



GV. LÊ VIẾT LONG

OPERATING SYSTEM

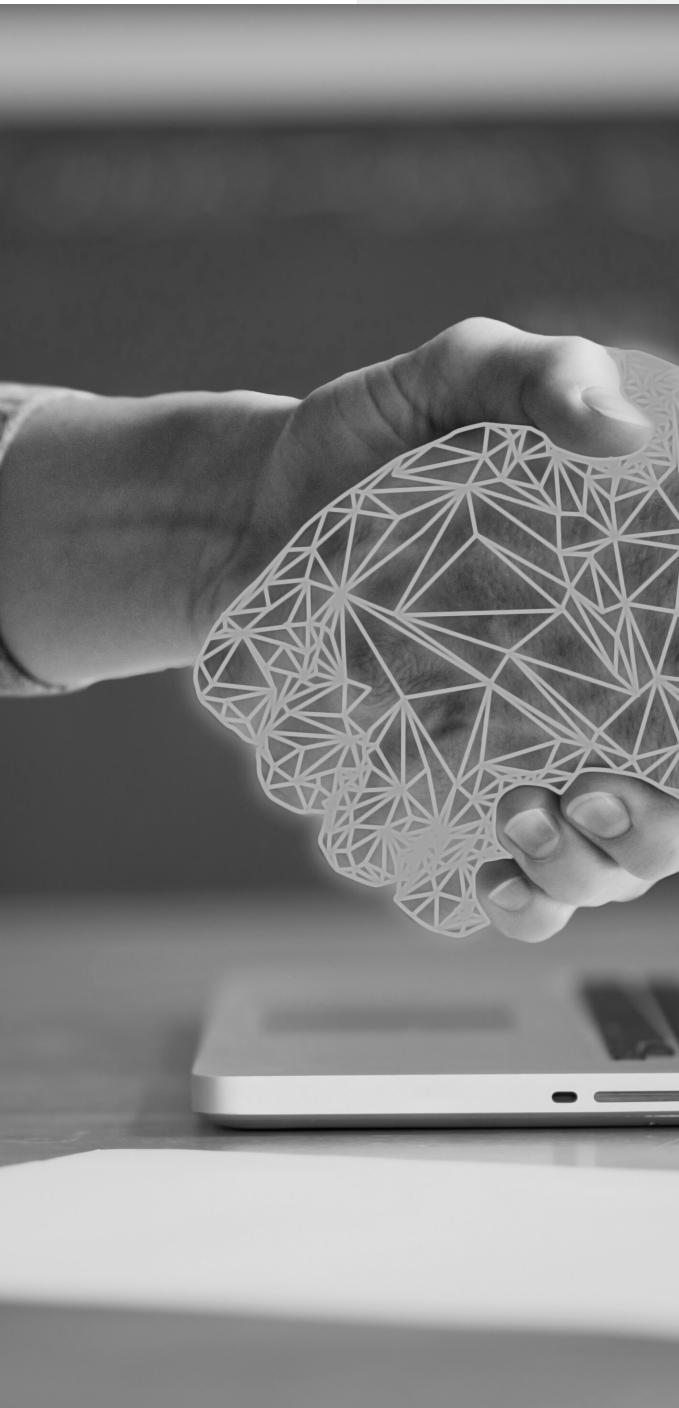
REPORT

2021

QUẢN LÝ HỆ THÔNG TẬP TIN
TRÊN WINDOWS

PREPARED BY
NHÓM HỆ ĐIỀU HÀNH

Table of contents



3

THÔNG TIN NHÓM

4

BẢNG PHÂN CÔNG CÔNG VIỆC

5

ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH

6

FAT32 & CÁC BƯỚC THỰC HIỆN

19

NTFS & CÁC BƯỚC THỰC HIỆN

36

DEMO CHƯƠNG TRÌNH

45

VIDEO DEMO

46

NGUỒN THAM KHẢO

47

LỜI CẢM ƠN

IMFORMATION ABOUT US

TEAM

DESCRIPTION

HUỲNH QUỐC DUY

19120494
LỚP 19CTT3

NGUYỄN QUANG ĐỊNH

19120478
LỚP 19CTT3

LÊ THÀNH LỘC

19120562
LỚP 19CTT3

PHẠM ĐỨC HUY

19120534
LỚP 19CTT3

OPERATION SYSTEM

Bảng Phân công công việc

Operation system



1



2

Huỳnh Quốc Duy - 19120494
Phân công công việc
Thiết kế Fat32 ,NTFS
Làm báo cáo

Lê Thành Lộc - 19120562
Thiết kế phần mềm đọc
Fat32 và NTFS
Làm báo cáo



3

Nguyễn Quang Định - 19120478
Thiết kế phần mềm đọc Fat32
và NTFS
Làm báo cáo



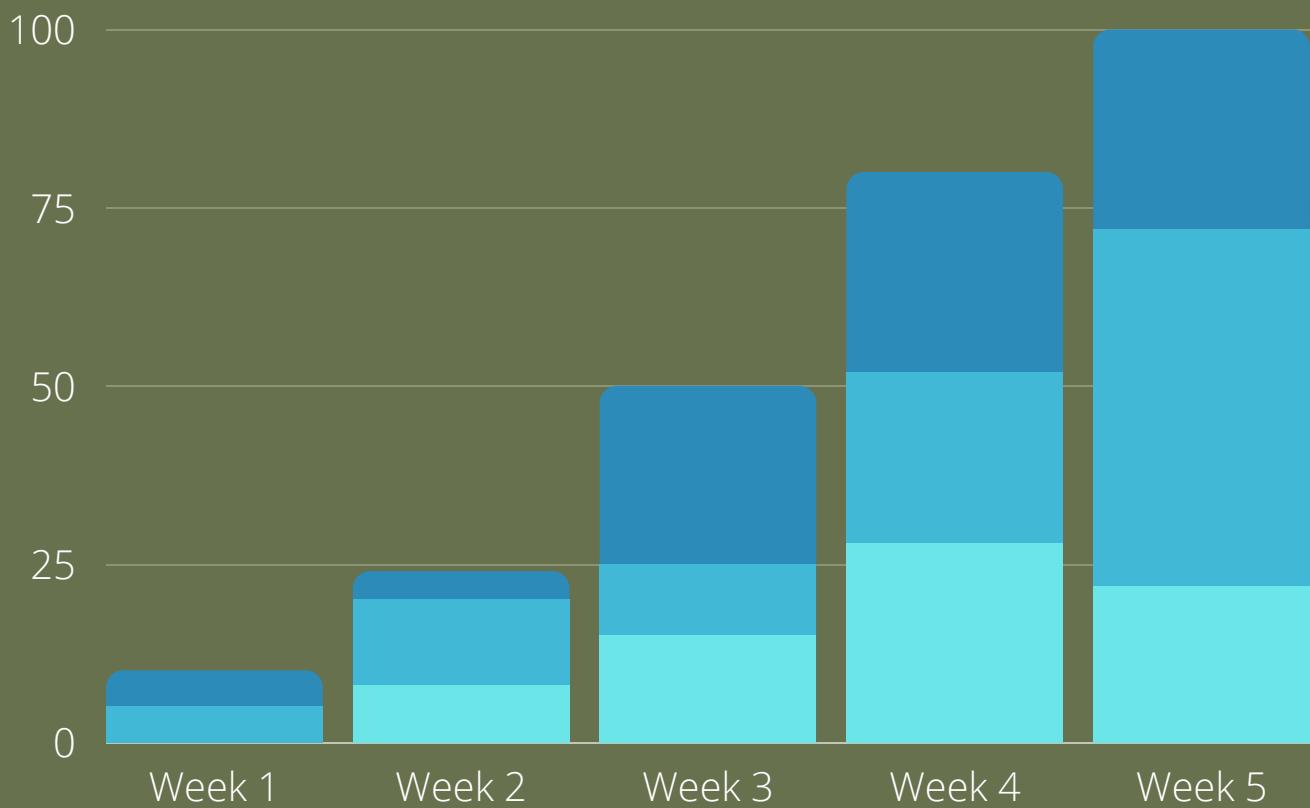
4

Phạm Đức Huy - 19120534
Thiết kế giao diện người dùng GUI
Quay video demo
Làm báo cáo

ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH

Dưới đây là biểu đồ cho thấy mức độ hoàn thành qua từng tuần của chúng em. Và cho đến cuối cùng thì sau khoảng 5 tuần làm đồ án thì chúng em đánh giá mức độ hoàn thành là 100% tương ứng với các yêu cầu mà đồ án này đưa ra.

Ngoài ra, tụi em còn cố gắng thiết kế và cải tiến một số yêu cầu. Đặc biệt ở phần code sử dụng ngôn ngữ Python và phần đồ họa sử dụng đồ họa của Python giúp cho người dùng dễ thao tác và linh hoạt. Cụ thể như thế nào mong thầy xem tiếp phần sau



OPERATION SYSTEM





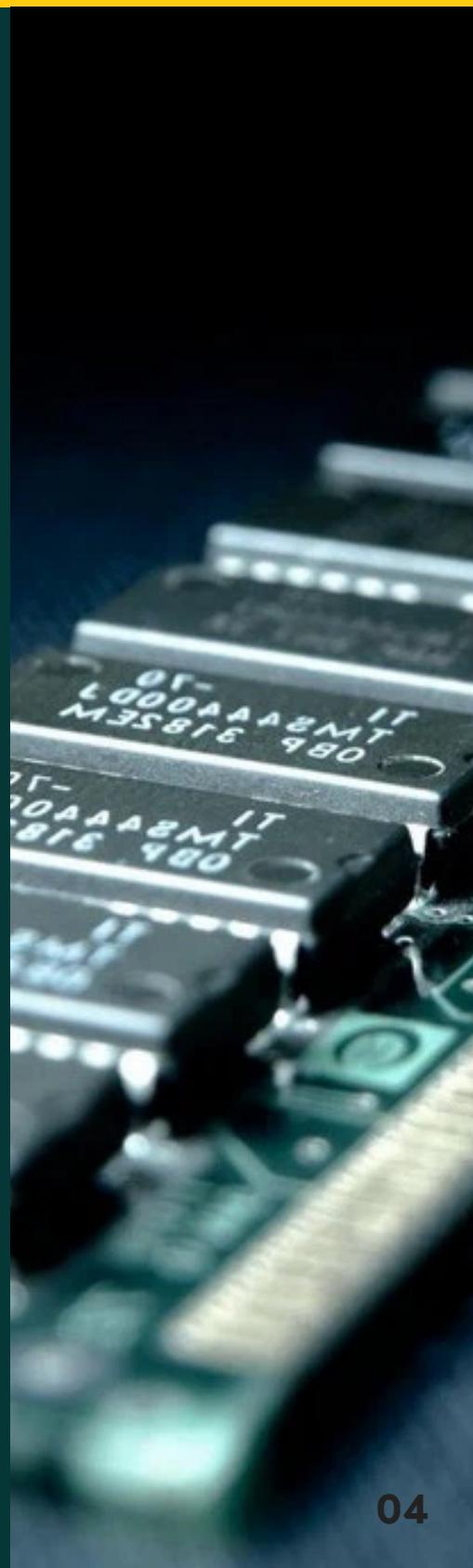
TỔNG QUAN VỀ FAT 32

FAT 32

OPERATING SYSTEM

About the **FAT32** **(File Allocation Table)**

FAT (File Allocation Table) là hệ thống quản lý tập tin được phát triển bởi Microsoft vào cuối những năm 1970 – đầu những năm 1980. Hệ điều hành MS-DOS của Microsoft là hệ điều hành đầu tiên sử dụng hệ thống FAT. Theo thời gian, hệ thống quản lý tập tin này luôn được nâng cấp để hỗ trợ nhiều loại đĩa có dung lượng lớn hơn. Ví dụ: đĩa cứng, đĩa USB, thẻ nhớ...v.v. Tất cả các hệ điều hành Windows và hầu hết các hệ điều hành Unix đều hỗ trợ hệ thống FAT.



THÀNH PHẦN

FAT 32

(File Allocation Table)



Cấu trúc của một ổ đĩa logic định dạng theo hệ thống FAT gồm ba vùng:

RESERVED

Reserved, chứa các thông tin mô tả của hệ thống quản lý tập tin. Kích thước của vùng này được lưu trong Boot Sector của ổ đĩa logic (trong VBR). Với FAT12, FAT16 vùng này thường là một sector, với FAT32 gồm nhiều sector.

FAT

chứa hai FAT Structure, vùng này bắt đầu ngay sau vùng Reserved, gồm: một FAT Structure chính, một FAT Structure dự phòng. Kích thước của vùng FAT được tính dựa vào số lượng FAT Structure nhân với kích thước của mỗi FAT Structure, hai thông tin này được lưu trong Boot Sector.

DATA

Data chứa các cluster lưu trữ nội dung của tập tin. Vùng này bắt đầu ngay sau vùng FAT. Kích thước được tính bằng tổng sector của ổ đĩa logic trừ đi sector bắt đầu của vùng Data. Tổng số sector của ổ đĩa được lưu trong Boot Sector. Root Directory (RDET) thường nằm ở phần đầu của vùng Data

OPERATION SYSTEM

MÔ TẢ CÁC BƯỚC THỰC HIỆN



NHÓM ...

