trị của cluster bắt đầu $N = 0x00040832$, đổi sang hệ thập phân là 264242.
 Áp dụng công thức trên:

Ta có công thức tính sector đầu tiên để tìm data của file hoặc folder

FAT Directory Entry

Name	Offset	Value
Short entry (HDH)	000	HDH
File name	000	HDH
File extension	008	
Attributes	011	0x30
(reserved)	012	0
Created time refinement in...	013	117
Created date/time	014	6/25/2021 4:41 PM
Last access date	018	6/25/2021
First cluster (high word)	020	0
Modified date/time	022	5/31/2021 7:15 PM
First cluster (low word)	026	6
File size	028	0

Offset	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	ASCII
0016777264	6D	00	61	00	74	00	69	00	6F	00	00	00	6E	00	00	00	m.a.t.i.o...n...
0016777280	01	53	00	79	00	73	00	74	00	65	00	0F	00	72	6D	00	.s.y.s.t.e...m.
0016777296	20	00	56	00	6F	00	6C	00	75	00	00	00	6D	00	65	00	.v.o.l.u...m.e.
0016777312	53	59	53	54	45	4D	7E	31	20	20	20	16	00	75	25	85	SYSTEM~1 ..u.
0016777328	D9	52	D9	52	00	00	26	85	D9	52	03	00	00	00	00	00	ÙÒÙÒ...&.ÙÒ.....

Ta có:

- First data sector = 32768
 - N = 6
 - Secperclus = 8
- ⇒ $8 * (6-2) + 32768 = 32800$ là sector đầu tiên chứa thông tin của file/folder đó

FAT Directory Entry

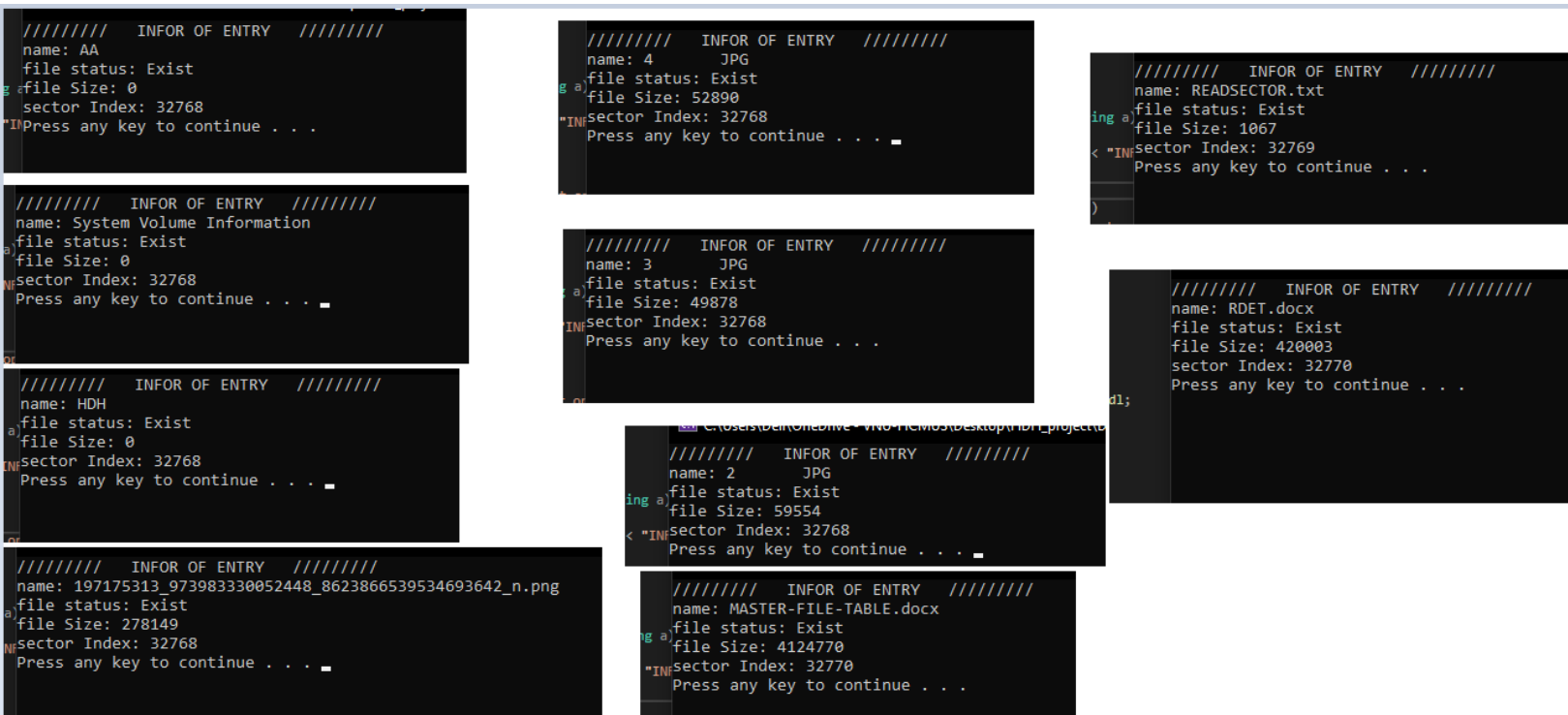
Name	Offset	Value
Short entry (LT)	000	LT
File name	000	LT
File extension	008	
Attributes	011	0x30
(reserved)	012	0
Created time refinement in...	013	119
Created date/time	014	6/25/2021 4:41 PM
Last access date	018	6/25/2021
First cluster (high word)	020	0
Modified date/time	022	6/7/2021 8:15 PM
First cluster (low word)	026	7
File size	028	0

