

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Mạc Lê Đức Minh - Phạm Phúc Lộc - Đặng
Mạnh Phúc - Lê Nguyễn Phương Nam - Võ
Đăng Huy

CHƯƠNG TRÌNH MÁY TÍNH ĐỌC
CÁC THÔNG TIN TRÊN PHÂN VÙNG
FAT32 VÀ NTFS

ĐỒ ÁN 1
QUẢN LÝ HỆ THỐNG TẬP TIN TRÊN WINDOWS

Tp. Hồ Chí Minh, tháng 03/2023

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN**

Mạc Lê Đức Minh - 21127526

Phạm Phúc Lộc - 21127100

Đặng Mạnh Phúc - 21127538

Lê Nguyễn Phương Nam - 21127647

Võ Đăng Huy - 21127062

**CHƯƠNG TRÌNH MÁY TÍNH ĐỌC
CÁC THÔNG TIN TRÊN PHÂN VÙNG
FAT32 VÀ NTFS**

ĐỒ ÁN 1

QUẢN LÝ HỆ THỐNG TẬP TIN TRÊN WINDOWS

GIÁO VIÊN HƯỚNG DẪN

THS. Lê Việt Long

Tp. Hồ Chí Minh, tháng 03/2023

Lời cảm ơn

Trước khi bắt đầu thảo luận về đề tài "Chương trình máy tính đọc các thông tin trên phân vùng FAT32 và NTFS", nhóm chúng em muốn bày tỏ lòng cảm ơn đến THS. Lê Viết Long - giảng viên đã trực tiếp truyền đạt kiến thức và kinh nghiệm quý báu, cùng giúp đỡ chúng em trong quá trình học tập.

Chúng em đã nỗ lực rất nhiều để hoàn thiện báo cáo này một cách tốt nhất có thể, tuy nhiên không tránh khỏi thiếu sót và hạn chế. Chúng em mong muốn nhận được những góp ý, nhận xét từ giảng viên để có thể cải thiện bài tập lần này, cũng như rút kinh nghiệm để các đồ án sau này đạt được hiệu quả tốt hơn.

Mục lục

Lời cảm ơn	i
Mục lục	ii
1 Giới thiệu	1
1.1 Đặt vấn đề	1
1.2 Mục tiêu	1
1.3 Cách tiếp cận	2
1.4 Tỷ lệ đóng góp và phân công công việc	3
1.4.1 Tỷ lệ đóng góp	3
1.4.2 Đánh giá mức độ hoàn thành	3
1.4.3 Phân công công việc	3
2 Các hệ thống quản lý tập tin khác	5
2.1 File System trên macOS	5
2.2 File System trên Linux	5
3 Tổng quan chương trình	7
3.1 Ý tưởng ứng dụng	7
3.2 Phân tích yêu cầu	8
3.3 Mô tả thuật toán	8
3.3.1 FAT32	8
3.3.2 NTFS	14
3.4 Thiết kế chi tiết thuật toán	19
3.4.1 FAT32	19
3.4.2 NTFS	23
4 Giao diện chương trình	27
4.0.1 Đối với FAT32	28
4.0.2 Đối với NTFS	30

Danh sách hình

3.1	Cấu trúc FAT32	9
3.2	Nội dung của boot sector	10
3.3	Master boot record	12
3.4	Dãy các phần tử (gọi là entry)	13
3.5	Cấu trúc phân vùng NTFS	14
3.6	Cấu trúc MFT	16
4.1	Các tập tin của ổ đĩa	27
4.2	Màn hình input	27
4.3	Volume Boot Sector	28
4.4	Thông tin trong Boot Sector	29
4.5	Thông tin cây thư mục của phân vùng	29
4.6	Partition Boot Sector	30
4.7	Thông tin chi tiết của phân vùng	31
4.8	Cây thư mục gốc gồm tên tập tin / thư mục	31
4.9	Thông tin trên cây thư mục (1)	32
4.10	Thông tin trên cây thư mục (2)	33
4.11	Thông tin trên cây thư mục (3)	34

Danh sách bảng

1.1	Bảng tỷ lệ đóng góp của mỗi thành viên	3
1.2	Bảng phân công công việc	4
3.1	Các thành phần trong entry chính	14

Chương 1

Giới thiệu

1.1 Đặt vấn đề

Xây dựng chương trình máy tính đọc các thông tin trên phân vùng FAT32 và NTFS là một đề tài nghiên cứu mang tính ứng dụng cao. Với sự phát triển không ngừng của công nghệ thông tin, việc nghiên cứu và xây dựng các chương trình hỗ trợ cho việc quản lý và xử lý dữ liệu trên máy tính ngày càng được đặt ra yêu cầu cao hơn.

Phân vùng FAT32 và NTFS là hai loại hệ thống tập tin phổ biến trên hệ điều hành Windows. Tuy nhiên, việc đọc các thông tin từ những phân vùng này vẫn là một thách thức đối với nhiều người sử dụng. Để giải quyết vấn đề này, chúng em đã tiến hành nghiên cứu và xây dựng một chương trình máy tính đọc các thông tin trên phân vùng FAT32 và NTFS.[1]

Chương trình này được phát triển bằng ngôn ngữ lập trình C++, sử dụng các thư viện và công nghệ tiên tiến để đọc các thông tin từ phân vùng FAT32 và NTFS một cách nhanh chóng và chính xác. Chương trình có khả năng đọc các thông tin như tên tập tin, kích thước, thời gian sửa đổi, ngày tạo và nhiều thông tin khác từ những phân vùng này.