ĐỌC THÔNG TIN PHÂN VÙNG CỦA FAT32

11	2	Kích thước 1 sector (byte) Ví dụ: 512, 1024, 2048 hoặc 4096	
13	1	Số sector cho 1 cluster	Sc
14	2	Số sector để dành (khác 0) (Số sector trước bảng FAT)	S _B
16	1	Số bảng FAT	N _F
17	2	số entry trong bảng RDET (đối với FAT32 thì có giá trị là 0)	N _{RDET}
28	4	Số sector ẩn trước Volume	
32	4	Tổng số sector trong volume	N _V
36	4	Số sector trong 1 bảng FAT	S _F
44	4	Chỉ số cluster đầu tiên của RDET	

Các trường cần thiết trong BootSector

Để đọc các thông tin chi tiết về Boot Sector của FAT32, đầu tiên ta đọc 512 byte đầu tiên. Vì FAT32 lưu trữ dữ liệu theo kiểu little-endian nên ta sẽ đọc lần lượt các byte như bảng trên rồi dùng hàm little_endian_to_integer để có được thông tin đúng



```
def little_endian_to_integer(data, offset, size):
    power = 0
    sum = 0
    for i in range(size):
        x = data[i+offset]
        for j in range(8):
            if x & 1:
                sum = sum + int(pow(2, power))
            x = x >> 1
            power = power + 1
    return sum
```

Để đọc các thông tin chi tiết về Boot Sector của FAT32, đầu tiên ta đọc 512 byte đầu tiên. Vì FAT32 lưu trữ dữ liệu theo kiểu little-endian nên ta sẽ đọc lần lượt các byte như bảng trên rồi dùng hàm little_endian_to_integer để có được thông tin đúng

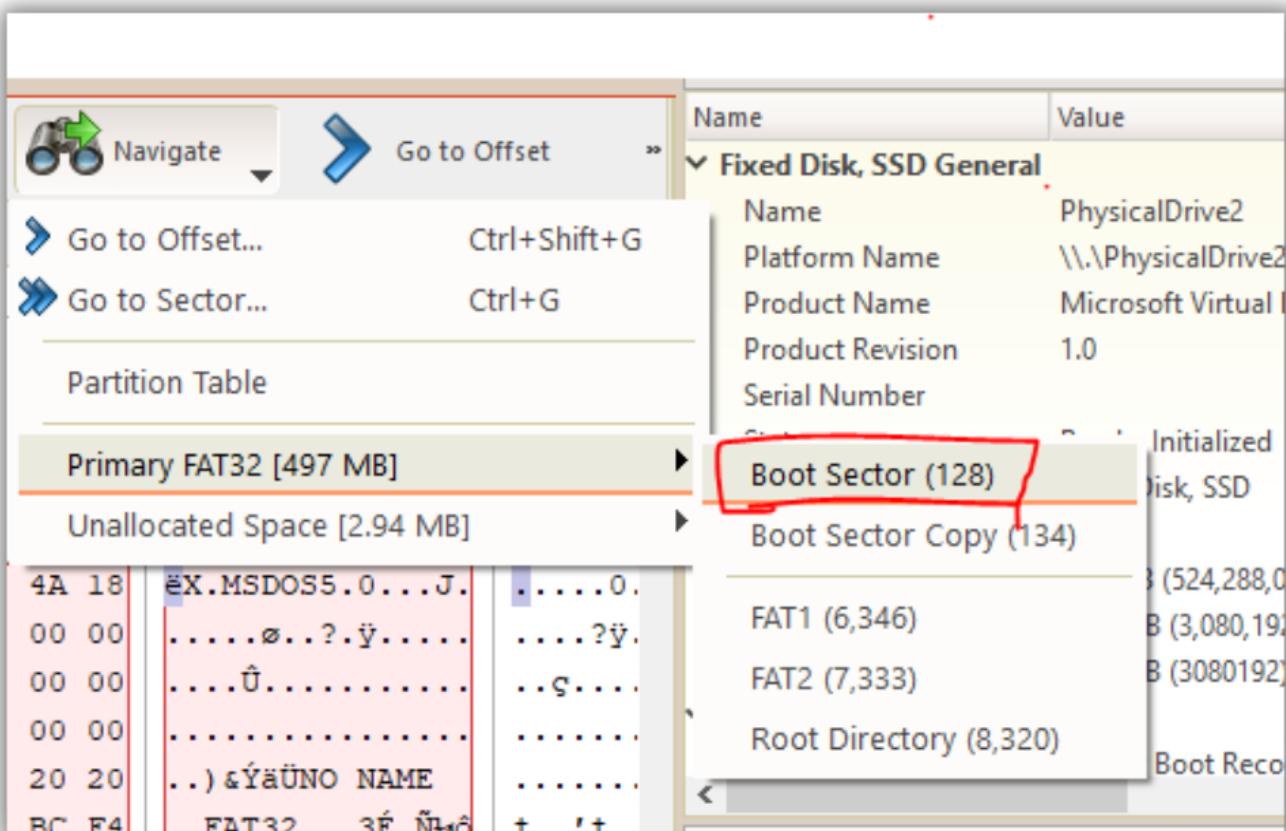
```
self.bytesPerSec = little_endian_to_integer(data, 11, 2)
self.secPerClus = little_endian_to_integer(data, 13, 1)
self.reservedSector = little_endian_to_integer(data, 14, 2)
self.numFATs = little_endian_to_integer(data, 16, 1)
self.entryOfRoot = little_endian_to_integer(data, 17, 2)
self.startSector = little_endian_to_integer(data, 28, 4)
self.totalSector = little_endian_to_integer(data, 32, 4)
self.fatSizeInSec = little_endian_to_integer(data, 36, 4)
self.rootClus = little_endian_to_integer(data, 44, 4)
```

Bytes per Sector: 512
 Sectors per Cluster: 8
 Reserved Sector: 6218
 Number of FAT: 2
 Entries of RDET: 0
 Boot sector Start: 128 *
 Total Sectors in Volume: 1017856
 Sectors per FAT: 987
 First Cluster of RDET: 2

Từ đó ta sẽ đạt được các trường thông tin mô tả hệ thống FAT32 như hình ảnh bên trên

Để kiểm tra thông tin của một ổ đĩa đã được đọc đúng hay chưa, nhóm sử dụng phần mềm hỗ trợ là Disk Editor.

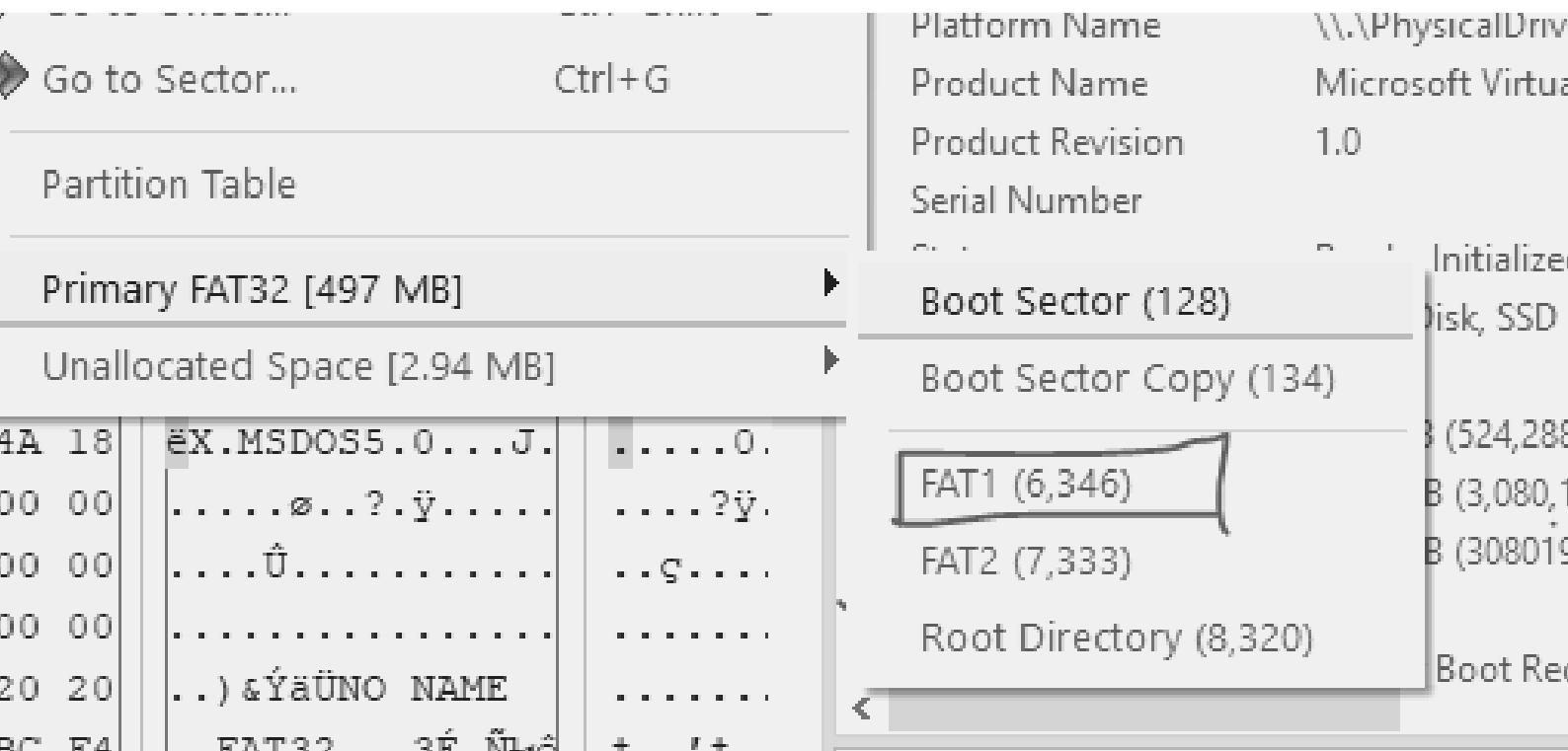
Ta thấy vị trí bắt đầu của BootSector là 128 đúng như trong Disk Editor



HIỂN THỊ CÂY THƯ MỤC GỐC

Vùng Fat: Chức năng và nhiệm vụ

- Vùng FAT có 2 nhiệm vụ quan trọng là:
 - Cho biết trạng thái cấp phát các cluster trong vùng Data
 - Xác định các cluster thuộc về tập tin. Nếu biết đc cluster bắt đầu của một tập tin, dựa vào vùng FAT sẽ biết được các cluster còn lại của tập tin này
- Các chỉ số sector đều phải cộng thêm số sector ẩn trước volume (vị trí sector bắt đầu của BootSector)
- Vùng FAT sẽ bắt đầu tại sector: $S_B + \text{số sector ẩn trước volume} = 6218 + 128 = 6346$ (tương đồng với Disk Editor như hình dưới)



- Kích thước của vùng FAT: $N_F * S_F = 1974$ sector. Vùng FAT được chia thành các entry, đối với FAT32 mỗi entry chiếm 4 byte
- Các entry được đánh chỉ mục từ 0. Vì 2 cluster đầu tiên không sử dụng để chứa số hiệu cluster nên ta sẽ bắt đầu từ cluster thứ 2. Chỉ số cluster bắt đầu đc lưu ở BootSector.
- Một số giá trị của entry:
 - 0: Nếu cluster chưa được cấp phát
 - Khác 0: vị trí cluster kế tiếp thuộc về tập tin
 - Lớn hơn 0xFFFFFFF8: thường là dấu hiệu kết thúc tập tin (EOF - End Of File)

003249152	FF FF FF OF FF FF FF FF	FF FF FF OF FF FF FF OF	0YY.YYYYYYYY.YYY.....
003249168	FF FF FF OF FF FF FF OF	FF FF FF OF FF FF FF OF	YY.YYY.YYY.YYY.....
003249184	FF FF FF OF FF FF FF OF	FF FF FF OF FF FF FF OF	YY.YYY.YYY.YYY.....
003249200	FF FF FF OF FF FF FF OF	FF FF FF OF 00 00 00 00	YY.YYY.YYY.....

Khi biết được số hiệu cluster bắt đầu của một tập tin, tìm tới entry tương ứng trong FAT Structure, liên kết tới các entry kế tiếp cho tới khi gặp dấu hiệu kết thúc EOF. Tập hợp các entry này là các cluster chứa dữ liệu của tập tin trong vùng Data.

Giá trị ở các entry của vùng FAT:

- Ta không quan tâm đến 2 entry bắt đầu
 - Giá trị 268435455 trong hệ 10 tương ứng với giá trị 0xFFFF FFFF trong hệ 16
- > Dấu hiệu kết thúc EOF
 -> Hầu hết các tập tin chỉ chiếm 1 cluster

```

FAT Entry[0] = 268435448
FAT Entry[1] = 4294967295
FAT Entry[2] = 268435455
FAT Entry[3] = 268435455
FAT Entry[4] = 268435455
FAT Entry[5] = 268435455
FAT Entry[6] = 268435455
FAT Entry[7] = 268435455
FAT Entry[8] = 268435455
FAT Entry[9] = 268435455
FAT Entry[10] = 268435455
FAT Entry[11] = 268435455
FAT Entry[12] = 268435455
FAT Entry[13] = 268435455
FAT Entry[14] = 268435455
  
```

HIỂN THỊ CÂY THƯ MỤC GỐC

- Để đọc cây thư mục của FAT32, đầu tiên ta tìm chỉ số sector đầu tiên của vùng dữ liệu (FirstDataSector) : Vị trí sector bắt đầu vùng FAT ($6346 +$ kích thước vùng FAT $= 8320$) = 8320. Đây cũng là sector chứa RDET.
- Một Directory Entry có kích thước 32 byte. Mỗi Directory Entry sẽ mô tả thông tin về một tập tin như tên, kích thước, thuộc tính cũng như địa chỉ cluster bắt đầu của vùng nội dung
- Trong RDET thường có cả entry chính và entry phụ:
 - Entry chính chỉ hỗ trợ tên tập tin dài nhất là 8 ký tự, phần mở rộng là 3 ký tự
 - Từ 2 byte cao và 2 byte thấp ta đọc giá trị của 4 byte sẽ được vị trí cluster bắt đầu

Byte thứ (hệ 10)	Kích thước (bytes)	Mô tả
0	1	Kí tự đầu tiên của tên tập tin
1	10	Kí tự thứ 2 đến 11 của tên tập tin
11	1	Byte thuộc tính (nếu giá trị bằng 10 thì là thư mục, nếu bằng 20 thì là tập tin)
20	2	Phần byte cao của cluster bắt đầu
26	2	Phần byte thấp của cluster bắt đầu
28	4	Kích thước của tập tin, đơn vị tính là byte. Nếu là thư mục, giá trị này là 0