Offset	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	ASCII
0016793584	10	0A	6A	95	10	0A	6A	95	10	0A	6A	95	10	0A	6A	95	..j...j...j...j...
0016793600	2E	20	20	20	20	20	20	20	20	20	20	10	00	75	2D	85
0016793616	D9	52	D9	52	00	00	2E	85	D9	52	06	00	00	00	00	00	ÙÒÙÒ...ÙÒ.....
0016793632	2E	2E	20	20	20	20	20	20	20	20	20	10	00	75	2D	85
0016793648	D9	52	D9	52	00	00	2E	85	D9	52	00	00	00	00	00	00	ÙÒÙÒ...ÙÒ.....
0016793664	5C	54	20	20	20	20	20	20	20	20	20	30	00	77	2D	85
0016793680	D9	52	D9	52	00	00	E8	A1	C7	52	07	00	00	00	00	00	ÙÒÙÒ...&_ÇR.....
0016793696	54	48	20	20	20	20	20	20	20	20	20	30	00	B8	2D	85	TH
0016793712	D9	52	D9	52	00	00	5A	7F	D9	52	CB	04	00	00	00	00	ÙÒÙÒ...z.ÙÒÈ.....

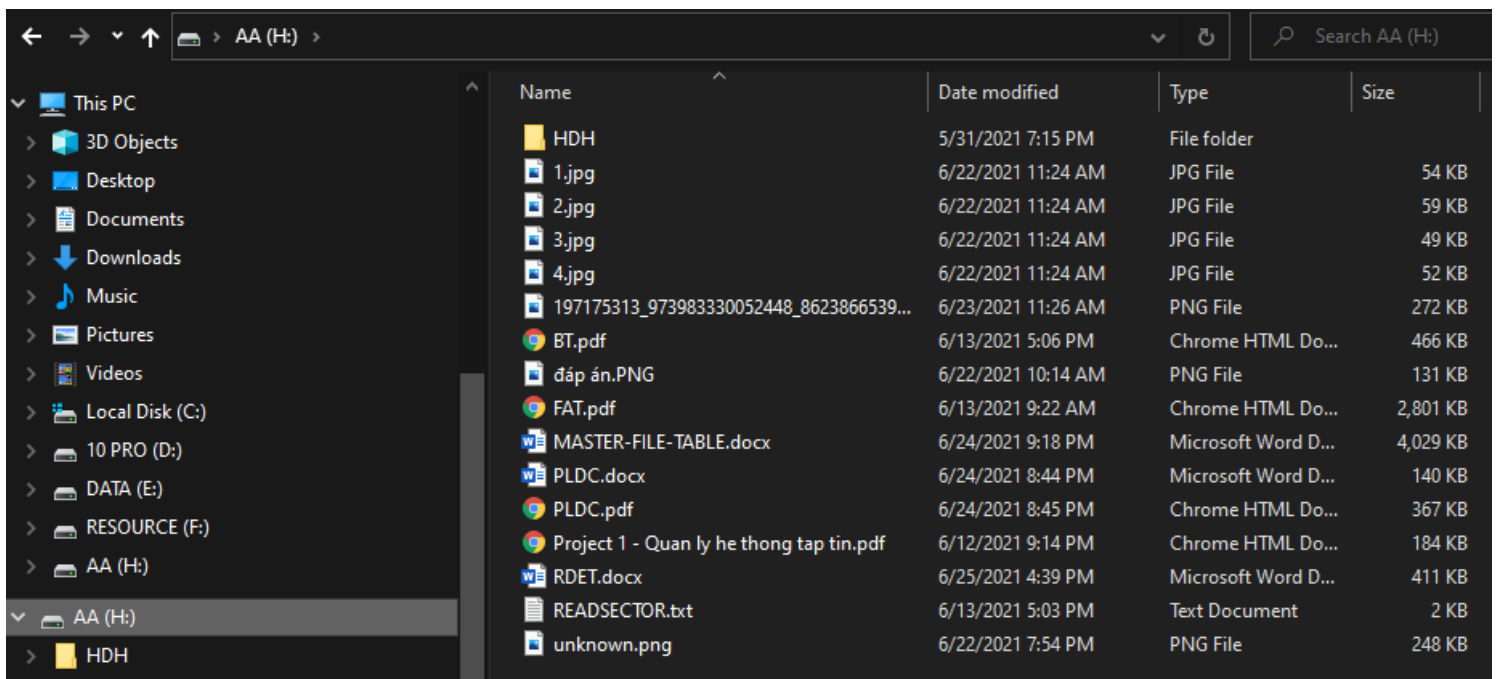
Ta có vị trí sector đầu của file/folder và có 2 cách tìm thông tin. Vì nếu là file thì nó sẽ chạy tới nơi có thông tin. Còn folder thì sẽ chạy ra thông tin các file/folder khác nằm trong nó và tương tự như trên.

2. Demo

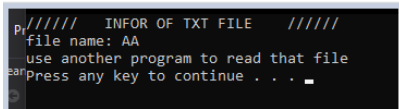
Hiện thị cây thư mục gốc vì nhiều quá nên em đã lược bớt vài cái



