

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN
[ĐỀ TÀI]
Hệ thống quản lý tập tin

[Giáo viên hướng dẫn]
Lê Việt Long

Môn: Hệ Điều Hành
Thành phố Hồ Chí Minh 2022

First word

Đây là báo cáo 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. Hiện thị cây thư mục của phân vùng FAT32
4. Hiện 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:

Số thứ tự	Yêu cầu	Chưa hoàn thành	Mức độ hoàn thành
1	Đọc các thông tin chi tiết của FAT32		90%
2	Đọc các thông tin chi tiết của NTFS		90%
3	Hiện thị cây thư mục FAT32		90%
4	Hiện thị cây thư mục NTFS	Không hiện thị được Folder trong Folder	80%
Mức độ hoàn thành đề án			85% %

MSSV	Họ và Tên	Nhiệm Vụ	Mức độ hoàn thành
20120444	Nguyễn Chí Công	Tester+Báo Cáo	100%
20120443	Nguyễn Tấn Chử	Coder FAT32	100%
20120446	Nguyễn Đình Cường	Coder FAT32	100%
20120447	Trịnh Quốc Cường	Coder NTFS	100%
20120383	Nguyễn Đức tiến	Coder NTFS	100%

Contents

I.Các phần mềm và công cụ hỗ trợ

II.Phân vùng FAT32

1.Đọc thông tin chi tiết của phân vùng FAT32

2.Hiển thị cây thư mục FAT32

III.Phân vùng NTFS

1.Đọc thông tin chi tiết của phân vùng NTFS

2.Hiển thị cây thư mục NTFS

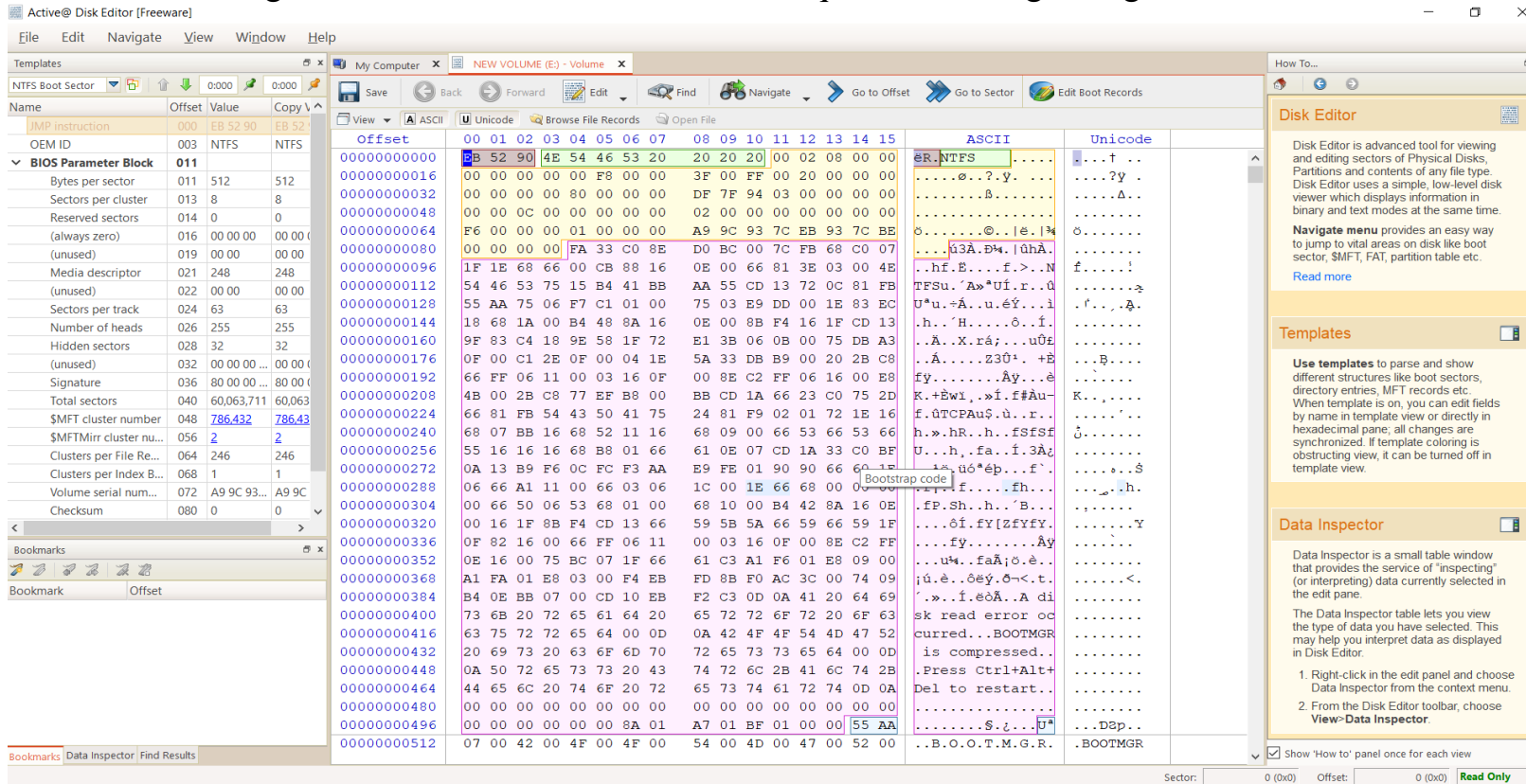
IV.Các nguồn tham khảo

I. Các phần mềm và công cụ hỗ trợ

∇ Microsoft Visual Studio

∇ Disk Editor

Ta dùng Disk Editor để kiểm tra xem các kết quả đọc có đúng không:



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

II. Phân vùng FAT32

1. Đọc thông tin chi tiết của phân vùng FAT32

Xây dựng class với thuộc tính

```
private:
    LPCWSTR _drive;
    HANDLE _device;

    int _bytePerSector; // so byte/sector
    int _sectorPerCluster; // so sector/cluster
    int _rsvdSecCnt; // so sector trước bảng FAT
    int _numberOfFAT; // so bảng FAT
    int _totSec32; // so sector trong volumn
    int _FATSector32; // so sector trong 1 bảng FAT
    int _rootClus; // chỉ số cluster đầu tiên của RDET

private:
    int _level = 0; // dùng xác định cấp độ của tập tin/ thư mục
```

```
FAT32();
FAT32(LPCWSTR drive);

// doc sector
int ReadSector(LPCWSTR _drive, int readPoint, BYTE sector[512]);