1.2 Mục tiêu

Mục tiêu của đề tài là tập trung vào việc thu thập thông tin từ khu vực đầu tiên (Boot Sector) của mỗi phân vùng và bảng thư mục gốc (RDET) trong hai hệ thống quản lý tập tin là FAT32 và NTFS. Chương trình sẽ được thiết kế để truy xuất và đọc các thông tin này một cách chính xác và hiệu quả, giúp người dùng dễ dàng quản lý và sử dụng dữ liệu trên các phân vùng này.

Để đạt được mục tiêu thu thập thông tin từ khu vực đầu tiên và bảng

thư mục gốc của các phân vùng FAT32 và NTFS, chương trình sẽ tập trung vào việc lấy các thông số quan trọng như byte per sector, sector per cluster, số lượng sector, cluster... thông qua việc phân tích các giá trị trong Boot Sector và các thuộc tính của tập tin trong bảng thư mục gốc (RDET). Những thông tin này rất quan trọng để chương trình có thể xác định được cấu trúc của phân vùng và truy xuất được các tập tin, thư mục trên đó một cách chính xác. Ngoài ra, chương trình cũng sẽ cung cấp cho người dùng các thông tin liên quan đến khối lượng dữ liệu trên phân vùng, dung lượng khả dụng và các thông tin khác để người dùng có thể quản lý tập tin và thư mục trên phân vùng một cách dễ dàng và hiệu quả.

1.3 Cách tiếp cận

Để lấy thông tin của Boot Sector, chương trình sẽ đọc các giá trị trong Boot Sector của phân vùng FAT32 hoặc NTFS. Thông tin này sẽ giúp chương trình hiểu được cấu trúc của phân vùng đang được đọc và tìm kiếm các thông tin khác trong phân vùng đó.

Sau khi có thông tin về Boot Sector, chương trình sẽ tiếp tục đọc bảng thư mục gốc (RDET) của phân vùng để lấy các thuộc tính của tập tin như tên tập tin, kích thước tập tin, địa chỉ cluster đầu tiên của tập tin và nhiều thuộc tính khác. Quá trình này sẽ được thực hiện thông qua việc tìm kiếm thông tin về tập tin trong bảng thư mục gốc và lấy các thuộc tính của tập tin đó.

Từ các thông tin lấy được từ Boot Sector và RDET, chương trình có thể phân tích và hiển thị các thông tin cần thiết về phân vùng và các tập tin trong đó. Quá trình lấy thông tin này sẽ giúp người dùng có thể hiểu được cấu trúc của phân vùng và quản lý các tập tin một cách hiệu quả hơn.

1.4 Tỷ lệ đóng góp và phân công công việc

1.4.1 Tỷ lệ đóng góp

Bảng 1.1: Bảng tỷ lệ đóng góp của mỗi thành viên

MSSV	Thành viên	Tỷ lệ đóng góp
21127526	Mạc Lê Đức Minh	20%
21127100	Phạm Phúc Lộc	20%
21127538	Đặng Mạnh Phúc	20%
21127647	Lê Nguyễn Phương Nam	20%
21127062	Võ Đăng Huy	20%

1.4.2 Đánh giá mức độ hoàn thành

Đã hoàn thành đầy đủ các yêu cầu trong project, bao gồm đọc và hiển thị các thông tin chi tiết của một phân vùng và hiển thị thông tin cây thư mục của phân vùng. Mọi chức năng đều được kiểm thử kỹ lưỡng và hoạt động chính xác. Ngoài ra, mã nguồn được viết một cách rõ ràng, dễ hiểu và tuân thủ các quy tắc lập trình tốt nhất. Do đó, đánh giá mức độ hoàn thành của project là 100%.

1.4.3 Phân công công việc

Bảng 1.2: Bảng phân công công việc

Công việc	Người phụ trách	
Thảo luận và đưa ra các hàm chính của chương trình	Cả nhóm	
Thảo luận và nhất quán thuật toán truy xuất tập tin.	Cả nhóm	
Xử lý đường dẫn của phân vùng cần truy xuất	Mạc Lê Đức Minh	
File System	NTFS	FAT32
Đọc và lưu thông tin các thuộc tính của Volume trong vùng Boot Sector	Mạc Lê Đức Minh	Đặng Mạnh Phúc
Chia đường dẫn thành các thành phần để xác định các thư mục con.	Lê Nguyễn Phương Nam	
Đọc bảng RDET để lấy thông tin về tất cả các tập tin/ thư mục trong phân vùng	Võ Đăng Huy	Phạm Phúc Lộc
Duyệt các thư mục con và lấy thông tin tập tin/ thư mục cần truy xuất.	Võ Đăng Huy	Phạm Phúc Lộc
Hiển thị thông tin về tập tin/ thư mục bao gồm: tên, kích thước, đường dẫn, vị trí trên đĩa cứng.	Lê Nguyễn Phương Nam	Phạm Phúc Lộc
Sử dụng bảng FAT để tìm các cluster chứa nội dung của tập tin.	Lê Nguyễn Phương Nam	Đặng Mạnh Phúc
Xác định offset của tập tin trong từng \cluster.	Mạc Lê Đức Minh	Đặng Mạnh Phúc
Viết mô tả thuật toán theo yêu cầu đề bài	Phạm Phúc Lộc	Đặng Mạnh Phúc