- Các tên tập tin dài hơn 8 ký tự hoặc các tên tập tin có ký tự đặc biệt sẽ được lưu ở các entry phụ. Các entry này sẽ được đặt trước entry chính của tập tin đó và theo thứ tự ngược.
- Do đó, entry phụ đầu tiên được tìm thấy sẽ là entry phụ cuối cùng lưu tên của một tập tin

Byte thứ	Mô tả
1-10	Ký tự thứ 1-5
14-25	Ký tự thứ 6-11
28-31	Ký tự thứ 12-13

- 1 file thường gồm 1 entry chính và 0 hoặc nhiều entry phụ. Ta sẽ đọc các entry phụ (nếu có) trước rồi sau đó mới đọc tới entry chính. Tên được lưu trữ trong entry phụ sẽ được cộng dồn theo chiều ngược để có được tên của file (ví dụ minh họa ở hình bên dưới).
- Các vị trí để lưu các ký tự của tên tập tin nếu không sử dụng hết sẽ được chèn kí tự NULL sau đó là các giá trị 0xFF

ENTRY PHỤ 004260112	E5 6D 00 65 00 6E 00 74 00 2E 00 0F 00 9F 74 00 78 00 74 00 00 00 FF FF FF FF FF FF FF E5 4E 00 65 00 77 00 20 00 54 00 0F 00 9F 65 00 78 00 74 00 20 00 44 00 6F 00 00 00 63 00 75 00	åm.e.n.t..x.t...ÿÿÿÿåN.e.w. .I.x.t. .D.o.
ENTRY CHÍNH	E5 45 57 54 45 58 7E 31 54 58 54 20 00 4F 84 46 59 53 59 53 00 00 85 46 59 53 00 00 00 00 00 00 41 20 20 20 20 20 20 20 54 58 54 20 18 4F 84 46	åEWTEX~1TXYSYS...FYS A TX

Như hình trên, tên long_name sẽ được cộng dồn theo chiều ngược:
New Text Docu + ment.txt → "New Text Document.txt"

HIỂN THỊ CÂY THƯ MỤC GỐC

- Từ chỉ số các cluster bắt đầu của file ta cũng có được chỉ số sector đầu tiên của file đc lưu trên đĩa cứng qua công thức:
 - (N - 2) * S_C + FirstDataSector với N là chỉ số cluster bắt đầu của file và FirstDataSector là chỉ số sector đầu tiên của vùng dữ liệu.
 - Ví dụ với thư mục folder_1 có chỉ số cluster là 12 thì sector đầu tiên chứa thông tin của thư mục là $(12-2)*8 + 8320 = 8400$

CHỈ SỐ SECTOR : 8400

File PY1 PY là tập tin nằm trong thư mục Folder 1

Hình minh họa cho ví dụ

HIỂN THỊ CÂY THƯ MỤC GỐC

- Ta có thể thấy từ cluster đầu của thư mục ta có thể tìm đến sector chứa các tập tin/thư mục khác nằm trong nó. Còn cluster đầu của tập tin lại dẫn đến sector chứa nội dung của tập tin đó. Nếu tập tin được lưu trong nhiều hơn 1 cluster thì các cluster còn lại của tập tin sẽ được tìm thấy trong bảng FAT.
- Từ chỉ số cluster ta sẽ có được chỉ số sector của file, từ đó đọc được nội dung của file. Ví dụ như File PY1 PY nằm trong thư mục Folder 1 nằm ở cluster 13 thì sector đầu tiên chứa thông tin của tập tin là $(13-2)*8 + 8320 = 8408$

Offset	00 01 02 03 04 05 06 07	08 09 10 11 12 13 14 15	ASCII
004304816	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
004304832	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
004304848	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
004304864	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
004304880	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
004304896	69 6D 70 6F 72 74 20 6F	73 00 00 00 00 00 00 00	import os.....
004304912	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00

Ở sector 8408 dễ thấy nội dung của tập tin là "import os"

- Tìm các cluster thuộc về một tập tin trong bảng FAT:
 - Chỉ số cluster bắt đầu cũng là chỉ số entry bắt đầu trong bảng FAT. Từ đó ta xem giá trị của 1 entry gồm 4byte

003249152	B8 FF FF OF	FF FF FF OF	0ÿÿ.ÿÿÿÿÿÿ.ÿÿÿ.
003249168	FF FF FF OF	FF FF FF OF	ÿÿ.ÿÿ.ÿÿ.ÿÿ.
003249184	FF FF FF OF	FF FF FF OF	ÿÿ.ÿÿ.ÿÿ.ÿÿ.
003249200	FF FF FF OF	FF FF FF OF	ÿÿ.ÿÿ.ÿÿ....

Nếu giá trị 4 byte đó khác 0 và bé hơn 0xFFFF FFFF thì ta tìm được cluster tiếp theo chứa nội dung của 1 tập tin

SECTION 2

NTFS

New Technology File System

OPERATION SYSTEM

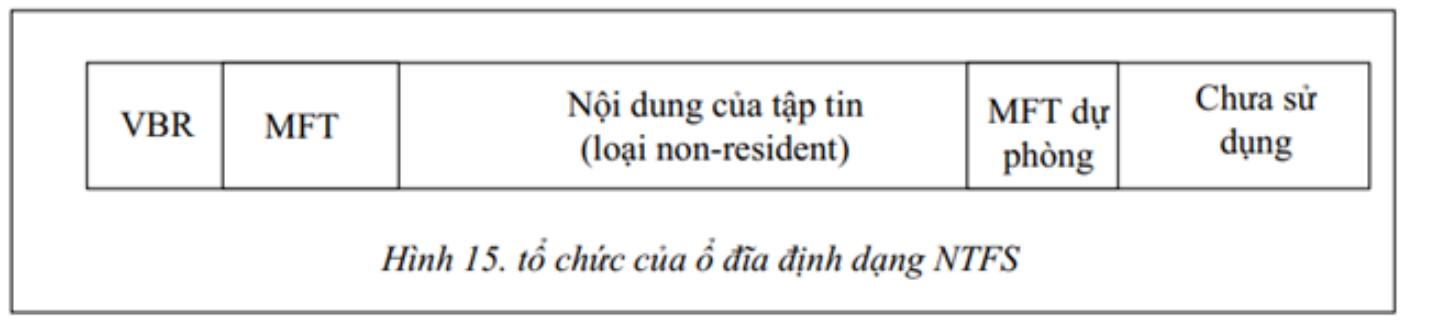
2021



MÔ TẢ TỔ CHỨC NTFS

VBR (Volume Boot Record) là bản ghi khởi động của ổ đĩa logic, nó luôn nằm ở vị trí đầu tiên của mỗi ổ đĩa logic. VBR chứa: mã khởi động, BPB, thông báo lỗi, và một số thông tin khác.

Tổ chức của ổ đĩa logic định dạng NTFS được minh họa ở Hình 15.



CẤU TRÚC CỦA VBR

BPB (Bios Parameter Block) bắt đầu tại offset 0xB tới offset 0x53 trong VBR, có kích thước 73 byte. BPB chứa một số thông tin mô tả về tổ chức của một ổ đĩa logic và hệ thống quản lý tập tin. Ví dụ: kích thước của một sector, số sector trong một cluster, tổng số sector trong ổ đĩa logic, vị trí bắt đầu của vùng MFT...

Địa chỉ (offset)	Mô tả	Kích thước (byte)
0 -> 2	Lệnh nhảy	3
3 -> A	Giá trị OEM ID hoặc tên hệ thống quản lý tập tin	8
B -> 53	BPB	73
54 -> 18B	Mã khởi động	312
18C -> 1F7	Thông báo lỗi	108
1F8 -> 1FD	Message offset	6
1FE -> 1FF	Dấu hiệu kết thúc VBR	2

CẤU TRÚC BPB (BIOS PARAMETER BLOCK)

BPB (Bios Parameter Block) bắt đầu tại offset 0xB tới offset 0x53 trong VBR, có kích thước 73 byte. BPB chứa một số thông tin mô tả về tổ chức của một ổ đĩa logic và hệ thống quản lý tập tin. Ví dụ: kích thước của một sector, số sector trong một cluster, tổng số sector trong ổ đĩa logic, vị trí bắt đầu của vùng MFT...

Để đọc được BPB nằm trong VBR em xây dựng class VBR với các thông số hiển thị theo cấu trúc của trường BPB ở trên.

0Bh	2	Kích thước một sector. Đơn vị tính là byte.
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 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 logic.
20h	4	Luôn mang giá trị 0, hệ thống NTFS không sử dụng tới trường 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.

Các bước thực hiện

Class VBR

Em vừa kết hợp khởi tạo class và tạo các thuộc tính là các thông số của BPB

```
lass VBR(object):
    """docstring for VBR"""

    def __init__(self, disk):
        with open(f'\\\\\\\\\\\\{disk}:', "rb") as fr:
            data = fr.read(512) # đọc ra VBR chính là 512 byte đầu tiên của ổ đĩa logic

        # Đọc các thuộc tính trên BPB
        self.bytesPerSec = int.from_bytes(data[11:13], 'little')
        self.secPerClus = int.from_bytes(data[13:14], 'little')
        self.NumSecOrTrack = int.from_bytes(data[24:26], 'little')
        self.NumSide = int.from_bytes(data[26:28], 'little')
        self.startSecDisk = int.from_bytes(data[28:32], 'little')
        self.totalSector = int.from_bytes(data[40:48], 'little')
        self.startClusterMFT = int.from_bytes(data[48:56], 'little')
        self.startClusterMFTMirror = int.from_bytes(data[56:64], 'little')
        self.sizeOfMFTEntry = pow(2, abs(hex_2_signed_int(data[64:65].hex(), 8))) # Đơn vị Byte
        self.volSerialNum = data[76:80].hex()
```

Hình ảnh class VBR thực hiện đọc các thông tin NTFS

Các dữ liệu được lưu trữ theo kiểu Little endian nên em dùng hàm được tích hợp sẵn để chuyển Little endian(như đã đề cập ở phần đọc thông tin FAT) thành int.

Tiếp đó, em sẽ chuyển các chuỗi có dạng hex(hệ 16) sang các số nguyên có dấu với các hàm được cài đặt bằng phép dịch bit, như hình dưới.

```
def hex_2_signed_int(hex_str, bits):
    value = int(hex_str, 16)
    if value & (1 << (bits - 1)):
        value -= 1 << bits
    return value
```

Hình ảnh hàm chuyển các hex string

Next Steps

Đọc các thông số của BPB thì em cho in ra với các thuộc tính đã được tạo bên trong lớp

Detail information partition

This is a NTFS Disk !!!

Bytes per Sector: 512

Sectors per Cluster: 8

Number of Sector/Track: 63

Number of Disk Side: 255

Start Sector of Disk: 128

Total Sector Of Disk: 4188159

Start Cluster of MFT: 174506

Start Cluster of MFT Mirror: 2

Size of MFT Entry:1024

Volume Serial Number:

cc7bdfb0a5dfb064

▼ BIOS Parameter Block	011	
Bytes per sector	011	512
Sectors per cluster	013	8
Reserved sectors	014	0
(always zero)	016	00 00 00
(unused)	019	00 00
Media descriptor	021	248
(unused)	022	00 00
Sectors per track	024	63
Number of heads	026	255
Hidden sectors	028	128
(unused)	032	00 00 00 00
Signature	036	80 00 80 00
Total sectors	040	4,188,159
\$MFT cluster number	048	<u>174,506</u>
\$MFTMirr cluster num...	056	<u>2</u>
Clusters per File Recor...	064	246
Clusters per Index Block	068	1
Volume serial number	072	CC 7B DF B0 A5 DF B0 64

So sánh kết quả thu được và kết quả kiểm thử

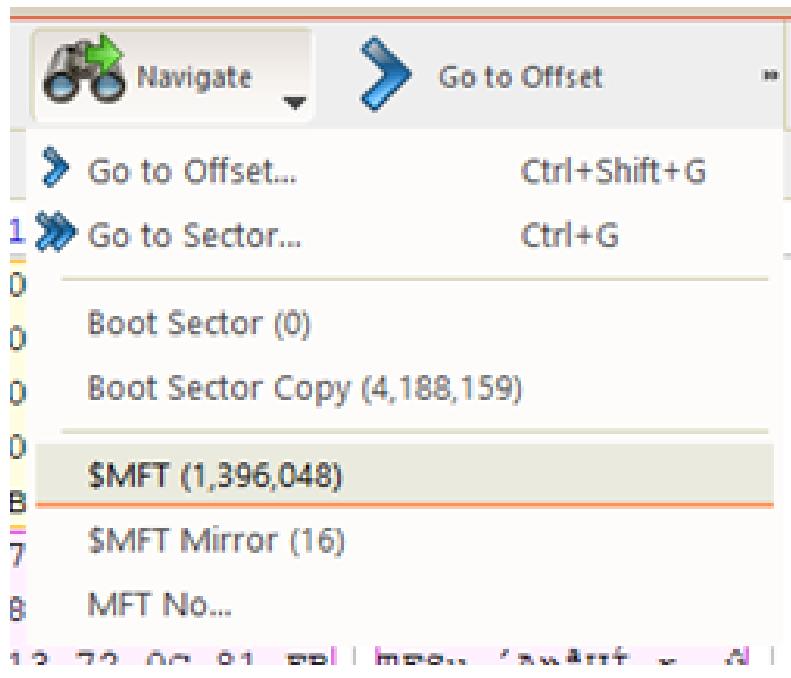
Như đã thấy ở hình trên, kết quả mà em in ra hoàn toàn trùng khớp với kết quả mà phần mềm kiểm thử đưa ra.

HIỂN THỊ CÂY THƯ MỤC

MFT (Master File Table):

Là thành phần quan trọng nhất trong hệ thống NTFS. MFT chứa thông tin về tất cả các tập tin và thư mục trong ổ đĩa logic.