// chuyen tu byte sang int
int64_t get_Int_From_Bytes(BYTE sector[], int offset, int size);
// chuyen tu bytes sang string
wstring get_String_From_Bytes(BYTE sector[], int offset, int size, bool isShort);
// tìm sector đầu tiên của cluster
int find_First_SectorOfCluster(int cluster);
// lấy thông tin của tập tin
void get_File_Info(BYTE sector[], int firstCluster);
// lấy ra file cluster
vector<int> getClusters(int firstCluster);
// lấy ra những sector của cluster tương ứng
vector<int> get_Sector_Of_FileSectors(vector<int> fileClusters);
// cấu hình console các thư mục/ tập tin con
void printTab();
// lấy thông tin size
void getSize(BYTE sector[], int index);
// in thông tin FAT32
void printInfoFAT32();
// lấy thông tin của cây thư mục
void getDirectory(int readPoint);
// lấy cluster đầu tiên của cây thư mục
int get_Root_of_Cluster();

void ReadData(wstring fileExtension, int firstCluster);
```

và các phương thức

đầu tiên ta có hàm chuyển đổi từ byte sang int để lấy được thông tin của các

```
int64_t FAT32::get_Int_From_Bytes(BYTE sector[], int offset, int size) {
    int64_t k = 0;
    memcpy(&k, sector + offset, size);
    return k;
}
```

sector:

Tiếp theo là hàm quan trọng nhất để đọc các sector:

int FAT32::ReadSector(LPCWSTR drive, int readPoint, BYTE sector[])

```

int FAT32::ReadSector(LPCWSTR drive, int readPoint, BYTE sector[]) {
    DWORD bytesRead;

    _device = CreateFile(drive,
        GENERIC_READ,
        FILE_SHARE_READ | FILE_SHARE_WRITE,
        NULL,
        OPEN_EXISTING,
        0,
        NULL);

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

    SetFilePointer(_device, readPoint, NULL, FILE_BEGIN);

    if (!ReadFile(_device, sector, BYTES_READ, &bytesRead, NULL))
    {
        return 1;
    }
    else {
        return 0;
    }
}

```

Để đọc được sector ở hàm main ta dùng như sau:

```

wstring driveName;
cout << "Nhập tên ổ đĩa: ";
wcin >> driveName;
driveName = L"\\\\\\\\" + driveName + L".";
LPCWSTR drive = driveName.c_str();

FAT32 fat32(drive);

fat32.printInfoFAT32();