Cây thư mục gốc



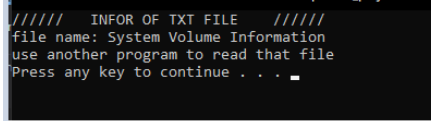
Hiển thị thông tin của file dưới dạng txt, còn lại thì xài phần mềm tương thích để đọc



```

Pr///// INFOR OF TXT FILE /////
file name: AA
use another program to read that file
Press any key to continue . . .

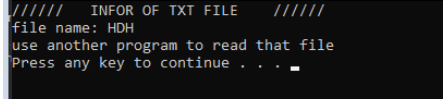
```



```

///// INFOR OF TXT FILE /////
file name: System Volume Information
use another program to read that file
Press any key to continue . . .

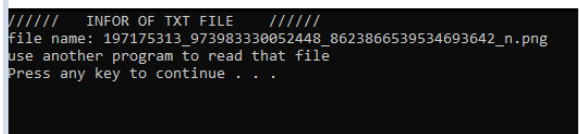
```



```

///// INFOR OF TXT FILE /////
file name: HDH
use another program to read that file
Press any key to continue . . .

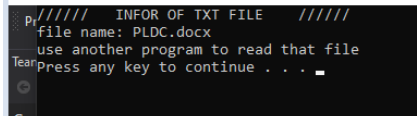
```



```

///// INFOR OF TXT FILE /////
file name: 197175313_973983330052448_8623866539534693642_n.png
use another program to read that file
Press any key to continue . . .

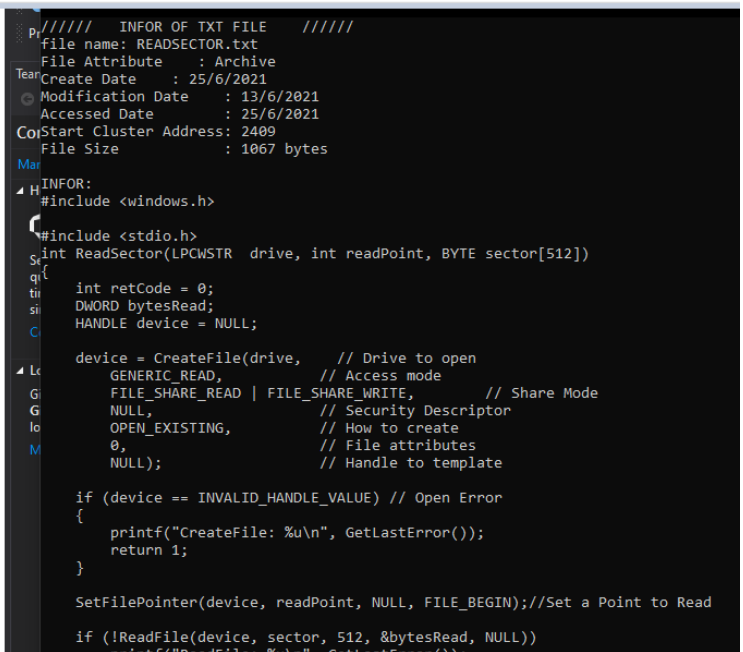
```



```

Pr///// INFOR OF TXT FILE /////
file name: PLDC.docx
use another program to read that file
Press any key to continue . . .