Từ BPB ta có được cluster bắt đầu của MFT là 174506
đó Số sector bắt đầu: Sector = Cluster * Sector/Cluster = 174
 $506 * 8 = 1\,396\,048$



Hình ảnh kiểm thử

5	46 49 4C	45 30 00 03 00	F2 4F 20	00 00 00 00 00 00	FILE0...80	..0... .
2	01 00 01 00 38 00 01 00	A0 01 00 00 00 04 00 008....	..8.o.È.		
3	00 00 00 00 00 00 00 00	07 00 00 00 00 00 00 00		
4	02 00 00 00 00 00 00 00	10 00 00 00 60 00 00 00`.`.		
0	00 00 18 00 00 00 00 00	48 00 00 00 18 00 00 00H....H...		
6	2A 5C 68 DE 84 D1 D7 01	2A 5C 68 DF 84 D1 D7 01	*\hB.Ñx.*\hB.Ñx.	...0...0		
2	2A 5C 68 DF 84 D1 D7 01	2A 5C 68 DF 84 D1 D7 01	*\hB.Ñx.*\hB.Ñx.	...0...0		
3	06 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00		
4	00 00 00 00 00 01 00 00	00 00 00 00 00 00 00 00A....A...		
0	00 00 00 00 00 00 00 00	30 00 00 00 68 00 00 000...h...0.h..		
6	00 00 18 00 00 00 03 00	4A 00 00 00 18 00 01 000...h...0.h..		
2	05 00 00 00 00 00 05 00	2A 5C 68 DF 84 D1 D7 01J.....J...		
3	2A 5C 68 DF 84 D1 D7 01	2A 5C 68 DF 84 D1 D7 01*\hB.Ñx.0		
4	2A 5C 68 DF 84 D1 D7 01	00 40 00 00 00 00 00 00	*\hB.Ñx.*\hB.Ñx.	...0...0		
0	00 40 00 00 00 00 00 00	06 00 00 00 00 00 00 00	*\hB.Ñx...0....	...0....		
6	04 03 24 00 4D 00 46 00	54 00 00 00 00 00 00 00	0.....		
2	80 00 00 00 48 00 00 00	01 00 40 00 00 00 06 00	..S.M.F.T.....	\$MFT...		
3	00 00 00 00 00 00 00 00	3F 00 00 00 00 00 00 00H....0....	..H..0..		
4	40 00 00 00 00 00 00 00	00 00 04 00 00 00 00 00?.....?...		
0	00 00 04 00 00 00 00 00	00 00 04 00 00 00 00 00	0.....		
6	31 40 AA A9 02 00 00 00	B0 00 00 00 50 00 00 00	10*Ó.....P....	...Ó...P..		
2	01 00 40 00 00 00 05 00	00 00 00 00 00 00 00 00@.....	...@....		
3	01 00 00 00 00 00 00 00	40 00 00 00 00 00 00 00@.....	...@....		

Phần nội dung bắt đầu của \$MFT

14791952	0F 00 00 00 38 00 01 00	20 01 00 00 00 04 00 008.....	..8.Ø.E.
14791968	00 00 00 00 00 00 00 00	03 00 00 00 0F 00 00 00
14791984	01 00 00 00 00 00 00 00	10 00 00 00 00 10 00 00 00H....H..
14792000	00 00 18 00 00 00 00 00	30 00 00 00 00 10 00 00 000.....0...
14792016	2A 5C 60 DF 04 D1 D7 01	2A 5C 60 DF 04 D1 D7 01	*\hB.ñ*, *\hB.ñ*	...Ø...Ø
14792032	2A 5C 60 DF 04 D1 D7 01	2A 5C 60 DF 04 D1 D7 01	*\hB.ñ*, *\hB.ñ*	...Ø...Ø
14792048	06 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
14792064	50 00 00 00 80 00 00 00	00 00 18 00 00 00 02 00	P.....	P.....
14792080	64 00 00 00 18 00 00 00	01 00 04 80 48 00 00 00	d.....H..	d....H..
14792096	54 00 00 00 00 00 00 00	14 00 00 00 02 00 34 00	T.....4..	T.....4..
14792112	02 00 00 00 00 14 00	9F 01 12 00 01 01 00 00Ø..Ø..
14792128	00 00 00 05 12 00 00 00	00 00 18 00 9F 01 12 00d....Ø..
14792144	01 02 00 00 00 00 05	20 00 00 00 20 02 00 00d ..Ø..
14792160	01 01 00 00 00 00 05	12 00 00 00 01 02 00 00d ..Ø..
14792176	00 00 00 05 20 00 00 00	20 02 00 00 00 00 00 00d ..Ø]..
14792192	00 00 00 00 18 00 00 00	00 00 18 00 00 00 01 00
14792208	00 00 00 00 18 00 00 00	FF FF FF FF 00 00 00 00FFFF..
		00 00 00 00 00 00 00 00
		00 00 00 00 00 00 00 00
		00 00 00 00 00 00 00 00

MFT ENTRY

hình ảnh MFT Entry

MFT (Master File Table) được chia nhỏ thành các phần bằng nhau gọi là MFT entry.

Kích thước của một MFT entry được quy định trong BPB, thường là 1024 byte .

Khi một tập tin hoặc thư mục được tạo ra, sẽ có ít nhất một MFT entry được tạo ra trong MFT, để mô tả thông tin cho tập tin hoặc thư mục đó.

Windows dành một số MFT entry đầu tiên trong MFT cho các tập tin siêu dữ liệu. Các MFT entry còn lại cho các tập tin dữ liệu.

Entry thứ 15 (bắt đầu từ 0) là entry siêu dữ liệu cuối dành cho Windows. Sau đó sẽ có 1 vài entry trống rồi bắt đầu tới các entry để chứa các tập tin thư mục.

Và entry chứa các dữ liệu tập tin thư mục bắt đầu tại entry 24

0714792960	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714792976	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714792992	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793008	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793024	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793040	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793056	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793072	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793088	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793104	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793120	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793136	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793152	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793168	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793184	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793200	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793216	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793232	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793248	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793264	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793280	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00
0714793296	00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00

hình ảnh MFT Entry thứ 15 (Entry siêu dữ liệu)

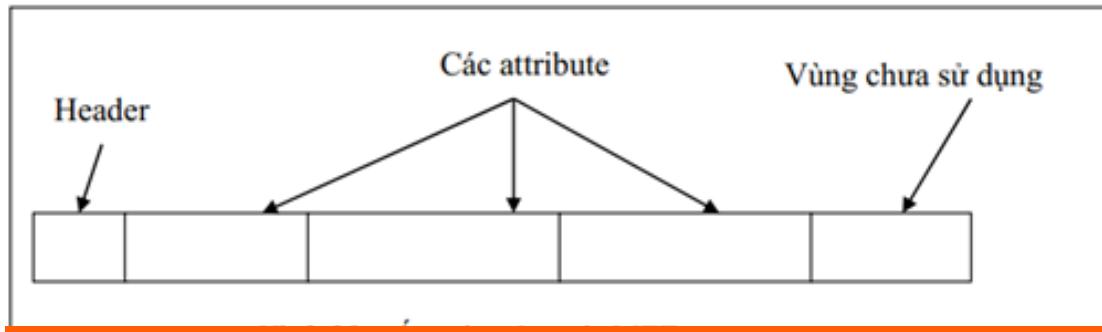
MFT ENTRY

0714801168	01 00 01 00 38 00 0D 00	70 02 00 00 00 04 00 008...p....	...8.wq.t...
0714801184	00 00 00 00 00 00 00 00	04 00 00 00 18 00 00 00
0714801200	09 00 00 00 00 00 00 00	10 00 00 00 60 00 00 00
0714801216	00 00 00 00 00 00 00 00	48 00 00 00 18 00 00 00H.....H....
0714801232	2C 25 A3 DF 84 D1 D7 01	2C 25 A3 DF 84 D1 D7 01	,%EB.Ñ%,%EB.Ñ%	T..ØT..t...
0714801248	2C 25 A3 DF 84 D1 D7 01	2C 25 A3 DF 84 D1 D7 01	,%EB.Ñ%,%EB.Ñ%	T..ØT..t...
0714801264	06 00 00 20 00 00 00 00	00 00 00 00 00 00 00 00
0714801280	00 00 00 00 01 01 00 00	00 00 00 00 00 00 00 008...
0714801296	00 00 00 00 00 00 00 00	30 00 00 00 68 00 00 000..h...0.h...
0714801312	00 00 00 00 00 00 01 00	4E 00 00 00 18 00 01 00N.....N...
0714801328	0B 00 00 00 00 00 0B 00	2C 25 A3 DF 84 D1 D7 01,%EB.Ñ%T..t...
0714801344	2C 25 A3 DF 84 D1 D7 01	2C 25 A3 DF 84 D1 D7 01	,%EB.Ñ%,%EB.Ñ%	T..ØT..t...
0714801360	2C 25 A3 DF 84 D1 D7 01	00 00 00 00 00 00 00 00	,%EB.Ñ%.....	T..Ø...
0714801376	00 00 00 00 00 00 00 00	06 00 00 20 00 00 00 00
0714801392	06 00 24 00 51 00 75 00	6F 00 74 00 61 00 00 00	.S.Q.u.o.t.a...	.SQuota...
0714801408	90 00 00 00 78 00 00 00	00 02 18 00 00 00 03 00x.....	...x.À...
0714801424	58 00 00 00 20 00 00 00	24 00 4F 00 00 00 00 00	X....S.O.....	X..SO...
0714801440	00 00 00 00 11 00 00 00	00 10 00 00 01 00 00 00
0714801456	10 00 00 00 48 00 00 00	48 00 00 00 00 00 00 00H...H...	..H.H...
0714801472	20 00 04 00 00 00 00 00	28 00 10 00 00 00 00 00(.....	...(..
0714801488	01 02 00 00 00 00 05	20 00 00 00 20 02 00 00	A..d ..ñ...
0714801494	AA AA AA AA AA AA AA	AA AA AA AA AA AA AA

hình ảnh MFT Entry thứ 24 (Entry chứa các tập dữ liệu)

CẤU TRÚC MFT ENTRY

Cấu trúc của một MFT ENTRY



Cấu trúc của một MFT ENTRY

HEADER

Offset	Số byte	Mô tả		
0x0 – 0x03	4	Dấu hiệu nhận biết MFT entry.	0x14 – 0x15	2
0x04 – 0x05	2	Địa chỉ (offset) của Update sequence.	0x16 – 0x17	2
0x06 – 0x07	2	Số phần tử của mảng Fixup, mảng này chứa các giá trị bị thay thế trong quá trình thao tác với Update sequence.	0x18 – 0x1B	4
0x08 – 0x0F	8	\$LogFile Sequence Number (LSN): mã định danh MFT entry của file log (log record).	0x1C – 0x1F	4
0x10 – 0x11	2	Sequence Number: cho biết số lần MFT entry này đã được sử dụng lại. Giá trị này được tăng lên một đơn vị sau mỗi lần tập tin tương ứng với MFT entry này bị xóa. Mang giá trị 0 nếu MFT entry này chưa được sử dụng.	0x20 – 0x27	8
0x12 – 0x13	2	Reference Count: cho biết số thư mục mà tập	0x28 – 0x29	2

Dựa vào những thông tin này ta sẽ thử làm một ví dụ thực tế trên một entry được lấy từ phần mềm kiểm thử để xem thông tin lấy ra được sẽ như thế nào

VÍ DỤ

Phần nội dung MFT entry của tập tin test.txt tại offset 714 821 632

Offset	00 01 02 03 04 05 06 07	08 09 0A 0B 0C 0D 0E 0F	ASCII	Unicode
2A9B4FF0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 07 00
2A9B5000	46 49 4C 45 30 00 03 00	9A AC 20 00 00 00 00 00 00	FILE0.....	..0... ..
2A9B5010	01 00 01 00 38 00 01 00	60 01 00 00 00 04 00 008...`.....	..8.š.È.
2A9B5020	00 00 00 00 00 00 00 00	05 00 00 00 2C 00 00 00,.....,
2A9B5030	0A 00 00 00 00 00 00 00	10 00 00 00 60 00 00 00`.....`.
2A9B5040	00 00 00 00 00 00 00 00	48 00 00 00 18 00 00 00H.....H...
2A9B5050	83 53 79 24 8A D1 D7 01	3B 91 06 AC 8A D1 D7 01	.SyS.Ñw.;.-.Ñw.	..0...0
2A9B5060	3B 91 06 AC 8A D1 D7 01	3B 91 06 AC 8A D1 D7 01	;.-.Ñw.;.-.Ñw.	..0...0
2A9B5070	20 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
2A9B5080	00 00 00 00 0C 01 00 00	00 00 00 00 00 00 00 00č....
2A9B5090	00 00 00 00 00 00 00 00	30 00 00 00 70 00 00 000...p...0.p...
2A9B50A0	00 00 00 00 00 00 03 00	52 00 00 00 18 00 01 00R.....R...
2A9B50B0	05 00 00 00 00 05 00	83 53 79 24 8A D1 D7 01SyS.Ñw.Ü
2A9B50C0	83 53 79 24 8A D1 D7 01	83 53 79 24 8A D1 D7 01	.SyS.Ñw..SyS.Ñw.	..0...0
2A9B50D0	83 53 79 24 8A D1 D7 01	00 00 00 00 00 00 00 00	.SyS.Ñw.....	..0....
2A9B50E0	00 00 00 00 00 00 00 00	20 00 00 00 00 00 00 00
2A9B50F0	08 00 74 00 65 00 73 00	74 00 2E 00 74 00 78 00	..t.e.s.t...t.x.	.test.tx
2A9B5100	74 00 00 00 00 00 00 00	40 00 00 00 28 00 00 00	t.....@...{...	t...@.(.
2A9B5110	00 00 00 00 00 00 04 00	10 00 00 00 18 00 00 00
2A9B5120	3D 4B CA 62 53 3D EC 11	BE 2F DC F5 05 23 1B C4	=KÉbs=i.%/Üß.#.À
2A9B5130	80 00 00 00 28 00 00 00	00 00 18 00 00 00 01 00(.....(.....
2A9B5140	0C 00 00 00 18 00 00 00	46 69 6C 65 20 69 6E 20File in
2A9B5150	52 6F 6F 74 00 00 00 00	FF FF FF FF 82 79 47 11	Root.....yyyy.yG.
2A9B5160	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
2A9B5170	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00

Tập tin test.txt

Header gồm 42 bytes đầu tiên:

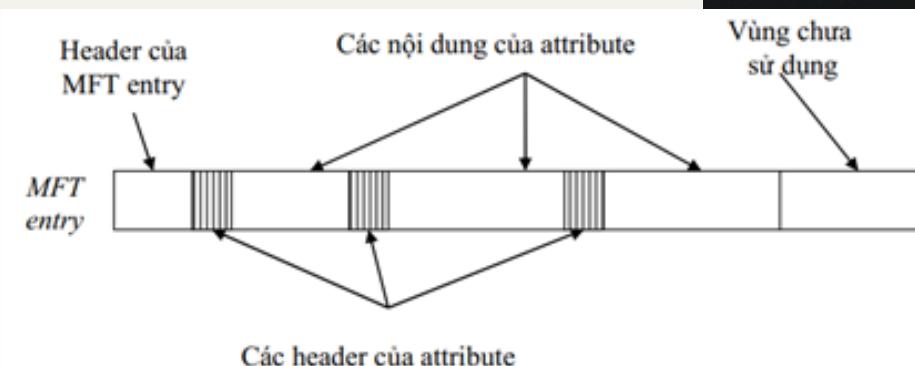
- Offset 0x00 – 0x03 → FILE: Dấu hiệu nhận biết MFT entry là “FILE” nếu entry bị lỗi sẽ hiển thị “BAAD”.
- Offset 0x14 – 0x15 → 0038h = 56: Địa chỉ bắt đầu của Attribute trong MFT entry này là byte thứ 56.
- Offset 0x18 – 0x1B → 0160h = 352: Số bytes mà entry này đã sử dụng 352 bytes.