Chương 2

Các hệ thống quản lý tập tin khác

File system là một thành phần quan trọng của mỗi hệ thống máy tính. Nó đóng vai trò quản lý dữ liệu và các tệp tin trong hệ thống. Trong chương này, chúng ta sẽ tìm hiểu về các hệ thống tập tin phổ biến trên hai hệ điều hành phổ biến là macOS và Linux.

2.1 File System trên macOS

MacOS được phát triển bởi Apple và được cung cấp kèm theo các dòng máy tính Mac. File system mặc định trên macOS là APFS (Apple File System). Đây là một hệ thống tập tin hiện đại và tối ưu cho các ổ đĩa flash và SSD. APFS được thiết kế để giải quyết các vấn đề liên quan đến bảo mật, đồng bộ hóa và nhanh chóng truy cập dữ liệu. [2]

Ngoài ra, macOS cũng hỗ trợ các hệ thống tập tin khác như HFS+ (Hierarchical File System Plus), FAT32 và ExFAT. HFS+ là một hệ thống tập tin truyền thống của Apple, được sử dụng trên các phiên bản trước của macOS. FAT32 là một hệ thống tập tin đơn giản và phổ biến được sử dụng trên các ổ đĩa di động và thiết bị đa phương tiện. ExFAT được sử dụng để tương thích với các thiết bị khác nhau và có khả năng xử lý các tập tin lớn.[3]

2.2 File System trên Linux

Linux là một hệ điều hành mã nguồn mở phổ biến, được sử dụng trên nhiều loại thiết bị, từ máy tính để bàn đến máy chủ. Linux hỗ trợ nhiều hệ thống tập tin khác nhau như Ext2, Ext3, Ext4, XFS, JFS, Btrfs, ReiserFS

và NTFS.[4]

Trong số đó, Ext4 là hệ thống tập tin phổ biến nhất trên Linux và được coi là phiên bản nâng cấp của Ext3. Nó có khả năng quản lý các tập tin lớn hơn và tốc độ đọc/ghi nhanh hơn so với các hệ thống tập tin trước đó. XFS là một hệ thống tập tin có hiệu suất cao, được thiết kế để xử lý các tập tin lớn. JFS là một hệ thống tập tin có thể khôi phục được trong trường hợp hệ thống bị lỗi và Btrfs là một hệ thống tập tin mới, được thiết kế để hỗ trợ các tính năng mới như khả năng snapshot và mã hóa dữ liệu.

[5]

XFS là một loại file system phổ biến trên các máy chủ Linux, nó hỗ trợ dung lượng lớn và tốc độ đọc/ghi cao. JFS và ReiserFS cũng là những loại file system tương đối phổ biến trên Linux, nhưng đã ít được sử dụng hơn sau khi Ext4 được giới thiệu. [6]

Chương 3

Tổng quan chương trình

3.1 Ý tưởng ứng dụng

Có rất nhiều ứng dụng của việc lấy các thuộc tính của tập tin trong bảng thư mục gốc và các thông số của Boot Sector trong việc quản lý và khai thác dữ liệu trên các phân vùng FAT32 và NTFS. Dưới đây là một số ý tưởng về việc áp dụng các thông tin này:

1. Khôi phục dữ liệu: Trong trường hợp một phân vùng bị hỏng hoặc xảy ra lỗi, việc lấy thông tin từ Boot Sector và bảng thư mục gốc sẽ giúp phục hồi dữ liệu. Dữ liệu có thể bị mất hoặc bị lỗi trong trường hợp phân vùng bị hỏng, vì vậy việc lấy thông tin này có thể giúp định vị và khôi phục lại các tập tin bị mất.[7]
2. Kiểm tra tính toàn vẹn của hệ thống tập tin: Thông tin về byte per sector, sector per cluster, số lượng sector và cluster của Boot Sector sẽ giúp xác định tính toàn vẹn của hệ thống tập tin. Nếu các thông số này không đúng, có thể làm ảnh hưởng đến khả năng truy cập và lưu trữ dữ liệu trên phân vùng. Việc kiểm tra tính toàn vẹn của hệ thống tập tin giúp đảm bảo an toàn cho dữ liệu.
3. Tối ưu hóa lưu trữ: Các thông số byte per sector, sector per cluster, số lượng sector và cluster của Boot Sector cũng cung cấp thông tin về cách hệ thống tập tin được tổ chức và lưu trữ dữ liệu. Dựa trên thông tin này, có thể tối ưu hóa việc lưu trữ dữ liệu và cải thiện hiệu suất truy cập dữ liệu trên phân vùng.
4. Phân tích dữ liệu: Việc lấy các thuộc tính của tập tin trong bảng thư mục gốc cũng cung cấp nhiều thông tin quan trọng về dữ liệu, chẳng hạn như kích thước tập tin, định dạng và thông tin tác giả.