```



```

Pr///// INFOR OF TXT FILE /////
file name: READSECTOR.txt
File Attribute : Archive
Create Date : 25/6/2021
Modification Date : 13/6/2021
Accessed Date : 25/6/2021
Start Cluster Address: 2409
File Size : 1067 bytes

#include <windows.h>
#include <stdio.h>
int ReadSector(LPCWSTR drive, int readPoint, BYTE sector[512])
{
    int retCode = 0;
    DWORD bytesRead;
    HANDLE device = NULL;

    device = CreateFile(drive, // Drive to open
        GENERIC_READ, // Access mode
        FILE_SHARE_READ | FILE_SHARE_WRITE, // Share Mode
        NULL, // Security Descriptor
        OPEN_EXISTING, // How to create
        0, // File attributes
        NULL); // Handle to template

    if (device == INVALID_HANDLE_VALUE) // Open Error
    {
        printf("CreateFile: %u\n", GetLastError());
        return 1;
    }

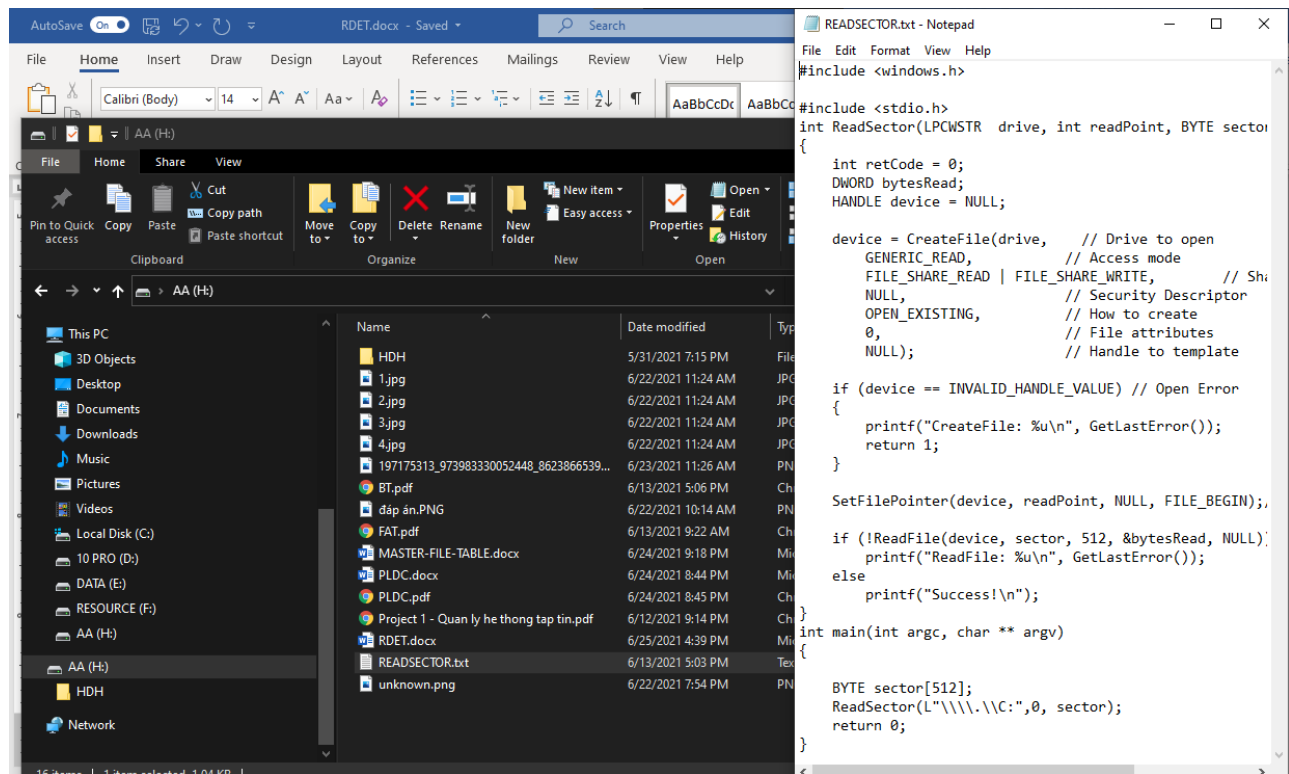
    SetFilePointer(device, readPoint, NULL, FILE_BEGIN); // Set a Point to Read

    if (!ReadFile(device, sector, 512, &bytesRead, NULL))
        printf("ReadFile: %u\n", GetLastError());
}

int main(int argc, char ** argv)
{
    BYTE sector[512];
    ReadSector(L"\\\\.\\C:", 0, sector);
    return 0;
}

```

Còn đây là vị trí file txt đã được đọc



IV. ĐỌC THÔNG TIN CHI TIẾT PHÂN VÙNG NTFS

1. Mô tả

"NTFS"