ATTRIBUTE

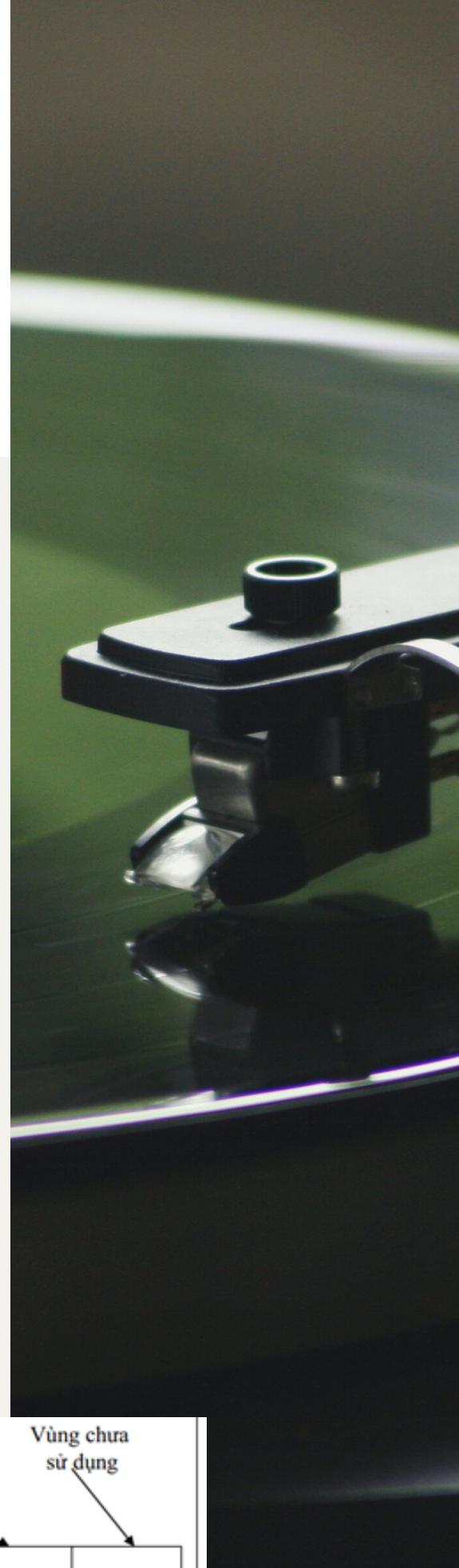
OPERATION SYSTEM

Attribute là một cấu trúc dữ liệu, được sử dụng để chứa nội dung của tập tin, chứa các thông tin liên quan đến tập tin, thư mục,...v.v trong hệ thống NTFS.

Có nhiều loại attribute, mỗi loại có cấu trúc tổ chức riêng, có một mã loại (type ID) riêng. Mã loại là một số nguyên. Microsoft sắp xếp thứ tự các attribute trong mỗi MFT entry theo chiều tăng dần của mã loại, nghĩa là, attribute nào có mã loại nhỏ sẽ đứng trước, attribute nào có mã loại lớn sẽ đứng sau.



Hình 24. cấu trúc của một attribute



MỘT SỐ LOẠI ATTRIBUTE

Bảng sau liệt kê một số loại attribute.

Mã loại (hệ 10)	Loại attribute	Mô tả
16	\$STANDARD_INFORMATION	Chứa thông tin chung, ví dụ: các cờ, thời gian tạo, thời gian truy cập mới nhất, thời gian ghi mới nhất, người sở hữu, định danh bảo mật (security ID).
32	\$ATTRIBUTE_LIST	Cho biết vị trí các attribute của một tập tin.
48	\$FILE_NAME	Chứa tên tập tin (dạng Unicode), thời gian tạo, thời điểm ghi tập tin mới nhất, thời điểm truy cập mới nhất.
64	\$VOLUME_VERSION	Chứa thông tin về ổ đĩa. Chỉ có ở phiên bản 1.2
64	\$OBJECT_ID	Chứa định danh duy nhất của tập tin hoặc thư mục. Chỉ có ở phiên bản 3.0 trở về sau.
80	\$SECURITY_DESCRIPTOR	Chứa thông tin về bảo mật và thông tin kiểm soát truy cập của tập tin.
96	\$VOLUME_NAME	Chứa tên ổ đĩa logic.
112	\$VOLUME_INFORMATION	Chứa thông tin về phiên bản của hệ thống quản lý tập tin và các cờ hiệu.
128	\$DATA	Chứa nội dung của tập tin.
144	\$INDEX_ROOT	Chứa nút gốc (root node) của cây chỉ mục (index tree).
160	\$INDEX_ALLOCATION	Chứa các nút của cây chỉ mục (index tree) có gốc thuộc attribute \$INDEX_ROOT.
176	\$BITMAP	Chứa bitmap cho siêu tập tin \$MFT và cho các chỉ mục.
192	\$SYMBOLIC_LINK	Chứa thông tin liên kết mềm. Chỉ có ở phiên bản 1.2
192	\$REPARSE_POINT	Chứa thông tin liên kết mềm. Có ở các phiên bản 3.0 trở về sau.
208	\$EA_INFORMATION	Chứa thông tin đảm bảo việc tương thích với các ứng dụng trên nền OS/2.
224	\$EA	Chứa thông tin đảm bảo việc tương thích với các ứng dụng trên nền OS/2.
256	\$LOGGED.Utility_STREAM	Chứa khóa (key) và thông tin mã hóa attribute (encrypted attribute) trong các phiên bản từ 3.0 trở về sau.

ATTRIBUTE \$STANDARD_INFORMATION

Byte thứ	Mô tả
0 - 15	Cấu trúc header chuẩn (có trong tất cả các loại attribute – Hình 25).
16 – 19	Cho biết kích thước phần nội dung của attribute.
20 - 21	Cho biết nơi bắt đầu (offset) của phần nội dung.

Header của attribute

Từ header ở trên ta biết được offset bắt đầu của attribute đầu tiên là \$STANDARD_INFORMATION ở byte thứ 56 → 38h và header gồm 16 bytes

2A9B4FF0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 07 00
2A9B5000	46 49 4C 45 30 00 03 00	9A AC 20 00 00 00 00 00 00	FILE0.....	..0... ..
2A9B5010	01 00 01 00 38 00 01 00	60 01 00 00 00 00 04 00 008...`.....	..8.S.E.
2A9B5020	00 00 00 00 00 00 00 00 00 00	05 00 00 00 2C 00 00 00 00,.....
2A9B5030	0A 00 00 00 00 00 00 00 00 00	10 00 00 00 60 00 00 00 00`....`..
2A9B5040	00 00 00 00 00 00 00 00 00 00	48 00 00 00 18 00 00 00 00H.....	...H...
2A9B5050	83 53 79 24 8A D1 D7 01	3B 91 06 AC 8A D1 D7 01	.sy\$.Ñx.,...-.Ñx.	...Û...Û
2A9B5060	3B 91 06 AC 8A D1 D7 01	3B 91 06 AC 8A D1 D7 01	;...-.Ñx.;...-.Ñx.	...Û...Û
2A9B5070	20 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00

- Byte thứ 4 – 7 → 00 00 00 60h = 96: Kích thước attribute: 96 bytes
- Byte thứ 16 – 19 → 00 00 00 48h = 72: Kích thước nội dung của attribute 72 bytes
- Byte thứ 20 – 21 → 00 18h = 24: Byte bắt đầu của phần nội dung là byte thứ 24 + 56 = 80

Từ đây em xây dựng class cho attribute này. Có các thuộc tính cũng là các thông tin trong Attribute nhưng chỉ lấy những thông tin cần thiết

2A9B5010	01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2A9B5020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	05 00 00 00 2C 00 00 00 00 00 00 00 00 00 00 00,,
2A9B5030	0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	10 00 00 00 60 00 00 00 00 00 00 00 00 00 00 00`....`..
2A9B5040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	48 00 00 00 18 00 00 00 00 00 00 00 00 00 00 00H.....	...H...
2A9B5050	83 53 79 24 8A D1 D7 01	3B 91 06 AC 8A D1 D7 01	.sy\$.Ñx.,...-.Ñx.	...Û...Û
2A9B5060	3B 91 06 AC 8A D1 D7 01	3B 91 06 AC 8A D1 D7 01	;...-.Ñx.;...-.Ñx.	...Û...Û
2A9B5070	20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2A9B5080	00 00 00 00 0C 01 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00Č.....
2A9B5090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	30 00 00 00 70 00 00 00 00 00 00 00 00 00 00 000...p...	...0.p..
2A9B50A0	00 00 00 00 00 00 00 00 03 00	52 00 00 00 18 00 01 00R.....	...R...

Phần nội dung của attribute \$STANDARD_INFORMATION

ATTRIBUTE \$FILE_NAME

Attribute này để lưu các thông tin về tên của tập tin thư mục và nằm sau attribute \$STANDARD_INFORMATION

Cũng như attribute trước, cấu trúc của attribute gồm các phần sau

Byte thứ	Mô tả
0 – 15	Cấu trúc header chuẩn của attribute \$FILE_NAME.
16 – 19	Kích thước phần nội dung của attribute \$FILE_NAME.
20 – 21	Nơi bắt đầu (offset) của phần nội dung attribute \$FILE_NAME.

Header của attribute

- Byte thứ 4 – 7 → 00 00 00 70h = 112: Kích thước attribute: 112 bytes
- Byte thứ 16 – 19 → 00 00 00 52h = 82: Kích thước nội dung của attribute 82 bytes
- Byte thứ 20 – 21 → 00 18h = 24: Byte bắt đầu của phần nội dung là byte thứ 56 + 96 + 24 = 176

2A9B5080	00 00 00 00 0C 01 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00č.....
2A9B5090	00 00 00 00 00 00 00 00	30 00 00 00 70 00 00 00 00 00 00 00 00 00 00 000...p...0.p...
2A9B50A0	00 00 00 00 00 00 03 00	52 00 00 00 18 00 01 00R.....R...
2A9B50B0	05 00 00 00 00 00 05 00	83 53 79 24 8A D1 D7 01Sy\$.Ñx..ñ...
2A9B50C0	83 53 79 24 8A D1 D7 01	83 53 79 24 8A D1 D7 01	.Sy\$.Ñx..Sy\$.Ñx..	...ñ...ñ
2A9B50D0	83 53 79 24 8A D1 D7 01	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.Sy\$.Ñx.....	...ñ.....
2A9B50E0	00 00 00 00 00 00 00 00	20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2A9B50F0	08 00 74 00 65 00 73 00	74 00 2E 00 74 00 78 00	..t.e.s.t...t.x.	.test.tx
2A9B5100	74 00 00 00 00 00 00 00	40 00 00 00 28 00 00 00	t.....@...(t...@...(.
2A9B5110	00 00 00 00 00 00 04 00	10 00 00 00 18 00 00 00		

Phần nội dung của attribute \$FILE_NAME

- -Byte thứ 64 → 08h = 8: Chiều dài tên tập tin: 8 kí tự
- -Byte thứ 66 – 66 + 8 = 74: Tên tập tin: test.txt

Từ đó, dựa vào những thông tin trên em xây dựng class cho attribute này để phục vụ cho việc đọc cây thư mục.