```

In ra kết quả với hàm: void FAT32::printInfoFAT32()

```

void FAT32::printInfoFAT32() {
    cout << "\t\t ----- THÔNG TIN TRONG BOOTSECTOR FAT32 -----" << endl;
    cout << " So bytes/sector: " << _bytePerSector << endl;
    cout << " So sector/cluster: " << _sectorPerCluster << endl;
    cout << " So sector Reserved: " << _rsvdSecCnt << endl;
    cout << " So bảng FAT: " << _numberOfFAT << endl;
    cout << " So sector của một bảng FAT: " << _FATSector32 << endl;
    cout << " Tổng sector volume: " << _totSec32 << endl;
    cout << " Địa chỉ sector đầu tiên bảng FAT: " << _rsvdSecCnt << endl;
    cout << " Địa chỉ sector đầu tiên data: " << _rsvdSecCnt + _numberOfFAT * _FATSector32 << endl;
    cout << "\t\t ----- CẤY THU MỤC -----" << endl;
}

```

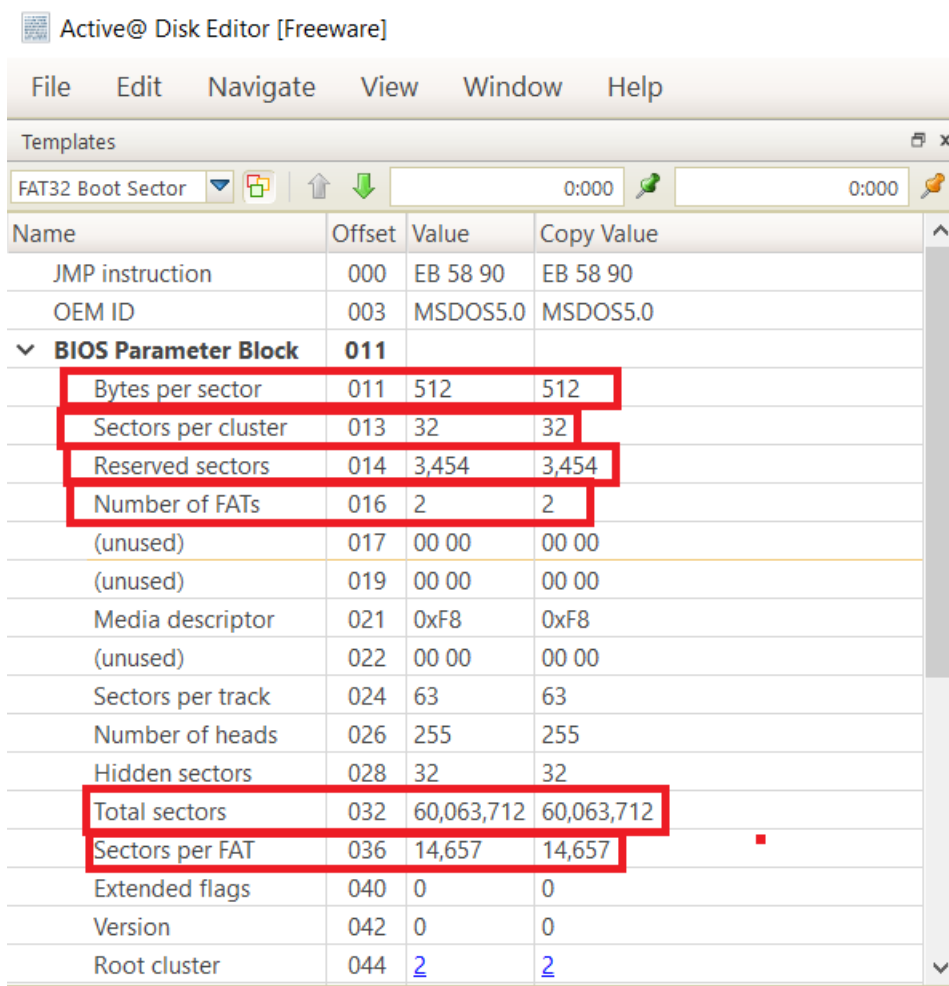
Kết quả:

```

Nhap ten o dia: e
----- THÔNG TIN TRONG BOOTSECTOR FAT32 -----
So bytes/sector:          512
So sector/cluster:        32
So sector Reserved:       3454
So bang FAT:              2
So sector của một bang FAT: 14657
Tong sector volume:       60063712
Dia chỉ sector đầu tiên bang FAT: 3454
Dia chỉ sector đầu tiên data: 32768