Những thông tin này có thể được sử dụng để phân tích dữ liệu và đưa ra các quyết định về quản lý và khai thác dữ liệu.

3.2 Phân tích yêu cầu

Yêu cầu của đề án là đọc các thông tin chi tiết của một phân vùng và hiển thị thông tin cây thư mục của phân vùng đó. Để thực hiện được yêu cầu này, chương trình cần lấy các thông tin từ Boot Sector hoặc Partition Boot Sector (đối với phân vùng NTFS) để xác định các thông số như byte per sector, sector per cluster, số lượng sector, cluster của phân vùng. Với phân vùng FAT32, chương trình cần đọc và phân tích bảng RDET và bảng FAT để lấy các thông tin về cây thư mục và các tập tin trong phân vùng. Với phân vùng NTFS, chương trình cần đọc và phân tích Master File Table để lấy các thông tin tương tự.

Khi hiển thị thông tin cây thư mục của phân vùng, chương trình sẽ hiển thị cây thư mục gốc gồm tên tập tin / thư mục, trạng thái, kích thước (nếu có) và chỉ số sector lưu trữ trên đĩa cứng. Nếu đối tượng là tập tin có phần mở rộng là txt, chương trình sẽ hiển thị nội dung tập tin đó. Đối với các loại tập tin khác, chương trình sẽ hiển thị thông báo để sử dụng phần mềm tương thích để đọc nội dung. Nếu đối tượng là thư mục, chương trình cho phép hiển thị cây thư mục con và hiển thị các thông tin tương tự như cây thư mục gốc.

3.3 Mô tả thuật toán

3.3.1 FAT32

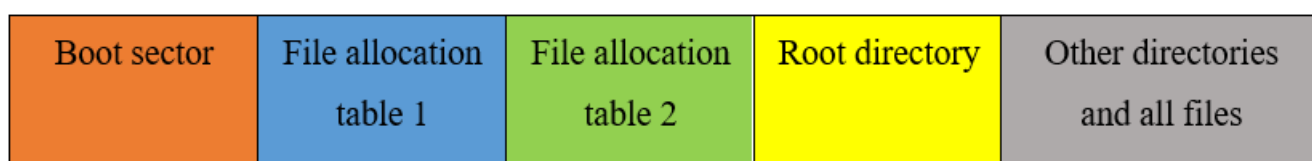
Giới thiệu

FAT32 là một định dạng đĩa hoặc hệ thống tệp được sử dụng để sắp xếp các tệp được lưu trữ trên ổ đĩa, được Microsoft giới thiệu vào năm 1996 với hệ điều hành Windows 95 OEM Service Releases 2 (OSR2), là một phần mở rộng của hệ thống tệp FAT16.

Mục đích của FAT32 là khắc phục những hạn chế của FAT16 và thêm hỗ trợ cho phương tiện lớn hơn. Sử dụng không gian địa chỉ 32 bit, FAT32

hỗ trợ nhiều cluster trên một partition hơn, do vậy không gian đĩa cứng được tận dụng nhiều hơn. Kích thước ổ đĩa tối đa tiêu chuẩn cho FAT16 là 2GB, FAT32 tăng giới hạn này lên 2TB đáng kể bằng cách tăng số lượng bit được sử dụng để đánh địa chỉ cụm, giúp tăng hiệu suất và tính linh hoạt. FAT32 dự trữ 32 bit cho mỗi mục nhập cụm, trong đó 28 bit thấp hơn thực sự được sử dụng để đánh địa chỉ các cụm. Tuy nhiên, FAT32 có nhược điểm là tính bảo mật và khả năng chịu lỗi không cao.

Cấu trúc FAT32



Hình 3.1: Cấu trúc FAT32

PARTITION BOOT SECTOR

Đây là một sector đặc biệt nằm ở đầu mỗi partition trên đĩa, chứa thông tin về cấu hình đĩa, kích thước, và loại hệ điều hành được cài đặt cùng với mã lệnh khởi động mới cho hệ điều hành. Để tránh các ứng dụng độc hại như virus hiệu chỉnh nội dung sector này, các máy tính đời mới thường được trang bị BIOS có chức năng bảo vệ boot sector. Khi các ứng dụng cố gắng thay đổi nội dung của boot sector, chúng sẽ phải yêu cầu BIOS làm việc này. Nếu sector bị hiệu chỉnh là boot sector, BIOS sẽ hiển thị thông báo để người dùng quyết định cho phép hoặc cấm chức năng bảo vệ này. Tùy theo nhu cầu sử dụng, người dùng có thể vào BIOS Setup để thay đổi cài đặt bảo vệ boot sector.

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)

Hình 3.2: Nội dung của boot sector

-Máy tính sử dụng boot sector để thực thi các chỉ dẫn trong suốt quá trình khởi động. Quá trình này được mô tả như sau:

+Đầu tiên, BIOS và CPU sẽ thực hiện power-on self test (POST).

