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

Môn học: Hệ điều hành

ĐỒ ÁN 1

Quản lý hệ thống tập tin trên Windows

Lớp: 22CLC07
Thành viên nhóm:

1. Phạm Thế Bằng - MSSV: 22127025
2. Phan Thành Quang Huy - MSSV: 22127162
3. Nguyễn Hoàng Khang - MSSV: 22127178
4. Võ Phương Nam - MSSV: 22127289
5. Nguyễn Minh Trực - MSSV: 22127428

Thành phố Hồ Chí Minh, 2024

Mục lục

1 Bảng phân công công việc

STT	Họ tên	MSSV	Công việc
1	Phạm Thế Bằng	22127025	Viết main và UI
2	Phan Thành Quang Huy	22127162	FAT32
3	Nguyễn Hoàng Khang	22127178	FAT32
4	Võ Phương Nam	22127289	NTFS
5	Nguyễn Minh Trực	22127428	Viết báo cáo

2 Đánh giá mức độ hoàn thành

2.1 Trên từng yêu cầu

STT	Yêu cầu	Hoàn thành
1.1	Đọc thông tin chi tiết của phân vùng FAT32	100%
1.2	Đọc thông tin chi tiết của phân vùng NTFS	100%
2.1	Hiển thị thông tin cây thư mục của phân vùng FAT32	100%
2.2	Hiển thị thông tin cây thư mục của phân vùng NTFS	100%

2.2 Toàn bộ đồ án

Hoàn thành tất cả các yêu cầu được giao.

3 Danh sách các lớp và hàm sử dụng

3.1 FAT32

```
class FAT:
    def __init__(self, data):...

    def get_cluster_chain(self, starting_index: int) -> 'list[int]':...


class Attribute(Flag):
    READ_ONLY = auto()
    HIDDEN = auto()
    SYSTEM = auto()
    VOL_LABLE = auto()
    DIRECTORY = auto()
    ARCHIVE = auto()
```

```
class RDETentry:
    def __init__(self, data) -> None:...

    def parse_entry(self):...

    def set_date_time(self):...

    def extract_start_cluster_size(self):...

    def extract_long_name(self):...

    def is_active_entry(self) -> bool:...

    def is_directory(self) -> bool:...

    def is_archive(self) -> bool:
        return Attribute.ARCHIVE in self.attr
```

```
class RDET:
    def __init__(self, data: bytes) -> None:...

    def get_full_entry_name(self) -> list[RDETentry]:...

    def get_active_entries(self) -> 'list[RDETentry]':...

    def find_entry(self, name) -> RDETentry:...


class Fat32_Main:
    def __init__(self, volume_name) -> None:...

    def __str__(self) -> str:...

    def __del__(self):...

    def extract_boot_sector(self):...

    def convert_cluster_to_sector_index(self, index):...

    def get_all_cluster_data(self, cluster_index):...

    @staticmethod
    def isFAT32(volume_name):...

    def parsePath(self, path):...

    def visitDirectory(self, path) -> RDET:...

    def getCWD(self):...

    def getDirectory(self, path=""):...

    def changeDirectory(self, path=""):...

    def getText(self, path: str) -> str:...
```

3.2 NTFS

```python
class Attribute(Flag):
    READ_ONLY = auto()
    HIDDEN = auto()
    SYSTEM = auto()
    VOLLABLE = auto()
    DIRECTORY = auto()
    ARCHIVE = auto()


def getDatetime(timestamp):...


class MFTRecord:
    def __init__(self, data) -> None:...

    def isDirectory(self):...

    def isLeaf(self):...

    def isActive(self):...
```

```python
class MFTFile:
    def __init__(self, data: bytes) -> None:...


class NTFS:
    importantInfo = [
        "OEM ID",
        "Serial Number",
        "Bytes Per Sector",
        "Sectors Per Cluster",
        "Reserved Sectors",
        "No. Sectors In Volume",
        "First Cluster of $MFT",
        "First Cluster of $MFTMirr",
        "MFT record size"
    ]

    def __init__(self, name: str) -> None:...

    @staticmethod
    def isNTFS(name: str):...
```

```
    def findRecord(self, name: str):...

    def getRecords(self) -> 'list[MFTRecord]':...

    def parseData(self, start):...

    def parseFileName(self, start):...  # unicode

    def parseInfo(self, start):...


class DirectoryTree:
    def __init__(self, nodes: 'list[MFTRecord]') -> None:...

    def findRecord(self, name: str):...

    def getParentRecord(self, record: MFTRecord):...

    def getActiveRecords(self) -> 'list[MFTRecord]':...
```

```
    @staticmethod
    def isNTFS(name: str):...

    def extractBootSector(self):...

    def parsePath(self, path):...

    def visitDir(self, path) -> MFTRecord:...

    def getDirectory(self, path=""):...

    def changeDirectory(self, path=""):...

    def getCWD(self):...

    def getText(self, path: str) -> str:...

    def __str__(self) -> str:...

    def __del__(self):...
```