```

Cuối cùng kiểm tra lại với Disk Editor:



2. Hiện thị cây thư mục FAT32

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

```
//Lay ra cluster dau tien cua root directory
int FAT32::get_Root_of_Cluster() {
    return _rootClus;
}

//Tim sector dau tien cua cluster
int FAT32::find_First_SectorOfCluster(int cluster) {
    return (cluster - 2) * _sectorPerCluster + _rsvdSecCnt + _numberOfFAT * _FATSector32;
}
```

Các hàm tìm cluster đầu tiên của cây và tìm sector đầu tiên của cluster

Sau đó ta sẽ tìm thêm các sector tương ứng thông qua hàm:

vector<int> FAT32::getClusters(int firstCluster)

```
vector<int> FAT32::getClusters(int firstCluster) {
    vector<int> fileClusters;

    BYTE sector[BYTES_READ];
    int readPoint = _rsvdSecCnt * _bytePerSector;

    SetFilePointer(_device, readPoint, NULL, FILE_BEGIN);
    ReadSector(_drive, readPoint, sector);
    readPoint = readPoint + BYTES_READ;

    int clusterValue = firstCluster;
    do {
        fileClusters.push_back(clusterValue);
        clusterValue = get_Int_From_Bytes(sector, (4 * clusterValue) % 512, 4);

        if (clusterValue > readPoint / 4)
        {
            SetFilePointer(_device, readPoint, NULL, FILE_BEGIN);
            ReadSector(_drive, readPoint, sector);
            readPoint = readPoint + BYTES_READ;
        }
    } while (clusterValue != STOP_CLUSTER);

    return fileClusters;
}
```

Cuối cùng ta lấy thông tin tập tin


```

void FAT32::get_File_Info(BYTE sector[], int firstCluster)
{
    vector<int> fileClusters = getClusters(firstCluster);
    vector<int> fileSectors = get_Sector_Of_FileSectors(fileClusters);
    printTab();
    cout << "+ Cluster bat dau: " << firstCluster << endl;
    printTab();
    cout << "+ Chiem cac cluster: ";

    for (unsigned int i = 0; i < fileClusters.size(); i++)
    {
        cout << fileClusters[i];
    }
    cout << endl;
    printTab();
    cout << "+ chiem cac sector: ";
    for (unsigned int i = 0; i < fileSectors.size(); i++)
    {
        cout << fileSectors[i] << " ";
    }
    cout << endl;
}

```

Và đọc nội dung tập tin:

```

void FAT32::ReadData(wstring fileExtension, int firstCluster) {
    transform(fileExtension.begin(), fileExtension.end(), fileExtension.begin(), ::toupper);
    printTab();
    cout << "+ Noi dung :";
    if (wcscmp(fileExtension.c_str(), L"TXT") == 0 || wcscmp(fileExtension.c_str(), L"txt") == 0) {
        if (firstCluster != 0) {
            vector<int> fileClusters = getClusters(firstCluster);
            vector<int> fileSectors = get_Sector_Of_FileSectors(fileClusters);

            for (unsigned int i = 0; i < fileSectors.size(); i++) {
                BYTE sectorFile[BYTES_READ];

                int readPointFile = fileSectors[i] * _bytePerSector;

                SetFilePointer(_device, readPointFile, NULL, FILE_BEGIN);
                ReadSector(_drive, readPointFile, sectorFile);

                for (int j = 0; j < BYTES_READ && sectorFile[j] != '\0'; j += 1)
                    cout << sectorFile[j];
            }
            cout << endl;
        }
        else
            cout << "Dungphan mem khac de doc file khac .txt\n";
        cout << endl;
    }
}

```

ở hàm main ta xây dựng như sau:

```

int rootClus = fat32.get_Root_of_Cluster();
fat32.getDirectory(rootClus);