+Sau đó, BIOS sẽ tìm một thiết bị khởi động.

+Nếu thiết bị khởi động là ổ cứng, BIOS sẽ nạp Master Boot Record (MBR). Mã lệnh của MBR sẽ nạp boot sector của phân vùng active và chuyển quyền điều khiển cho sector này. Trên Windows 2000, mã thực thi của boot sector sẽ tìm và nạp NTLDR vào bộ nhớ và chuyển quyền điều khiển cho file này.

+Nếu trong ổ đĩa mềm có đĩa có thể khởi động, được định dạng với các

tệp hệ thống, thì BIOS sẽ nạp sector đầu tiên (boot sector) của đĩa vào bộ nhớ. Mã lệnh của boot sector sẽ nạp tệp `io.sys` - một tệp hệ thống chính của MS-DOS - vào bộ nhớ và chuyển quyền điều khiển cho tệp này.

+Sau khi hệ điều hành được nạp vào bộ nhớ, quyền điều khiển sẽ được chuyển sang hệ điều hành và hệ thống sẽ được điều khiển bởi hệ điều hành đó.

Bảng FAT

Bảng FAT1 và FAT2 chứa thông tin về việc cấp phát và định vị các file, cho phép hệ điều hành truy xuất đến các file một cách chính xác. Bên cạnh đó, bảng này cũng cung cấp thông tin về dung lượng còn trống trên đĩa và đánh dấu các vị trí BAD trên đĩa.

Bảng FAT là một bảng ánh xạ toàn bộ các cluster trên đĩa, nhưng nó chỉ chứa thông tin về vị trí các cluster trên ổ cứng mà không lưu trữ dữ liệu.

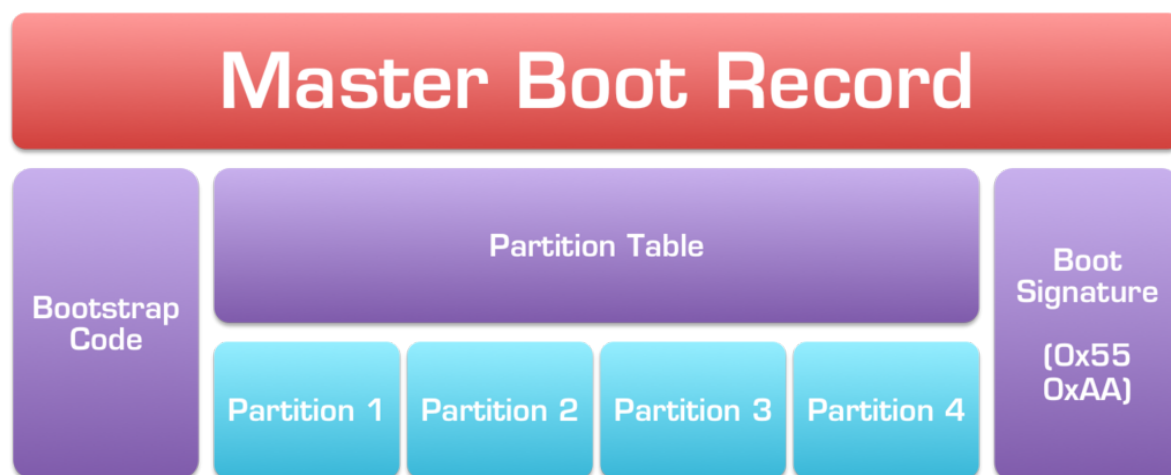
Root folder

Bảng thư mục gốc là một loại bảng thông tin trong hệ thống tập tin giống như bảng thư mục của một cuốn sách. Nó chứa thông tin liên quan đến các tập tin hoặc thư mục như tên, ngày giờ tạo, thuộc tính và vị trí lưu trữ trên ổ đĩa.

Other files or folders

Đây là nơi lưu trữ của các file và thư mục con trong hệ thống tập tin.

Master Boot Record(MBR)



Hình 3.3: Master boot record

MBR (Master Boot Record) là một phần quan trọng của ổ đĩa cứng, chứa thông tin về các phân vùng trên ổ đĩa. Mỗi hệ điều hành có cách tạo MBR riêng. Ví dụ, trong hệ điều hành WinNT4 và Win2k, MBR tương đương với file boot.ini. Khi khởi động, chương trình bootloader của hệ điều hành sẽ đọc nội dung của MBR để tìm và load hệ điều hành cần thiết cho người dùng.

Tuy nhiên, đối với Windows 98 và Windows ME, chúng không có boot-loader như NT bootloader. Thay vào đó, hệ điều hành sẽ mặc định load hệ điều hành đầu tiên trong phân vùng đầu tiên trên ổ đĩa cứng.

Thông thường, MBR được lưu trữ ở đầu ổ đĩa cứng và ở phân vùng nhỏ nhất của nó (phân vùng đầu tiên trên ổ đĩa số 0). MBR rất quan trọng để BIOS tìm đến khi khởi động máy tính.

MBR chứa một phần môi khởi động cùng với bảng phân vùng (partition table) gồm 4 phần, để lưu trữ thông tin về các phân vùng chính (primary partition) trên ổ đĩa. Thông tin này giúp máy tính xác định được số lượng phân vùng trên ổ đĩa, vị trí và kích thước của chúng.