3.3 UI

```
class UI(cmd.Cmd):
    intro = ("COMMANDS LIST.\n"
             "1. Type 'info' to print information of volume.\n"
             "2. Type 'tree' to print root directory tree.\n"
             "3. Type 'data + filename' to retrieve file content.\n"
             "     - First, you have to be into the directory that contains this file.\
             "4. Type 'cd + directory' to change the current directory.\n"
             "5. Type 'exit' to quit the program.\n")

    def __init__(self, volume: Union[Fat32_Main, NTFS]) -> None:...

    def updateDirectory(self):...

    def do_cd(self, arg):...

    def do_tree(self, arg):...

    def do_data(self, arg):...

    def do_info(self, arg):...

    def do_exit(self, arg):...

    def close(self):...
```

4 Các bước thực hiện

4.1 FAT32

4.1.1 Vùng Boot Sector

Đọc byte ngược từ dưới lên. (Ví dụ: 2 byte là 00, 02 $\Longrightarrow 0200_h = 512$ byte)

- Số byte cho 1 sector: Đọc 2 byte từ offset 0B
- Số sector cho 1 cluster (S_C): Đọc 1 byte từ offset 0D
- Số sector cho vùng Boot Sector (S_B): Đọc 2 byte từ offset 0E
- Sector đầu tiên của FAT (S_B): Đọc 2 byte từ offset 0E
- Số bảng FAT (N_F): Đọc 1 byte từ offset 10
- Tổng số sector trên đĩa (S_V): Đọc 4 byte từ offset 20
- Kích cỡ FAT (S_F): Đọc 4 byte từ offset 24
- Loại FAT: Đọc 8 byte từ offset 52
- Sector đầu tiên của RDET (S_{RDET}): Đọc 4 byte từ offset 2C
- Sector đầu tiên của vùng data: $S_B + N_F * S_F + S_{RDET}$

4.1.2 Bảng cây thư mục gốc (RDET)

- Tên:

  + Tên thư mục/ tập tin: Đọc 8 byte từ offset 00. Sau đó đổi sang mã ASCII ta được tên thư mục/ tập tin

  + Phần mở rộng: Đọc 3 byte từ offset 08. Sau đó đổi sang mã ASCII ta được tên phần mở rộng

  Lưu ý: Nếu tổng số kí tự của tên và phần mở rông lớn hơn 11 kí tự sẽ xuất hiện thêm entry phụ (mỗi entry phụ chỉ lưu được 26 kí tự đối với mã ASCII $\longrightarrow$ tên càng dài thì cần càng nhiều entry phụ)

- Trạng thái: Đọc 1 byte tại offset 0B. Đổi giá trị vừa đọc được sang hệ nhị phân nếu bit 5 được bật $\Longrightarrow$ tập tin, nếu bit 5 không được bật $\Longrightarrow$ thư mục

  Ví dụ: 1 byte vừa đọc là 20 $\Longrightarrow 20_h = 00100000_b \longrightarrow$ bit thứ 5 được bật (bit thứ 5 có giá trị bằng 1) $\longrightarrow$ tập tin

- Cluster bắt đầu:

  + 2 byte cao: Đọc 2 byte từ offset 14

  + 2 byte thấp: Đọc 2 byte từ offset 1A

  + Gộp 2 byte cao và 2 byte thấp theo thứ tự đọc ban đầu $\longrightarrow$ đổi sang hệ thập phân $\longrightarrow$ tra bảng FAT $\longrightarrow$ thư mục/ tập tin ở cluster nào và mỗi cluster chiếm S_C sector theo công thức:

  $$i = S_B + N_F * S_F + S_{RDET} + S_C * (k - 2)$$

  với k là giá trị cluster sau khi tra bảng FAT, $(k-2)$ là do vùng dữ liệu bắt đầu từ 2

  Ví dụ: 2 byte cao đọc được là 00, 00 và 2 byte thấp đọc được là 00, 0D; 2 sector/cluster

  $\longrightarrow 0000000D_h = 13$

  $\longrightarrow$ Tra bảng FAT ta được: 13, 14

  $\longrightarrow$ Chiếm các sector: i, $(i+1)$, $(i+2)$, $(i+3)$ với $i = S_B + N_F * S_F + S_{RDET} + 2 * (13 - 2)$

- Kích cỡ: Đọc 4 byte từ offset 1C (đọc byte ngược từ dưới lên tương tự như ở trên)

4.1.3 Bảng thư mục con (SDET)

Tương tự như RDET nhưng đầu mỗi SDET luôn có 2 entry '.' và '..' ở đầu bảng mô tả về chính thư mục này và thư mục cha của nó.