```

Kết quả:

```
Ten:New folder
+ Loại tập tin: Thu mục
+ Cluster bắt đầu: 6
+ Chiem cac cluster: 6
+ chiem cac sector: 32896 32897 32898 32899 32900 32901 32902 32903 32904 32905 32906 32907 32908 32909 32910 32911 32912 32913 32914
32915 32916 32917 32918 32919 32920 32921 32922 32923 32924 32925 32926 32927
  Ten:New folder
  + Loại tập tin: Thu mục
  + Cluster bắt đầu: 9
  + Chiem cac cluster: 9
  + chiem cac sector: 32992 32993 32994 32995 32996 32997 32998 32999 33000 33001 33002 33003 33004 33005 33006 33007 33008 33009
9 33010 33011 33012 33013 33014 33015 33016 33017 33018 33019 33020 33021 33022 33023
    Ten:New Text Document.txt
    + Loại tập tin: Tập tin
    + Kích thước có 0 byte
    + Nội dung :

    Ten:KHTN.TXT
    + Loại tập tin: Tập tin
    + Cluster bắt đầu: 12
    + Chiem cac cluster: 12
    + chiem cac sector: 33088 33089 33090 33091 33092 33093 33094 33095 33096 33097 33098 33099 33100 33101 33102 33103 33104 33105 33106
33107 33108 33109 33110 33111 33112 33113 33114 33115 33116 33117 33118 33119
    + Kích thước có 2560 byte
    + Nội dung :hcmus
```

III.Phân vùng NTFS

1.Đọc thông tin chi tiết của phân vùng NTFS

NTFS																ASCII		Unicode	
Offset	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15			
00000000000	EB	52	90	4E	54	46	53	20	20	20	20	00	02	08	00	00	ER.	NTFS† ..
00000000016	00	00	00	00	00	F8	00	00	3F	00	FF	00	20	00	00	00ø...?..ý?ý .
00000000032	00	00	00	00	80	00	00	00	DF	7F	94	03	00	00	00	00ß.....	Δ..
00000000048	00	00	0C	00	00	00	00	00	02	00	00	00	00	00	00	00
00000000064	F6	00	00	00	01	00	00	00	FC	31	5F	68	76	5F	68	B8	ö.....ül hv h,		ö.....
00000000080	00	00	00	00	FA	33	C0	8E	D0	BC	00	7C	FB	68	C0	07ú3Ä.Đ%.. úhÀ.	
00000000096	1F	1E	68	66	00	CB	88	16	0E	00	66	81	3E	03	00	4E	..hf.È....f.>..N		f.....!
00000000112	54	46	53	75	15	B4	41	BB	AA	55	CD	13	72	0C	81	FB	TFSu.´A»ªUí.r...û	¿
00000000128	55	AA	75	06	F7	C1	01	00	75	03	E9	DD	00	1E	83	EC	Uªu.÷Á...u.éý...ì		..´...ª.
00000000144	18	68	1A	00	B4	48	8A	16	0E	00	8B	F4	16	1F	CD	13	..h..´H....ô...í.	
00000000160	9F	83	C4	18	9E	58	1F	72	E1	3B	06	0B	00	75	DB	A3	..Ä..X.rá;...uŸ	
00000000176	0F	00	C1	2E	0F	00	04	1E	5A	33	DB	B9	00	20	2B	C8	..Ä.....Z3Ů¹. +È		...ß....
00000000192	66	FF	06	11	00	03	16	0F	00	8E	C2	FF	06	16	00	E8	fý.....Âý...è	
00000000208	4B	00	2B	C8	77	EF	B8	00	BB	CD	1A	66	23	C0	75	2D	K.+Èwì...»í.f#Au-		K... ..
00000000224	66	81	FB	54	43	50	41	75	24	81	F9	02	01	72	1E	16	f.ûTCPAu\$.ù...r..	´..
00000000240	68	07	BB	16	68	52	11	16	68	09	00	66	53	66	53	66	h.».hR...h..fSfSf		û.....
00000000256	55	16	16	16	68	B8	01	66	61	0E	07	CD	1A	33	C0	BF	U...h...fa...í.3Ä	
00000000272	0A	13	B9	F6	0C	FC	F3	AA	E9							1E	..´ò.úóªép...f`.	ó.Š
00000000288	06	66	A1	11	00	66	03	06	1C	00	1E	66	68	00	00	00	.fj...f.....fh...	h.

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.