Offset	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	ASCII
0000000000000	EB	52	90	4E	54	46	53	20	20	20	20	00	02	08	00	00	R NTFS
0000000000016	00	00	00	00	00	F8	00	00	3F	00	FF	00	00	60	3B	68ø...?.ý...';h
0000000000032	00	00	00	00	80	00	80	00	FF	FF	34	0C	00	00	00	00ýý4.....
0000000000048	00	00	0C	00	00	00	00	00	02	00	00	00	00	00	00	00
0000000000064	F6	00	00	00	01	00	00	00	22	F0	32	EA	22	33	EA	24	ø....."ð2ð"3ð\$
0000000000080	00	00	00	00	FA	33	C0	8E	D0	BC	00	7C	FB	68	C0	07ú3À.Đw. úhÀ.
0000000000096	1F	1E	68	66	00	CB	88	16	0E	00	66	81	3E	03	00	4E	..hf.È....f.>..N
0000000000112	54	46	53	75	15	B4	41	BB	AA	55	CD	13	72	0C	81	FB	TFSu.'A»°UÍ.r..û
0000000000128	55	AA	75	06	F7	C1	01	00	75	03	E9	DD	00	1E	83	EC	U°u.÷Á..u.éý...i
0000000000144	18	68	1A	00	B4	48	8A	16	0E	00	8B	F4	16	1F	CD	13	.h..'H.....ð..í.
0000000000160	9F	83	C4	18	9E	58	1F	72	E1	3B	06	0B	00	75	DB	A3	..Ă..X.rá;...uŮÈ
0000000000176	0F	00	C1	2E	0F	00	04	1E	5A	33	DB	B9	00	20	2B	C8	..Á.....z3Ů°. +È
0000000000192	66	FF	06	11	00	03	16	0F	00	8E	C2	FF	06	16	00	E8	fý.....Ăý...è
0000000000208	4B	00	2B	C8	77	EF	B8	00	BB	CD	1A	66	23	C0	75	2D	K.+ÈwI.,»Í.f#Àu-
0000000000224	66	81	FB	54	43	50	41	75	24	81	F9	02	01	72	1E	16	f.úTCPAu\$.ù...r..

Với NTFS em cũng xài structure với hàm memcpy để sao chép dữ liệu vào giống với FAT32

NTFS chỉ có cấu trúc # FAT32,16,12 nhưng thông tin để đọc nó vẫn nằm ở sector đầu tiên và có dấu hiệu nhận biết ở offset 03->10 với thông tin là "NTFS" để ta có thể dễ dàng nhận diện nó so với các FAT khác.

2. Demo

Đầu tiên để xác nhận xem nó có phải NTFS hay FAT32 em memcpy dữ liệu vào 1 structue được tạo để ứng với tt lưu trữ trong sector đầu tiên và em chỉ so sánh kí từ đầu tiên của oemid để xem nó có phải NTFS hay không nếu phải thì em in thông tin ra còn không thì nó là FAT32.

```
typedef struct NTFS
{
    uint8_t JumpCode[3]; // Field Name - BYTE // Jump Instruction - 3
    uint8_t OEMID[8]; // OEM ID - 8
    uint8_t Bytes_Sector[2]; // Bytes Per Sector - 2
    uint8_t Sectors_Cluster; // Sectors Per Cluster - 1
    uint8_t Reserved_Sector[2]; // Reserved Sectors - 2
    uint8_t always_0[3]; // always 0 - 3
    uint8_t not_used_by_NTFS1[2]; // not used by NTFS - 2
    uint8_t Media_Descriptor; // Media Descriptor - 1
    uint8_t always_0_2[2]; // always 0 - 2
    uint8_t sectors_Track[2]; // Sectors Per Track - 2
    uint8_t number_of_Heads[2]; // Number Of Heads - 2
    uint8_t Hidden_sectors[4]; // Hidden Sectors - 4
    uint8_t not_used_by_NTFS2[4]; // not used by NTFS - 4
    uint8_t not_used_by_NTFS3[4]; // not used by NTFS - 4
    long long total_sectors; // Total Sectors - 8
    long long logical_MFT; // Logical Cluster Number for the file $MFT - 8
    long long logical_MFTMirr; // Logical Cluster Number for the file $MFTMirr - 8
    uint8_t Cluster_FRs[4]; // Clusters Per File Record Segment - 4
    uint8_t Cluster_Index_Buffer; // Clusters Per Index Buffer - 1
    uint8_t not_used_by_NTFS4[3]; // not used by NTFS - 3
    long long Volume; // Volume Serial Number - 8
    uint8_t checksum[4]; // Checksum - 4
    uint8_t Bootstrap_Code[426]; // Bootstrap Code - 426
    uint8_t EndOfSectorMarker[2]; // EndOfSectorMarker - 2
} NTFS;
```

```
ReadSector(L"\\\\.\\H:", 0, sector, 0);
memcpy(&ntfs, sector, 512);
```

```
if (ntfs.OEMID[0] == 'N')
{
    PrintFloppyInformationNTFS(ntfs);
    system("pause");
    system("cls");
}
```