Bảng thư mục gốc (RDET - Root Directory Entry Table)

Trong hệ thống tập tin FAT, có một thông số quan trọng khác liên quan đến cách truy cập vùng FAT (hay các FAT Entry). Mỗi Cluster với số N sẽ có một Entry tương ứng (1-1) được lưu trữ ở đâu đó trong vùng

FAT. Với FAT16/FAT32, việc truy cập vùng FAT khá đơn giản, bởi vì FAT Entry được biểu diễn như một mảng 1 chiều các số nguyên 16 bit. Tuy nhiên, so với mảng trên bộ nhớ, FAT Entry có một vài điểm khác biệt. Đầu tiên, các chỉ số không nằm trên một vùng liên tục, ta chỉ có thể chắc chắn từ Sector đầu tiên chứa FAT, thì toàn bộ FAT sẽ nằm trên nhiều Sector liên tục nhau. [8]

Entry	1	2	...	16	17	18	...	32	33	...	208	209	210	...	224	225	226	...
Sector	1				2				...				14				...	

Hình 3.4: Dãy các phần tử (gọi là entry)

Mỗi tập tin hoặc thư mục trong hệ thống tập tin FAT có thể chiếm 1 hoặc nhiều entry. Entry đầu tiên của mỗi tập tin hoặc thư mục cho biết trạng thái hiện tại của entry này, có thể là:

- 0 nếu entry đang trống.
- E5h nếu tập tin hoặc thư mục chiếm entry này đã bị xóa.
- Giá trị khác nếu entry đang chứa thông tin của tập tin hoặc thư mục.

Trong hệ thống tập tin FAT, có hai loại entry bao gồm:

- Entry chính: chứa các thông tin của tập tin hoặc thư mục.
- Entry phụ: chỉ chứa tên của tập tin

Root Directory luôn tồn tại trong ổ đĩa và nằm ở tầng cao nhất trong hệ thống tập tin FAT. Trong FAT12/FAT16, Root Directory nằm ngay sau vùng FAT và có kích thước cố định. Trong khi đó, trong hệ thống tập tin FAT32, cả các tập tin và thư mục con được lưu trữ trong Root Directory.

Bảng 3.1: Các thành phần trong entry chính

Offset	Bytes	Nội dung
00	8	Tên tập tin
08	3	Phần mở rộng
0B	1	Thuộc tính
1A	2	Cluster bắt đầu
1C	4	Dung lượng tập tin

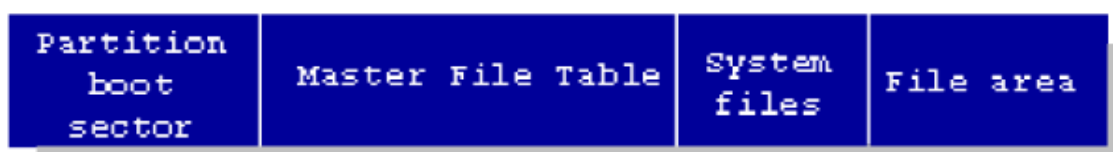
3.3.2 NTFS

Giới thiệu

Hệ thống tập tin NTFS (New Technology File System) được phát triển vào cuối những năm 1990 bởi Microsoft nhằm tạo ra một sản phẩm công nghệ đáng tin cậy và hiện đại cho hệ điều hành của họ. Một trong những điểm yếu lớn của MS-DOS và Windows 3.x là sự phụ thuộc vào hệ thống tập tin FAT. Để ngăn chặn sự suy giảm của Windows NT, Microsoft đã tạo ra một hệ thống tập tin mới không dựa trên FAT. Kết quả là NTFS được tạo ra với một triết lý riêng biệt không phụ thuộc vào hệ thống tập tin FAT cũ.

Trong suốt quá trình phát triển từ hệ điều hành Windows NT, Windows 2000, Windows XP đến Windows Server 2003, NTFS đã được cải tiến để đáp ứng nhu cầu lưu trữ lớn và yêu cầu bảo mật ngày càng cao.

Cấu trúc phân vùng NTFS



Hình 3.5: Cấu trúc phân vùng NTFS

PARTITION BOOT SECTOR

Partition Boot Sector là một khu vực quan trọng đầu tiên trong một phân vùng đĩa của hệ thống tập tin NTFS (New Technology File System). Khu vực này nằm ở offset 0 trên phân vùng và có kích thước 512 byte. Nó chứa các thông tin quan trọng về cấu trúc của phân vùng NTFS, bao gồm cả bảng phân vùng, file system metadata, ...

Partition Boot Sector bao gồm nhiều thông tin quan trọng như BIOS Parameter Block (BPB), kích thước phân vùng, địa chỉ LBA (Logical Block Addressing) của phân vùng, vị trí của Master File Table (MFT), địa chỉ các cluster, v.v. Nó cũng bao gồm một mã lệnh khởi động (boot code), được sử dụng để khởi động hệ thống từ phân vùng NTFS.