Xây dựng hàm kiểm tra đọc thành công sector hay không:

```
int ReadSector(LPCWSTR drive, int readPoint, BYTE*& sector)
```

```

int ReadSector(LPCWSTR drive, int readPoint, BYTE*& sector)
{
    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
    {
        cout << "CreateFile : " << GetLastError() << endl;
        cout << endl;
        return 0;
    }

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

    if (!ReadFile(device, sector, 512, &bytesRead, NULL))
    {
        cout << "ReadFile : " << GetLastError() << endl;
        return 0;
    }
    else
    {
        cout << "Success !!!" << endl;
        system("pause");

        cout << endl;
        return 1;
    }
}

```

Tương tự vs FAT32 ta cũng có hàm đọc sector:

```

void ReadSect2(LPCWSTR disk, BYTE*& DATA, unsigned int _nsect)
{
    DWORD dwBytesRead(0);

    HANDLE hFloppy = NULL;
    hFloppy = CreateFile(disk, // Floppy 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 (hFloppy != NULL)
    {
        LARGE_INTEGER li;
        li.QuadPart = _nsect * 512;
        SetFilePointerEx(hFloppy, li, 0, FILE_BEGIN);
        // Read the boot sector
        if (!ReadFile(hFloppy, DATA, 512, &dwBytesRead, NULL))
        {
            cout << "Error in reading floppy disk" << endl;
        }

        CloseHandle(hFloppy);
        // Close the handle
    }
}

```

Và các hàm hỗ trợ như lấy “number” byte từ vị trí “offset”:

```
int64_t Get_Bytes(BYTE* sector, int offset, int number);
```

, Chuyển "number" bytes DATA từ vị trí "offset" thành string:

```
string toString(BYTE* DATA, int offset, int number);
```

, Chuyển

hệ 10 sang hệ 2:

```
string toBinary(int n);
```

In ra kết quả thông qua:

```
void Read_BPB(BYTE* sector, LPCWSTR disk)
{
    unsigned int bytes_per_sector = Get_Bytes(sector, 0x0B, 2); // Bytes Per Sector
    unsigned int sectors_per_cluster = Get_Bytes(sector, 0x0D, 1); // Sectors Per Cluster
    unsigned int sectors_per_track = Get_Bytes(sector, 0x18, 2); // Sectors Per Track
    unsigned int total_sectors = Get_Bytes(sector, 0x28, 8); // Total Sectors
    unsigned int MFTStart = Get_Bytes(sector, 0x30, 8); // Cluster start of MFT
    unsigned int MFTMirrorStart = Get_Bytes(sector, 0x38, 8); // Cluster start of MFTMirror

    cout << endl;
    const int SPACE = 23;
    cout << " _____" << endl;
    cout << "|Bytes per sector :          |" << setw(SPACE) << bytes_per_sector << " |" << endl;
    cout << "|Bytes Per Sector :         |" << setw(SPACE) << bytes_per_sector << " |" << endl;
    cout << "|Sectors Per Cluster :      |" << setw(SPACE) << sectors_per_cluster << " |" << endl;
    cout << "|Sectors Per Track :       |" << setw(SPACE) << sectors_per_track << " |" << endl;
    cout << "|Total Sectors :           |" << setw(SPACE) << total_sectors << " |" << endl;
    cout << "|Cluster start of MFT :     |" << setw(SPACE) << MFTStart << " |" << endl;
    cout << "|Cluster start of MFTMirror : |" << setw(SPACE) << MFTMirrorStart << " |" << endl;
    cout << "|_____|" << " |" << endl;
    cout << endl;

    // Đọc $MFT Entry
    //read_MFT(MFTStart, sectors_per_cluster, disk);
}
```

ở hàm main ta cài đặt xử lý ổ đĩa và menu để chương trình chạy liền mạch hơn.

Kết quả:

```
|Bytes per sector :          |512|
|Bytes Per Sector :         |512|
|Sectors Per Cluster :      |8  |
|Sectors Per Track :       |63 |
|Total Sectors :           |60063711|
|Cluster start of MFT :     |786432|
|Cluster start of MFTMirror : |2  |
|_____|
```

Press any key to continue

Ta cũng kiểm tra lại tương tự FAT32 với Disk Editor:

NTFS Boot Sector				0:000	0:000
Name	Offset	Value	Copy Value		
JMP instruction	000	EB 52 90	EB 52 90		
OEM ID	003	NTFS	NTFS		
▼ BIOS Parameter Block	011				
Bytes per sector	011	512	512		
Sectors per cluster	013	8	8		
Reserved sectors	014	0	0		
(always zero)	016	00 00 00	00 00 00		
(unused)	019	00 00	00 00		
Media descriptor	021	248	248		
(unused)	022	00 00	00 00		
Sectors per track	024	63	63		
Number of heads	026	255	255		
Hidden sectors	028	32	32		
(unused)	032	00 00 00 ...	00 00 00 00		
Signature	036	80 00 00 ...	80 00 00 00		
Total sectors	040	60,063,711	60,063,711		
\$MFT cluster number	048	786,432	786,432		
\$MFTMirr cluster nu...	056	2	2		
Clusters per File Re...	064	246	246		
Bookmarks					