Em chỉ in 7 thông tin quan trọng của NTFS và để giúp cho việc làm bài sau

Về mặt chung với FAT32: thông tin đều nằm ở sector đầu chỉ thay đổi ở một chút về mặt thông Tin và các tổ chức [NTFS](#)

```
printf("FAT: %s\n", _ntfs.OEMID); //Loai FAT
printf("Bytes per sector: %d\n", reverseByte( _ntfs.Bytes_Sector, 2)); // So Byte cho 1 sector
printf("Sector per cluster: %d\n", _ntfs.Sectors_Cluster); // So Sector cho 1 cluster
printf("Reserved Sector: %d\n", reverseByte( _ntfs.Reserved_Sector, 2)); // So Sector vùng boot Sector
printf("Total Sectors : %lld\n", _ntfs.total_sectors); //Tong so sector, kích thước Volume
printf("Logical Cluster Number for the file $MFT : %lld\n", _ntfs.Logical_MFT); // so cum cluster cho MFT
printf("Logical Cluster Number for the file $MFTMirr : %lld\n", _ntfs.Logical_MFTMirr); // so cum cluster cho MFTMIRR
long long s = _ntfs.Logical_MFT * _ntfs.Sectors_Cluster;
printf("First Sector of Master File Table: %lld\n", s);
```

Kết quả:

```

long long Pri
{
    printf("F
    FAT: NTFS
    printf("
    printf("B
    printf("B
    printf("S
    printf("F
    printf("L
    printf("L
    long long
    printf("F
    system("p
    system("c
    return 0;
}

//////////      INFOR OF BOOT SECTOR      //////////
Floppy Disk Information:
=====
Bytes per sector: 512
Sector per cluster: 8
Reserved Sector: 0
Total Sectors : 15663311
Logical Cluster Number for the file $MFT : 786432
Logical Cluster Number for the file $MFTMirr : 2
First Sector of Master File Table: 6291456
Press any key to continue . . .

```

V. CÁC NGUỒN THAM KHẢO**1. Các nguồn tham khảo từ Giáo Trình – Sách – Tài Liệu**

- Giáo trình Hệ Điều Hành – Khoa Công Nghệ Thông Tin – Trường Đại Học Khoa Học Tự Nhiên, ĐHQG TP HCM (2019) – Nhà xuất bản Khoa Học và Kỹ Thuật – Tác giả: Thầy Trần Trung Dũng và Thầy Phạm Tuấn Sơn
- Tài liệu và video hướng dẫn của Thầy Lê Viết Long – Giảng viên môn Hệ Điều Hành – Lớp 19CLC6 – Trường Đại Học Khoa Học Tự Nhiên, ĐHQG TP HCM

2. Các nguồn tham khảo từ trên Internet

- http://ntfs.com/ntfs_basics.htm
- https://www.codeguru.com/cpp/cpp/cpp_mfc/files/article.php/c13809/Extract-Floppy-Disk-Geometry-from-the-Boot-Sector.htm

- https://www.codeguru.com/cpp/cpp/cpp_mfc/files/article.php/c13831/FAT-Root-Directory-Structure-on-Floppy-Disk-and-File-Information.htm
- https://www.codeguru.com/cpp/cpp/cpp_mfc/files/article.php/c13907/Long-File-Name-LFN-Entries-in-the-FAT-Root-Directory-of-Floppy-Disks.htm

THE END