Vị trí	Kích thước	Tên
0x00	3 bytes	Lệnh nhảy
0x03	8 bytes	OEM ID
0x0B	25 bytes	BPB
0x24	48 bytes	Dành cho phần mở rộng
0x54	426 bytes	Chứa lệnh thực thi
0x01FE	2 bytes	Tín hiệu kết thúc

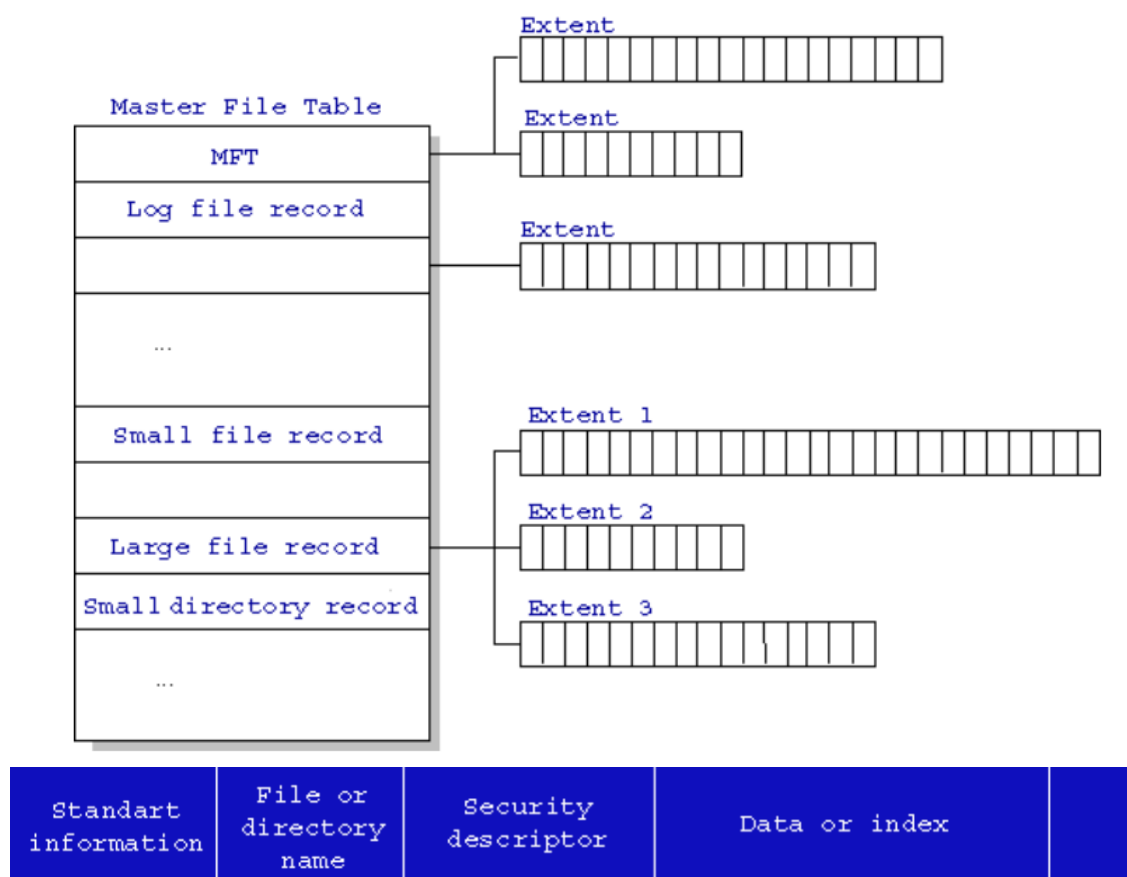
Trên ổ đĩa NTFS, trường dữ liệu được đánh theo trình tự từ BPB của phần mở rộng. Dữ liệu ở trường NTLDR (New Technology Loader) được kích hoạt trong suốt quá trình khởi động, trên ổ đĩa NTFS thì MFT không được đặt ở một sector xác định trước. Đó là lý do tại sao NTFS có thể di chuyển MFT nếu sector chứa MFT bị hỏng.

MASTER FILE TABLE (MFT)

Master File Table (MFT) là một cấu trúc dữ liệu đặc biệt trong hệ thống file NTFS, nơi lưu trữ thông tin chi tiết về các tập tin và thư mục trên đĩa cứng. Mỗi tập tin và thư mục trên hệ thống NTFS đều có một mục tương ứng trong MFT, bao gồm các thông tin như tên tập tin, kích thước, thời gian tạo và sửa đổi, quyền truy cập và địa chỉ vật lý của tập tin trên đĩa cứng.

MFT cũng chứa các thông tin về các phân vùng, file system metadata và các file ẩn trên hệ thống NTFS. Nó được lưu trữ trong một số sector đặc biệt trên đĩa cứng, với kích thước phụ thuộc vào kích thước đĩa cứng

và số lượng tập tin và thư mục được lưu trữ trong hệ thống file NTFS.