2. Hiển thị cây thư mục NTFS

Ta đọc thông tin của entry như sau

```

int Read_Entry_INFORMATION(BYTE* Entry, int start)
{
    int64_t status = Get_Bytes(Entry, start + 56, 4);

    string bin = toBinary(status);
    for (int i = bin.length() - 1; i >= 0; i--)
    {
        int n = bin.length();
        if (bin[i] == '1')
        {
            if (i == n - 2)
            {
                // Hidden
                return -1;
            }
            if (i == n - 3)
            {
                // File System
                return -1;
            }
        }
    }

    cout << "Attribute $STANDARD_INFORMATION" << endl;

    // Byte thứ 4 đến 7, Kích thước của attribute
    int size = Get_Bytes(Entry, start + 4, 4);
    cout << "\t- Length of attribute (include header): " << size << endl;
    cout << "\t- Status Attribute of File: " << bin << endl;
    for (int i = bin.length() - 1; i >= 0; i--)
    {
        int n = bin.length();
        if (bin[i] == '1')
        {
            if (i == n - 1)
                cout << "\t\t => Read Only" << endl;
            if (i == n - 4)
                cout << "\t\t => Vollabel" << endl;
            if (i == n - 5)
                cout << "\t\t => Directory" << endl;
            if (i == n - 6)
                cout << "\t\t => Archive" << endl;
        }
    }

    cout << endl;

    // trả về size của attribute
    return size;
}

```

```

int Read_Entry_FILE_NAME(BYTE* Entry, int start, int ID)
{
    cout << "Attribute $FILE_NAME" << endl;
    int size = Get_Bytes(Entry, start + 4, 4);
    cout << "\t- Length of attribute (include header): " << size << endl;
    int parent_file = Get_Bytes(Entry, start + 24, 6);
    cout << "\t- Parent file: " << parent_file << endl;

    parentID.push_back(parent_file);

    int lengthName = Get_Bytes(Entry, start + 88, 1);
    cout << "\t- Length of name file: " << lengthName << endl;
    string name = toString(Entry, start + 90, lengthName * 2);
    cout << "\t- Name of file: " << name << endl;

    //Lấy đuôi mở rộng
    string exts = "";
    for (int i = name.length() - 1; i >= name.length() - 5; i--)
    {
        if (name[i] == '.')
            break;
        exts += name[i];
    }
    reverse(exts.begin(), exts.end());

    //Hỗ trợ đọc file
    if (exts == "doc" || exts == "docx")
        cout << "\t\t\t => Use Microsoft Office Word to open!\n";
    if (exts == "ppt" || exts == "pptx")
        cout << "\t\t\t => Use Microsoft Office PowerPoint to open!\n";
    if (exts == "xls" || exts == "xlsx")
        cout << "\t\t\t => Use Microsoft Office Excel to open!\n";
    if (exts == "sln" || exts == "cpp" || exts == "js" || exts == "html" || exts == "css")
        cout << "\t\t\t => Use Microsoft Visual Studio to open!\n";
    if (exts == "pdf")
        cout << "\t\t\t => Use Foxit PDF Reader or Web Browsers (Edge, Chrome, ...) to open!\n";

    if (exts == "txt") chk = true;
    nameFile.push_back(name);
    cout << endl;

    return size;
}

```

```

void Read_Entry_DATA(BYTE* Entry, int start)
{
    cout << "Attribute $DATA" << endl;
    int size = Get_Bytes(Entry, start + 4, 4);
    cout << "\t- Length of attribute (include header): " << size << endl;
    int sizeFile = Get_Bytes(Entry, start + 16, 4);
    cout << "\t- Size of file: " << sizeFile << endl;

    int type = Get_Bytes(Entry, start + 8, 1);
    if (type == 0 && chk == true)
    {
        cout << "\t\t=> Resident" << endl;
        int cont_Size = Get_Bytes(Entry, start + 16, 4);
        int cont_Start = Get_Bytes(Entry, start + 20, 2);
        string content = toString(Entry, start + cont_Start, cont_Size);
        cout << endl;
        cout << "Content: " << endl << content << endl;
    }
    else
        cout << "\t\t=> Non-resident" << endl;
    cout << endl;
}