4.1.4 Bảng FAT

- Có 1 hay nhiều bảng FAT
- Mỗi bảng chứa thông tin trạng thái của các cluster trong vùng data. Mỗi entry chứa thông tin trạng thái của cluster k:

  + Trống (FREE): giá trị của cluster k = 0

  + Hư (BAD): giá trị của cluster k = 0: giá trị của cluster k = 0FFFFFF7

  + Chứa nội dung tập tin/ thư mục (USED): giá trị của cluster k = 2...0FFFFFFF

  + Cuối tập tin (EOF): giá trị của cluster k = 0FFFFFFF

4.2 NTFS

4.2.1 Vùng Partition Boot Sector

Đọc byte ngược từ dưới lên. (Ví dụ: 2 byte là 00, 02 $\Longrightarrow 0200_h = 512$ byte)

- Kích thước của 1 sector: Đọc 2 byte từ offset 0B
- Số sector/cluster: Đọc 1 byte từ offset 0D
- Sector bắt đầu của ổ đĩa logic: Đọc 4 byte từ offset 1C
- Số sector của ổ đĩa: Đọc 8 byte từ offset 28
- Cluster bắt đầu của MFT: Đọc 8 byte từ offset 30
- Cluster bắt đầu của MFT dự phòng: Đọc 8 byte từ offset 38
- Kích thước 1 bản ghi trong MFT (đơn vị tính là byte): Đọc 1 byte từ offset 40
- Số cluster của Index Buffer: Đọc 1 byte từ offset 44

4.2.2 Vùng Master File Record (MFT)

- Phần Header: chứa thông tin quản lý của một record, 16 record đầu tiên bắt đầu bằng kí tự $ cho biết thông tin quản lý: vị trí boot sector, các cluster bị hư,... các record tiếp theo dùng để lưu trữ cho các tập tin/ thư mục
- Phần Attribute (thuộc tính):

  + Nếu thư mục có kích thước < 900 bytes sẽ nằm trọn trong 1 record. Ngược lại, thư mục sẽ chứa chỉ mục tới các cluster trên vùng dữ liệu để lưu phần nội dung tập tin hoặc bảng thư mục con (tổ chức theo B-tree)

- Đọc dữ liệu:

  + Header:

    - Một record bình thường sẽ bắt đầu bằng chuỗi "FILE"

    - Thuộc tính đầu tiên sẽ bắt đầu tại vị trí nào: Đọc 2 byte từ offset 14

- Xác định thư mục/ tập tin: Đọc 2 byte từ offset 16

+ Attribute:

- 4 byte đầu tại vị trí mà thuộc tính đầu tiên bắt đầu: cho biết thuộc tính chứa thông tin nhằm tương thích với hệ thống DOS; 4 byte kế tiếp sẽ cho biết kích thước của thuộc tính đầu tiên

- Thuộc tính tiếp theo sẽ bắt đầu tại: (vị trí thuộc tính đầu tiên bắt đầu) + (kích thước của thuộc tính đầu tiên); 4 byte bắt đầu của record này cho biết đây là thuộc tính chứa tên tập tin/ thư mục (Unicode), kích thước, ngày giờ tạo,...; 4 byte kế tiếp cho biết kích thước của thuộc tính này; đọc 2 byte từ vị trí byte thứ 13 cho biết thông tin chính của thuộc tính này nằm tại vị trí nào; vị trí thứ 40 của phần thông tin chính của record cho biết tên ngắn của thư mục, tập tin có bao nhiêu kí tự Unicode; phần tên bắt đầu tại vị trí thứ 42,...