ATTRIBUTE \$DATA

Trước khi đến attribute \$DATA phải bỏ qua 40 là attribute \$OBJECT_ID
 Phần attribute \$DATA

Offset	00 01 02 03 04 05 06 07	08 09 0A 0B 0C 0D 0E 0F	ASCII	Unicode
2A9B5050	83 53 79 24 8A D1 D7 01	3B 91 06 AC 8A D1 D7 01	.Sy\$.Ñx.;...ñ.Ñx.	...Ù...Ù
2A9B5060	3B 91 06 AC 8A D1 D7 01	3B 91 06 AC 8A D1 D7 01	;...ñ.Ñx.;...ñ.Ñx.	...Ù...Ù
2A9B5070	20 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
2A9B5080	00 00 00 00 0C 01 00 00	00 00 00 00 00 00 00 00�....
2A9B5090	00 00 00 00 00 00 00 00	30 00 00 00 70 00 00 000...p...0.p.
2A9B50A0	00 00 00 00 00 00 03 00	52 00 00 00 18 00 01 00R.....R...
2A9B50B0	05 00 00 00 00 00 05 00	83 53 79 24 8A D1 D7 01Sy\$.Ñx.Ù
2A9B50C0	83 53 79 24 8A D1 D7 01	83 53 79 24 8A D1 D7 01	.Sy\$.Ñx..sy\$.Ñx.	...Ù...Ù
2A9B50D0	83 53 79 24 8A D1 D7 01	00 00 00 00 00 00 00 00	.Sy\$.Ñx.....	...Ù....
2A9B50E0	00 00 00 00 00 00 00 00	20 00 00 00 00 00 00 00
2A9B50F0	08 00 74 00 65 00 73 00	74 00 2E 00 74 00 78 00	..t.e.s.t...tx.	.test.tx
2A9B5100	74 00 00 00 00 00 00 00	40 00 00 00 28 00 00 00	t.....@...(...	t...@.(...
2A9B5110	00 00 00 00 00 00 04 00	10 00 00 00 18 00 00 00
2A9B5120	3D 4B CA 62 53 3D EC 11	BE 2F DC F5 05 23 1B C4	=K�bs=i.%/05.#.A
2A9B5130	80 00 00 00 28 00 00 00	00 00 18 00 00 00 01 00(.....	...(...
2A9B5140	0C 00 00 00 18 00 00 00	46 69 6C 65 20 69 6E 20File in
2A9B5150	52 6F 6F 74 00 00 00 00	FF FF FF FF 82 79 47 11	Root....ÿÿÿ.yG.
2A9B5160	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00

Phần nội dung của attribute \$DATA

- Byte thứ 4 - 7 → 28h = 40: Kích thước của attribute: 40 bytes
- Byte thứ 16 - 19 → 00 00 00 0Ch = 12: Kích thước nội dung 12 bytes
- Byte thứ 20 - 21 → 0018h = 24: Byte bắt đầu của phần nội dung là byte thứ 56 + 96 + 112 + 24 = 288

Dựa vào những thông tin trên, em lại tiếp tục xây dựng class cho Attribute \$Data.

Class MFT Entry

```
class MFTEntry:
    def __init__(self, entry_data):
        # Header của entry gồm 42 bytes đầu tiên
        self.header = entry_data[0:43]
        # Offset của attribute đầu tiên cũng là standard trong MFT entry là : 20 → 21 trong header
        offset_standard_info = int.from_bytes(self.header[20:22], 'little')

        # Đọc attribute standard
        self.standard_info = StandardInformation(entry_data[offset_standard_info:])

        # Đọc attribute file_name
        offset_file_name = offset_standard_info + self.standard_info.size
        self.file_name = FileName(entry_data[offset_file_name:])

        # Đọc attribute Data
        offset_data = offset_file_name + self.file_name.size + 40
        self.data = Data(entry_data[offset_data:])

        # List các tt/tm con
        self.sub_file = []
```

- Bao gồm các thuộc tính là các attribute được tạo ở trên và có thuộc tính là mảng chứa các tập tinh/thư mục con.

```
# Lấy những entry con với entry cha là thư mục
def get_sub(self, all_entry):
    for i in range(len(all_entry)):
        # Nếu tên entry con nằm trong data content của entry cha thì thêm vào list sub
        temp_entry = all_entry[i]
        if (temp_entry is not None) and (temp_entry.file_name.name in self.data.content) \
            and (temp_entry.file_name.name != self.file_name.name):
            #temp_entry = all_entry[i]
            self.sub_file.append(temp_entry)
            all_entry[i] = None
        # Nếu entry con là thư mục thì tiếp tục xem con của entry con đó
        if temp_entry.standard_info.flag == b'\x00\x00\x00\x00':
            temp_entry.get_sub(all_entry)
```

- Phương thức lấy thư mục con trong class MFT entry là xét tất cả các entry nếu như entry nào trùng tên với nội dung data của entry đang xét thì thêm entry con vào mảng tt/tm con. Đồng thời kiểm tra xem entry con có phải là thư mục không để lấy tiếp mảng tt/tm con của entry con đó.

Class MFT (Chứa MFT ENTRY)

```
class MFT:
    def __init__(self, vbr, disk):
        self.MFT_entries_head = []
        self.MFT_entries_data = []

        # Vị trí Byte đầu tiên của MFT
        startByte = (vbr.startClusterMFT * vbr.secPerClus) * vbr.bytesPerSec

        with open(f'\\\\\\\\\\\\\\\\{disk}:', "rb") as fr:
            # entry đang đọc
            index_entry = 0
            # Đọc các MFT entry head → entry rỗng ở giữa entry head và các entry data
            while True:
                # Vị trí đọc MFT entry
                indexSeek = startByte + index_entry * vbr.sizeOfMFTEntry
                fr.seek(indexSeek)

                entryData = fr.read(vbr.sizeOfMFTEntry)

                # Nếu đọc 4 bytes đầu trong entry rỗng thì thoát while
                if entryData[0:4] == b'\x00\x00\x00\x00':
                    break
                else:
                    self.MFT_entries_head.append(entryData)
```

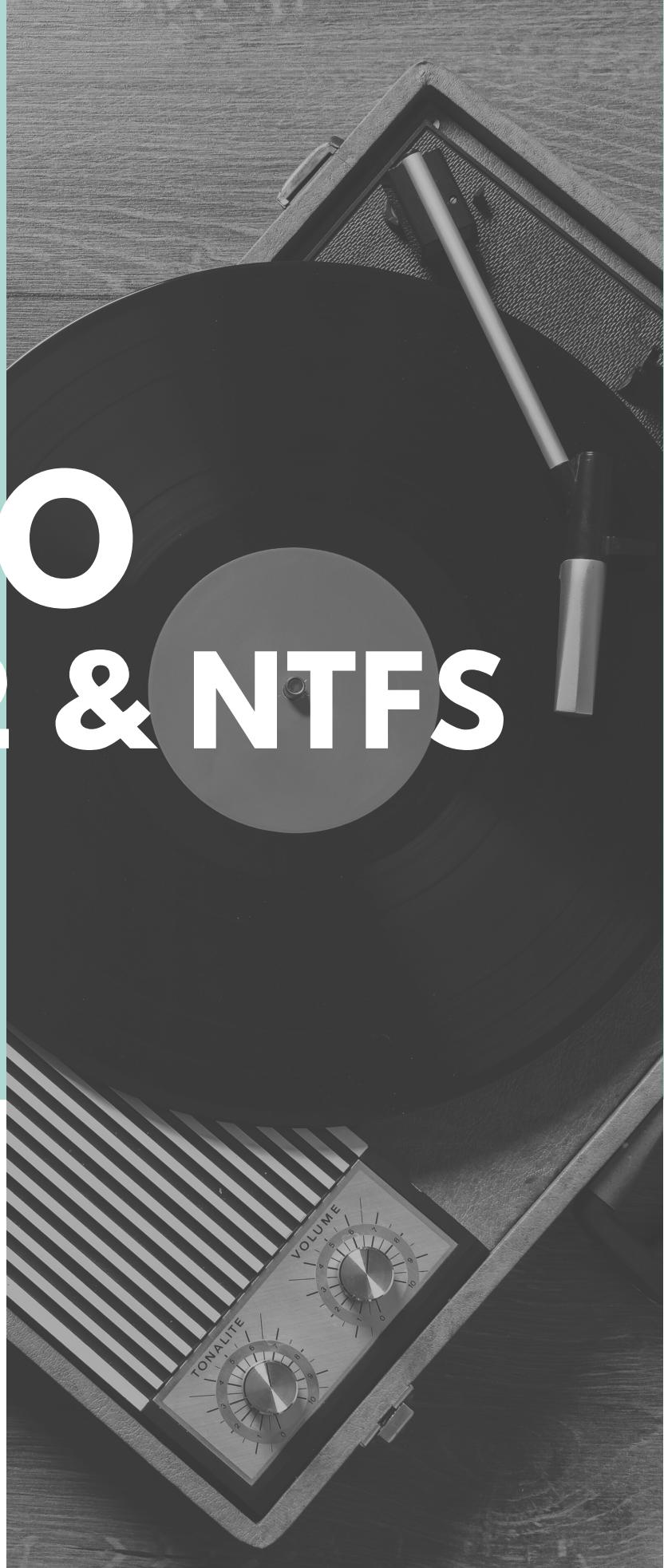
Hình ảnh minh họa class MFT, thực tế code dài hơn rất nhiều

- Với vòng lặp lần 1 ta sẽ đọc các entry dành cho Windows và đặt đó là các entry đầu cho đến khi hết các entry đầu.
- Như khảo sát ở trên sẽ có các entry rỗng nằm giữa entry đầu và entry chứa dữ liệu nên vòng lặp thứ 2 sẽ bỏ qua các entry rỗng này.
- Khi hết các entry rỗng sẽ đến entry chứa dữ liệu cây thư mục trong Volume và ta cứ đọc cho đến khi không còn entry nào nữa.
- Sau đó ta tiến hành lấy các tập tin hoặc thư mục con của các entry là thư mục.
- Cuối cùng, ta tiến hành lấy những tập tin/ thư mục nằm trên Root

DEMO FAT32 & NTFS



OPERATION SYSTEM



01

HÌNH ẢNH CỦA CHƯƠNG TRÌNH
(GIAO DIỆN ĐẸP MẮT, DỄ SỬ DỤNG)

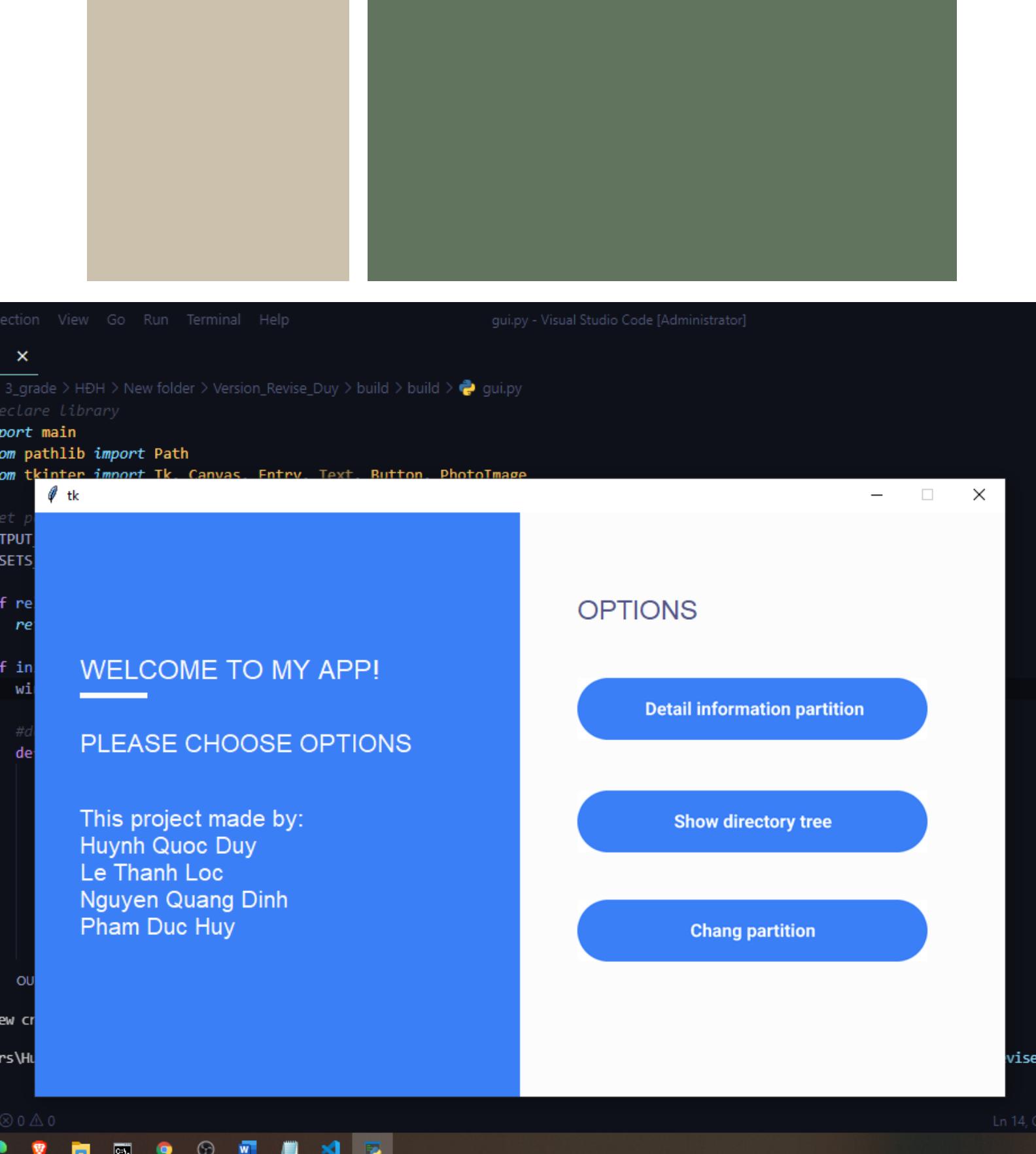
Hey There !