```

MFT bản chất là một tập tin, do vậy cũng có một MFT entry mô tả cho chính nó, đó chính là MFT entry đầu tiên trong MFT, có tên là \$MFT. Từ đó ta sẽ tìm được thông tin cho tất cả các thư mục con thông qua \$MFT. Xây dựng hàm đọc \$MFT Entry:


```

void read_MFT(unsigned int MFTStart, unsigned int sectors_per_cluster, LPCWSTR disk)
{
    BYTE* MFT = new BYTE[512];
    MFTStart *= sectors_per_cluster;
    ReadSect2(disk, MFT, MFTStart);

    // INFORMATION
    int Entry_in4 = Get_Bytes(MFT, 0x014, 2);

    int len_in4 = Get_Bytes(MFT, 0x048, 4);

    // FILE NAME
    int Entry_Name = Entry_in4 + len_in4;

    int len_Name = Get_Bytes(MFT, 0x09C, 4);

    // DATA
    int tmp = Get_Bytes(MFT, 0x108, 4);
    int Entry_Data = 0;
    if (tmp == 64) {
        Entry_Data = Entry_Name + len_Name + Get_Bytes(MFT, 0x10C, 4);

        int len_data = Get_Bytes(MFT, 0x134, 4);
    }
    else {
        Entry_Data = Entry_Name + len_Name;

        int len_data = Get_Bytes(MFT, 0x10C, 4);
    }

    // main DATA
    unsigned int len_MFT = MFTStart + (Get_Bytes(MFT, Entry_Data + 24, 8) + 1) * 8;

    // xử lí cây thư mục
    directory_Tree(len_MFT, MFTStart, disk);
}

```

Kết quả:

-----MENU-----

1. View the information
2. Display directory tree
3. Exit

Choose : 2

ID File: 39

Attribute \$STANDARD_INFORMATION

- Length of attribute (include header): 96
- Status Attribute of File: 0

Attribute \$FILE_NAME

- Length of attribute (include header): 120
- Parent file: 5
- Length of name file: 14
- Name of file: HCM - Copy (2)

Attribute \$DATA

- Length of attribute (include header): 80
- Size of file: 48
- => Non-resident

ID File: 40

Attribute \$STANDARD_INFORMATION

- Length of attribute (include header): 96
- Status Attribute of File: 0

Attribute \$FILE_NAME

- Length of attribute (include header): 96
- Parent file: 5
- Length of name file: 3
- Name of file: HCM

Attribute \$DATA

- Length of attribute (include header): 184
- Size of file: 152
- => Non-resident

```

ID File: 41
Attribute $STANDARD_INFORMATION
  - Length of attribute (include header): 96
  - Status Attribute of File: 100000
    => Archive

Attribute $FILE_NAME
  - Length of attribute (include header): 112
  - Parent file: 40
  - Length of name file: 8
  - Name of file: khtn.txt

Attribute $DATA
  - Length of attribute (include header): 32
  - Size of file: 5
    => Non-resident

-----

ID File: 42
Attribute $STANDARD_INFORMATION
  - Length of attribute (include header): 96
  - Status Attribute of File: 0

Attribute $FILE_NAME
  - Length of attribute (include header): 112
  - Parent file: 5
  - Length of name file: 10
  - Name of file: HCM - Copy

Attribute $DATA
  - Length of attribute (include header): 80
  - Size of file: 48
    => Non-resident

-----

                        Directory Tree
HCM - Copy (2)
HCM
    khtn.txt
HCM - Copy
Press any key to continue . . .

```

IV. Các nguồn tham khảo

- Tài liệu trên modle của thầy Lê Viết Long-Giảng Viên Trường Đại Học Khoa Học Tự Nhiên-Môn Điều Hành-20_1.
- Các video hướng dẫn trên youtube.
- <https://www.ntfs.com/ntfs-partition-boot-sector.htm>
- <https://github.com/DeDf/ParseNTFS/blob/master/ntfs.h>
- <https://lazytrick.wordpress.com/2015/12/27/khai-quat-ve-fat/?fbclid=IwAR0apR-RnhJ2a9xTbV5X5avrJCrvpDHFTdqqkvAyNyrMou9Z72VRlw-cTt8>

- <https://www.codeguru.com/cplusplus/long-file-name-lfn-entries-in-the-fat-root-directory-of-floppy-disks/>