Lưu ý: Khi đọc kích thước thuộc tính: 4 byte đọc được là 60, 00, 00, 00 $\longrightarrow 00000060_h$ $\Longrightarrow$ thuộc tính có kích thước 60_h bytes. Khi đọc vị trí: 2 byte đọc được là 18, 00 $\longrightarrow 0018_h$ $\Longrightarrow$ nằm tại vị trí 18_h của record

5 Demo chương trình

5.1 FAT32

```
COMMANDS LIST.
1. Type 'info' to print information of volume.
2. Type 'tree' to print root directory tree.
3. Type 'data + filename' to retrieve file content.
     - First, you have to be into the directory that contains this file.
4. Type 'cd + directory' to change the current directory.
5. Type 'exit' to quit the program.

┌──[E:\]
└──$ info
---VOLUME INFORMATION---
Volume name: E:
Bytes Per Sector: 512
Sectors Per Cluster: 8
Reserved Sectors: 2704
Number of FATs: 2
Sectors In Volume: 15424704
Sectors Per FAT: 15032
Starting Cluster of RDET: 2
FAT Name: FAT32
Starting Sector of Data: 32768
```

Thông tin volume

```
┌──[E:\]
└─$ tree
E:\
├── Test                                | DOT Entry                    | Size: 0
│   ├── theBang.txt                     | DOT Entry                    | Size: 65
│   ├── minhTruc.txt                    | DOT Entry                    | Size: 68
│   ├── QUANGHUY.TXT                    | DOT Entry                    | Size: 72
│   ├── hoangKhang.doc                  | DOT Entry                    | Size: 25088
│   ├── phuongNam.doc                   | DOT Entry                    | Size: 25088
│   └── DEMO.TXT                        | DOT Entry                    | Size: 88
├── 22CLC7.TXT                          | DOT Entry                    | Size: 48
└── OS.TXT                              | DOT Entry                    | Size: 42
┌──[E:\]
└─$ data 22clc7.txt
22127025
22127162
22127178
22127289
22127428
┌──[E:\]
└─$ cd Test
┌──[E:\Test]
└─$ data demo.txt
pham the bang
phan thanh quang huy
nguyen hoang khang
vo phuong nam
nguyen minh truc
┌──[E:\Test]
└─$ exit
```

Cây thư mục và Nội dung file .txt

5.2 NTFS

```
COMMANDS LIST.
1. Type 'info' to print information of volume.
2. Type 'tree' to print root directory tree.
3. Type 'data + filename' to retrieve file content.
     - First, you have to be into the directory that contains this file.
4. Type 'cd + directory' to change the current directory.
5. Type 'exit' to quit the program.

┌──[E:\]
└─$ info
---VOLUME INFORMATION---
Volume name: E:
OEM ID: NTFS
Serial Number: B8D0-6CBD
Bytes Per Sector: 512
Sectors Per Cluster: 8
Reserved Sectors: 0
No. Sectors In Volume: 15424703
First Cluster of $MFT: 786432
First Cluster of $MFTMirr: 2
MFT record size: 1024
```

Thông tin Volume

```
┌──[E:\]
└─$ tree
E:\
├── Test                                | DOT Entry                    | Size: 0
│   ├── theBang.txt                     | DOT Entry                    | Size: 67
│   ├── quangHuy.txt                    | DOT Entry                    | Size: 74
│   ├── minhTruc.txt                    | DOT Entry                    | Size: 70
│   ├── hoangKhang.doc                  | DOT Entry                    | Size: 25088
│   ├── phuongNam.doc                   | DOT Entry                    | Size: 25088
│   └── demo.txt                        | DOT Entry                    | Size: 90
├── 22clc7.txt                          | DOT Entry                    | Size: 50
└── os.txt                              | DOT Entry                    | Size: 42
┌──[E:\]
└─$ data 22clc7.txt
22127025
22127162
22127178
22127289
22127428

┌──[E:\]
└─$ cd Test
┌──[E:\Test]
└─$ data demo.txt
pham the bang
phan thanh quang huy
nguyen hoang khang
vo phuong nam
nguyen minh truc
```

Cây thư mục và Nội dung file .txt

6 Tham khảo

1. Trần Trung Dũng - Phạm Tuấn Sơn, Giáo trình Hệ điều hành, NXB Khoa học và Kỹ thuật.
2. Lê Viết Long, Silde Bài giảng HĐH.
3. https://github.com/PSDat123/FAT32-and-NTFS-explorer?fbclid=IwAR0abpKLUspuJAw-LEseC4YWqdW_x_gPFTNedNrdVRsjtvzpXUhfV98am0
4. https://www.win.tue.nl/~aeb/linux/fs/fat/fat-1.html?fbclid=IwAR3GGMz4x8cbs-DI00H9Mb
5. https://www.easeus.com/resource/fat32-disk-structure.html?fbclid=IwAR35RmcHZy-4ZfAp
6. https://www.youtube.com/watch?v=1JuYQgpbrW0&t=793s