Enter your directory

Name directory (Please wait it's still running)

READ

GIAO DIỆN XUẤT HIỆN ĐỂ ĐỌC
PHÂN VÙNG (NHẬP TÊN PHÂN
VÙNG MUỐN ĐỌC)



02

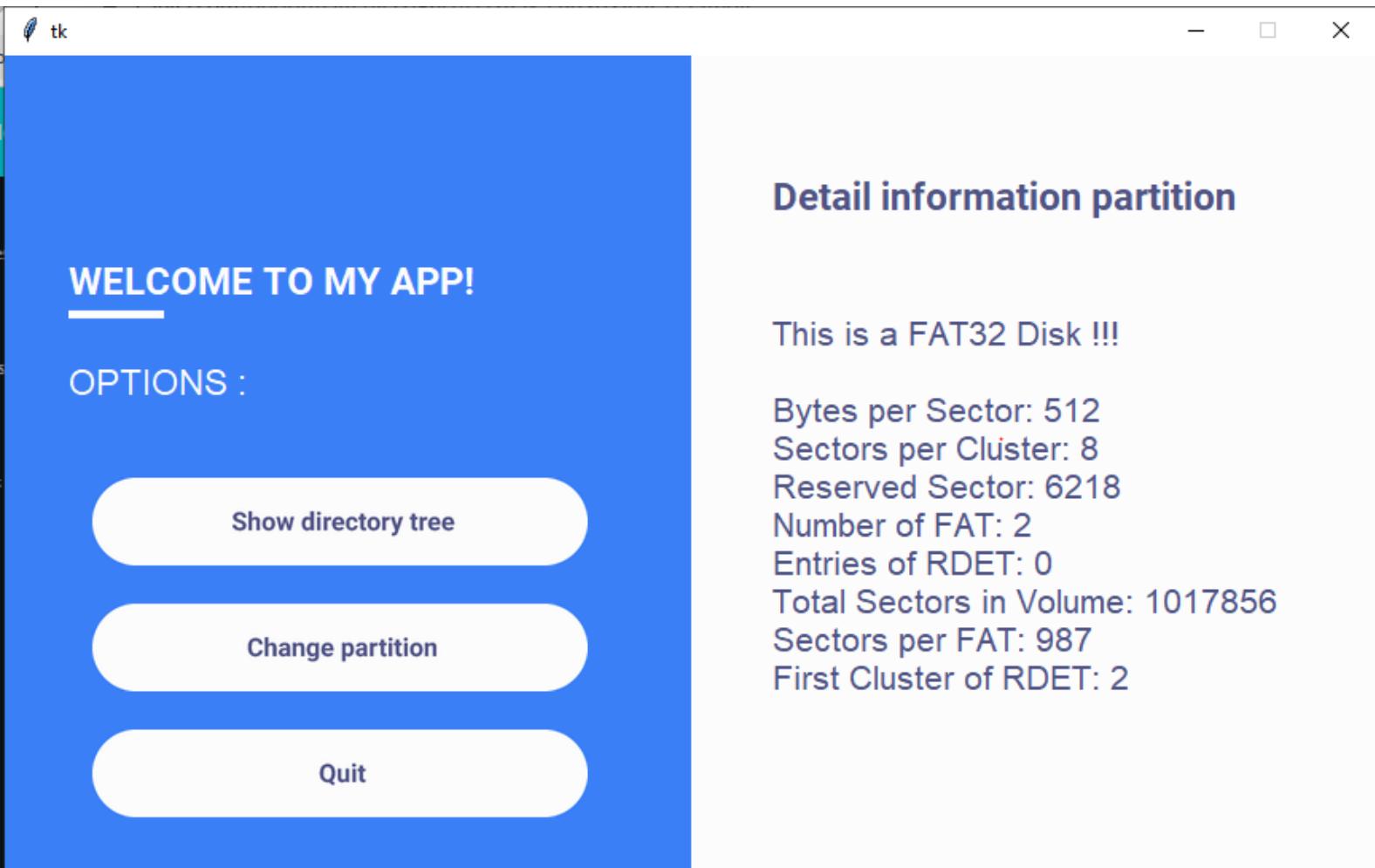
ĐƯA RA CÁC CHỌN LỰA
TIẾP THEO SAU KHI ĐÃ
ĐỌC ĐƯỢC PHÂN VÙNG

FAT 32

03 | VỚI LỰA CHỌN ĐẦU TIÊN ĐỌC THÔNG TIN CỦA PHÂN VÙNG (TRƯỜNG HỢP FAT)

CÁC THÔNG TIN ĐƯỢC XUẤT RA:

- Số byte trên 1 sector
- Số sector trên 1 cluster
- Reserved sector
- Số bảng FAT
- Số entries của RDET
- Tổng số sector trên volume
- Số sector mỗi bảng Fat
- Chỉ số cluster đầu tiên của RDET

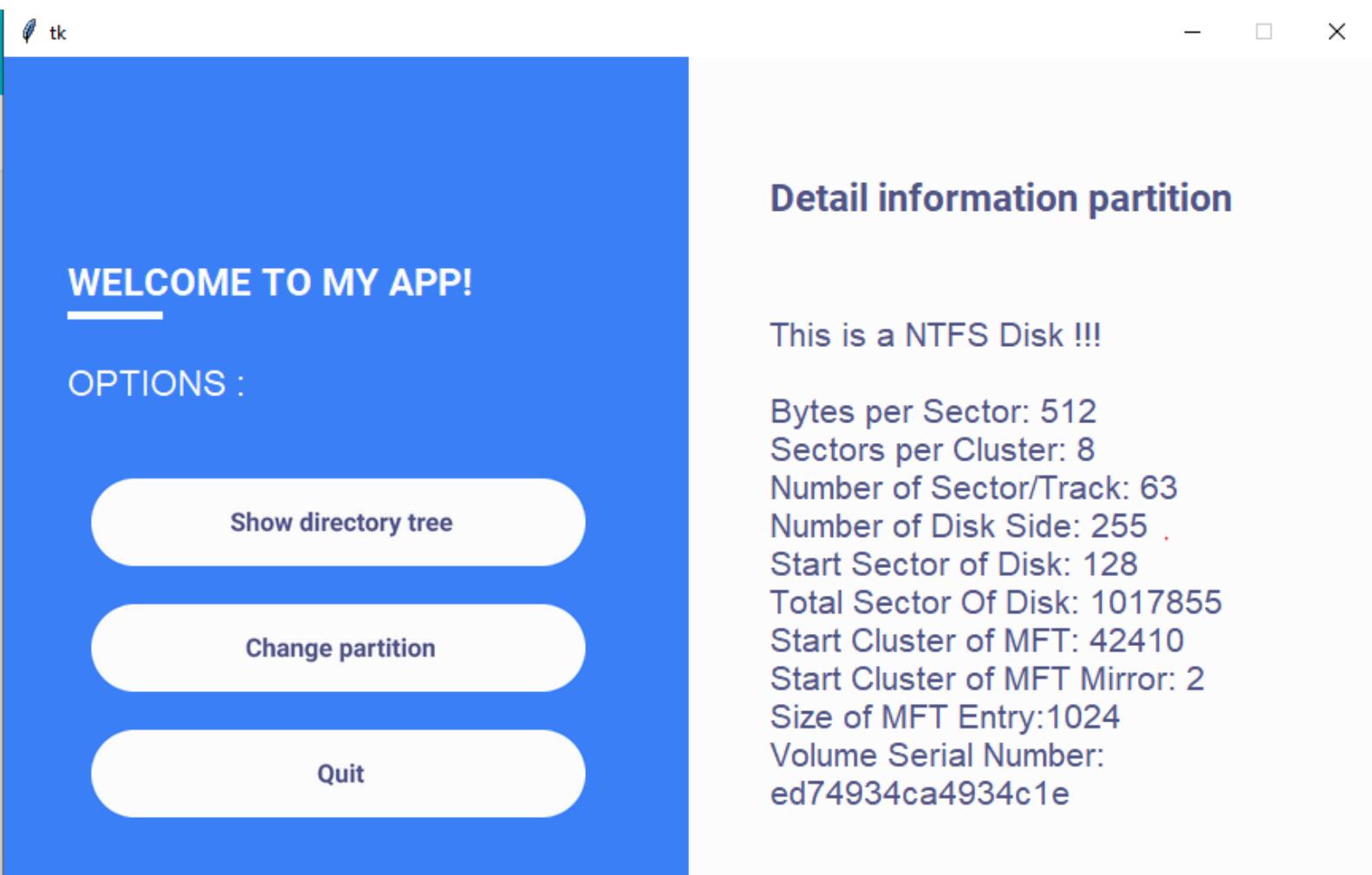


03 | VỚI LỰA CHỌN ĐẦU TIÊN ĐỌC THÔNG TIN CỦA PHÂN VÙNG (TRƯỜNG HỢP NTFS)

CÁC THÔNG TIN ĐƯỢC XUẤT RA:

- Số byte trên 1 sector
- Số sector trên 1 cluster
- Số sector/track
- Số mặt của volume
- Sector bắt đầu của volume
- Tổng số sector trên volume
- Chỉ số cluster bắt đầu của MFT
- Chỉ số cluster bắt đầu của MFT Mirror
- Kích thước của 1 MFT Entry
- Số hiệu của volume

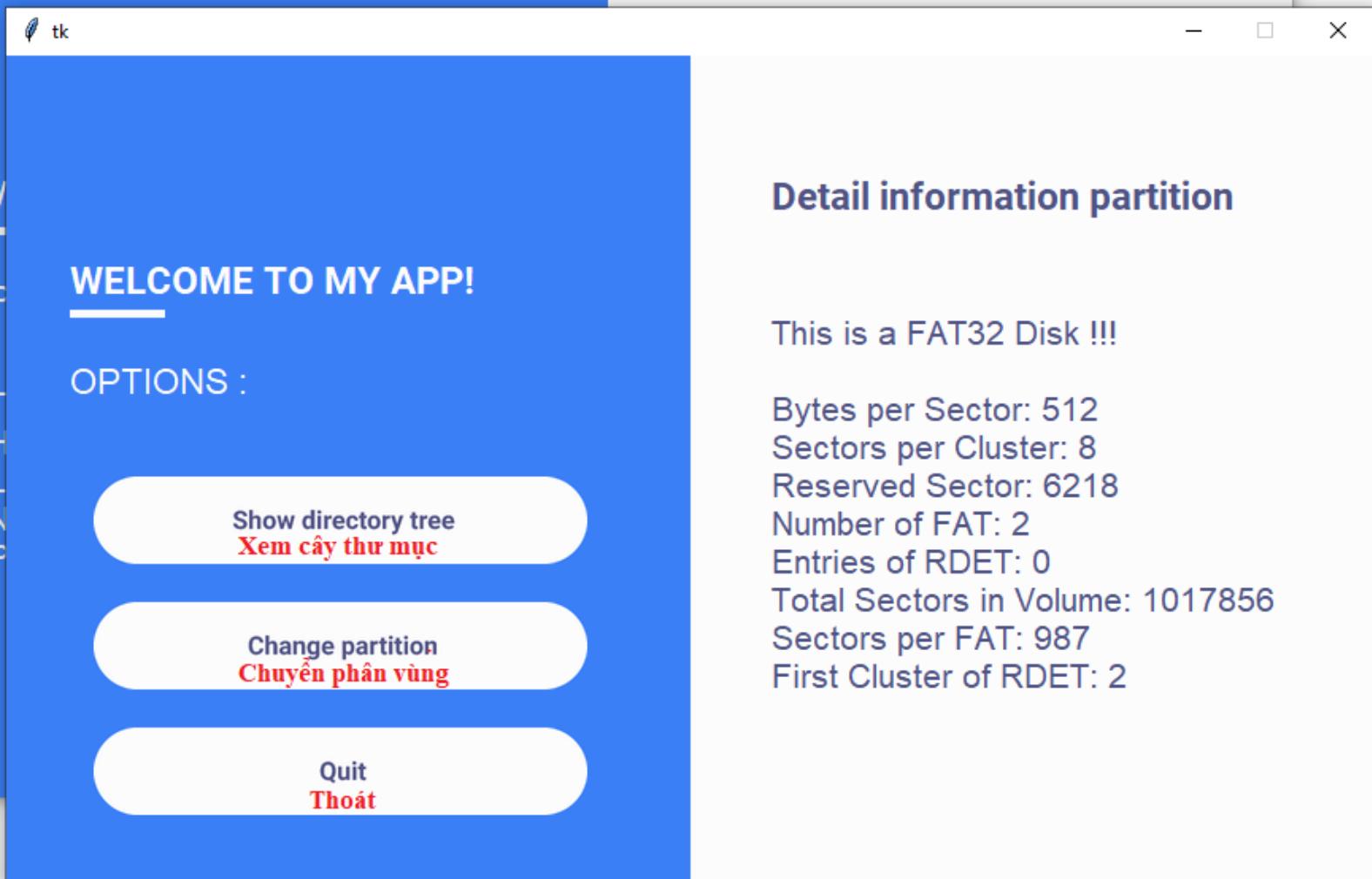
NTFS

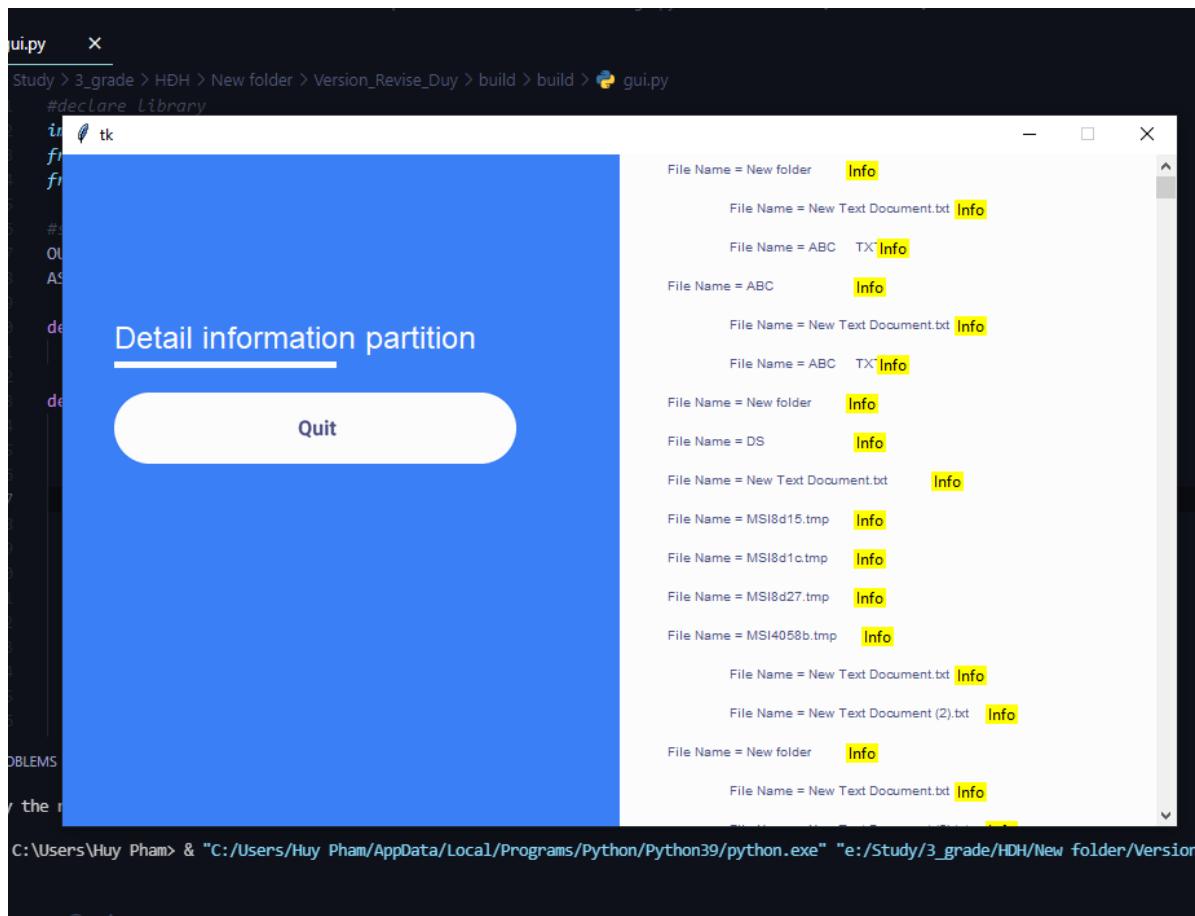


03

VỚI LỰA CHỌN ĐẦU TIÊN
ĐỌC THÔNG TIN CỦA
PHÂN VÙNG

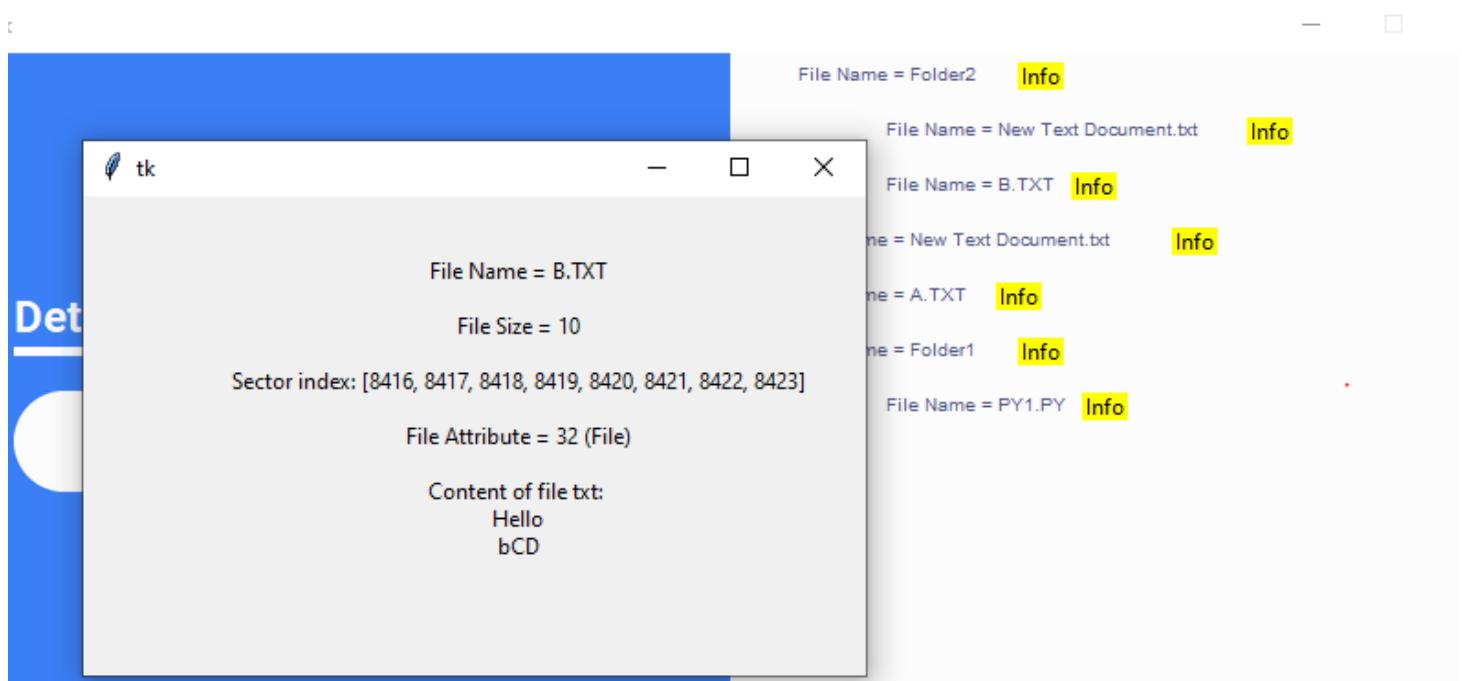
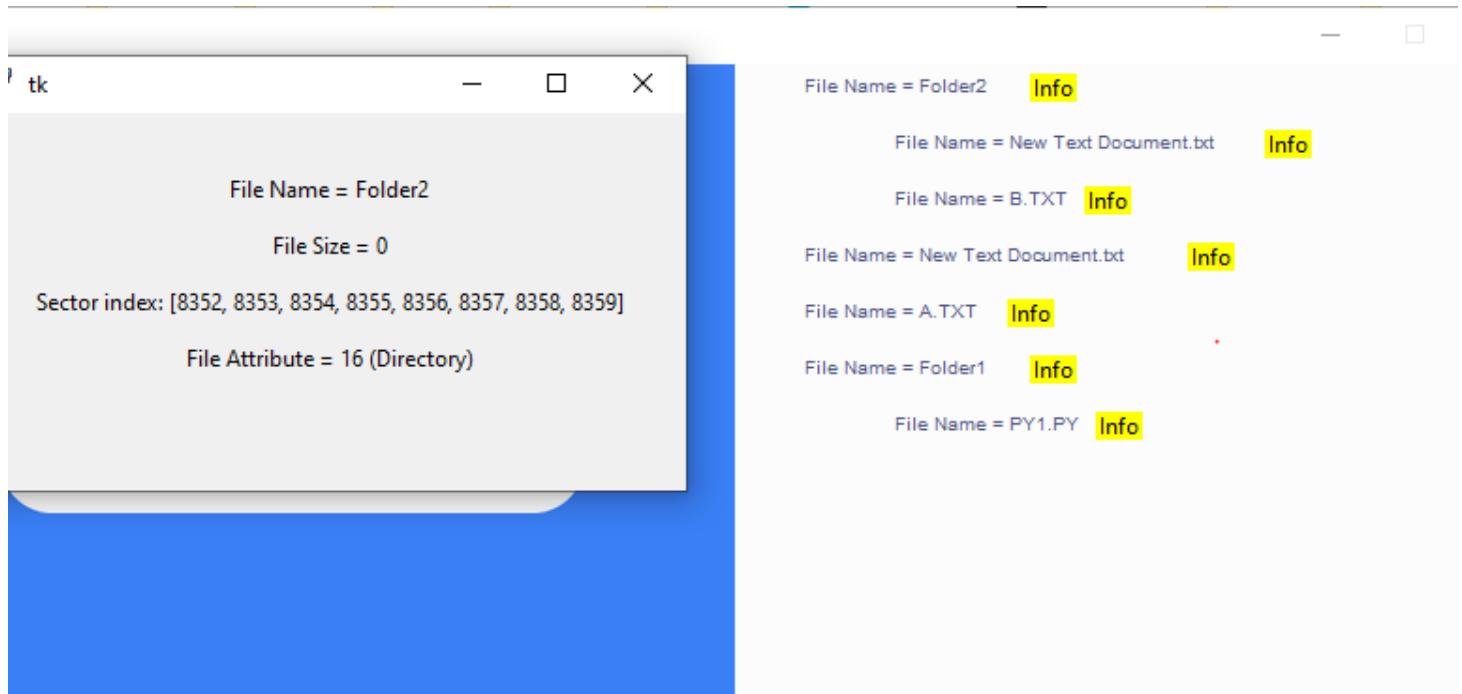
NGOÀI RA, TỪ LỰA CHỌN ĐẦU TIÊN CÓ
THỂ TIẾN THẮNG TỚI LỰA CHỌN THỨ 2.
HOẶC THAY ĐỔI PHÂN VÙNG
CŨNG NHƯ THOÁT KHỎI LỰA CHỌN 1





03 | LỰA CHỌN HIỂN THỊ CÂY THƯ MỤC

Với lựa chọn này cây thư mục sẽ được hiển thị như bên trong hình. kèm theo đó các file (kể cả thư mục) có kèm theo thông tin. Khi click vào (nút info kế bên tên của file) thì sẽ có thông tin như trang dưới



FAT 32

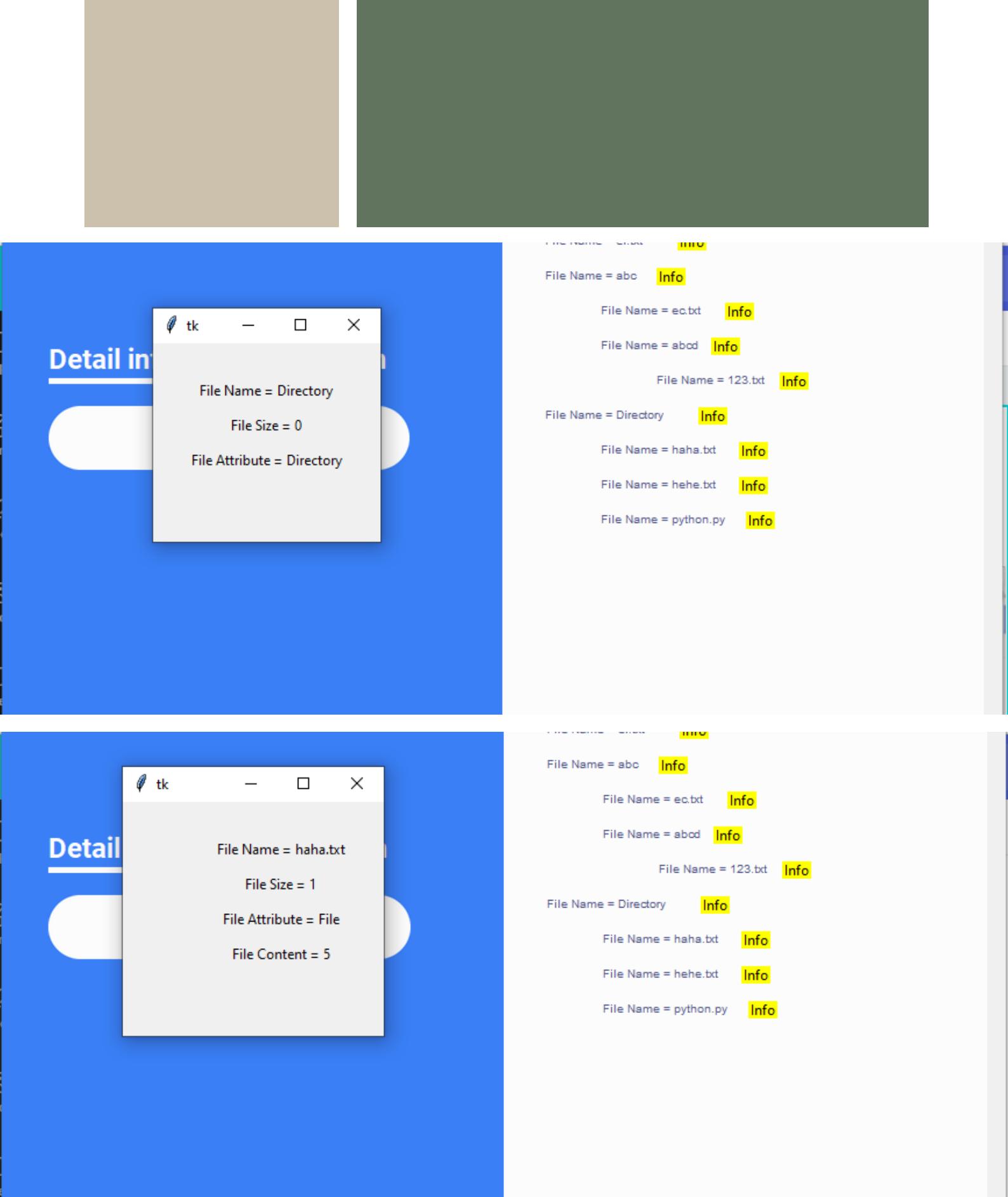
THÔNG TIN HIỂN THỊ (VỚI FAT32)

- File name
- File size
- Sector index
- File attribute
- Content of file(với file nội dung)

NTFS

THÔNG TIN HIỂN THỊ (VỚI NTFS)

- File name
- File size
- File attribute
- Content of file(với file nội dung)



VIDEO DEMO

VIDEO DEMO FAT & NTFS

Để thuận tiện cho việc truy cập
cũng như lưu trữ. Nhóm tụi em đã
quay video demo và dẫn link
youtube ở bên dưới

**ĐỒ ÁN FAT32 & NTFS
MÔN HỆ ĐIỀU HÀNH
KHOA CÔNG NGHỆ THÔNG TIN TRƯỜNG
ĐẠI HỌC KHOA HỌC TỰ NHIÊN - KHÓA
2019**

<https://youtu.be/D00ba3e1hjo>

NGUỒN THAM KHẢO

123DOC

<https://123docz.net/document/4849381-khao-sat-he-thong-fat32.htm>

123DOCZ

<https://text.123docz.net/document/4849417-khao-sat-he-thong-ntfs.htm>

WIKIPEDIA

<https://vi.wikipedia.org/wiki/FAT>

NTFS.COM

http://ntfs.com/ntfs_basics.htm

SLIDE

Slide bài giảng GV. Lê Việt Long

GITHUB

<https://github.com/shiva-prasad-reddy/Reading-and-writing-raw-data-to-a-FAT32-formatted-disk>



THAT'S ALL

THANK YOU
FOR READING

OPERATING SYMTEM