Hình 3.6: Cấu trúc MFT

Trong hệ thống tập tin NTFS, mỗi file được đại diện bởi một bản ghi trong một file đặc biệt gọi là bảng thông tin tập tin chính (MFT). Để đảm bảo tính toàn vẹn của MFT, NTFS dành riêng 16 bản ghi đầu tiên cho các thông tin quan trọng, trong đó bản ghi đầu tiên chứa thông tin về chính MFT. Một bản sao của MFT cũng được lưu trữ trong 16 bản ghi đầu tiên. Trong trường hợp bản ghi đầu tiên của MFT bị hỏng, NTFS sẽ đọc bản ghi thứ hai để tìm bản sao của MFT, còn được gọi là MFT mirror file. Vị trí của các segment dữ liệu cho MFT và MFT mirror được lưu trữ trong boot sector. Một bản sao của boot sector được lưu trữ giữa đĩa để đảm bảo tính toàn vẹn của dữ liệu.

Bản ghi thứ ba trong MFT là Log File, được sử dụng để ghi lại các hoạt động thay đổi trong MFT và giúp khôi phục dữ liệu khi có sự cố xảy ra. Các bản ghi tiếp theo trong MFT (bắt đầu từ bản ghi thứ 17) dành cho mỗi tập tin và thư mục trên đĩa, chứa thông tin về các thuộc tính của

tập tin và thư mục, vị trí của các phân vùng dữ liệu của tập tin và thư mục trên đĩa, và các thông tin khác.

MFT sử dụng phần vùng trống trong các bản ghi để lưu trữ thông tin về các file và thuộc tính của chúng. Nếu dung lượng của file nhỏ hơn hoặc bằng 1.5KB, thì file đó có thể được lưu trữ toàn bộ trong một bản ghi MFT duy nhất. Các file lớn hơn được lưu trữ trên các vùng dữ liệu riêng biệt và các thông tin về file này được lưu trữ trong các bản ghi MFT tương ứng.

Các bản ghi của thư mục cũng được lưu trong MFT, tuy nhiên chúng không chứa dữ liệu của thư mục mà chỉ chứa thông tin về chỉ số của thư mục. Các thư mục có kích thước nhỏ có thể được lưu trữ hoàn toàn trong MFT, trong khi các thư mục lớn hơn thường được tổ chức theo cấu trúc cây nhị phân. Tuy nhiên, cũng có trường hợp các bản ghi của thư mục không thể chứa trong MFT.

FILE METADATA

File metadata là các thông tin về file hoặc thư mục, bao gồm các thuộc tính như tên file, kích thước, ngày tạo, ngày sửa đổi, quyền truy cập, ấn bản, mã hóa và vị trí trên ổ đĩa. Trong hệ thống tập tin NTFS, các thông tin này được lưu trữ trong MFT và các file metadata được đặt tên bắt đầu bằng ký tự đặc biệt "\$". Ví dụ: \$MFT, \$LogFile, \$Volume.

FILE SYSTEM DATA

File system data chỉ các thông tin liên quan đến hệ thống tập tin và được lưu trữ trên ổ đĩa, nhưng không nằm trong Master File Table (MFT). Nó bao gồm các thông tin như thông tin về cấu trúc ổ đĩa, nhưng không giới hạn ở đó, mà còn bao gồm các thông tin về các cluster, vùng chưa được sử dụng, các trình điều khiển tập tin, thông tin về quyền truy cập và các thông tin quản lý khác.

Các thông tin này được lưu trữ trong các cấu trúc dữ liệu như boot sector, bảng MFT Mirror, bảng MFT thay thế và các bảng dữ liệu khác được lưu trữ ở nhiều vị trí khác nhau trên đĩa. Hệ thống tập tin của một hệ điều hành nhất định sẽ có cách tổ chức và lưu trữ các thông tin này khác nhau.

MASTER FILE TABLE COPY

Master File Table Copy (MFT Copy) là một bản sao của MFT được lưu trữ ở cuối vùng dữ liệu NTFS trên ổ đĩa. Mục đích của MFT Copy là để đảm bảo rằng nếu bất kỳ vấn đề gì xảy ra với MFT chính, hệ thống vẫn có thể sử dụng MFT Copy để truy cập vào các file và thư mục trên ổ đĩa.

MFT Copy cũng được chia thành các entry giống như MFT chính và có thể được đọc bởi hệ thống tập tin NTFS để khôi phục các file và thư mục bị hỏng. MFT Copy thường được cập nhật cùng với MFT chính khi có thay đổi về file hoặc thư mục trên ổ đĩa NTFS.

Đọc các thông tin được mô tả trong Partition Boot Sector

Partition Boot Sector NTFS là một khu vực đặc biệt trên ổ đĩa được sử dụng bởi hệ điều hành Windows để lưu trữ thông tin về phân vùng NTFS (New Technology File System). Các thông tin được mô tả trong Partition Boot Sector NTFS bao gồm:

- **Jump Instruction:** Đây là các lệnh nhảy được sử dụng để truy cập vào phân vùng.
- **OEM ID:** Đây là một chuỗi ký tự đại diện cho nhà sản xuất của phần mềm.
- **Bytes Per Sector:** Số byte trong một sector.
- **Sectors Per Cluster:** Số sector trong một cụm (cluster).
- **Reserved Sectors:** Số sector được dành riêng cho các mục đích hệ thống.
- **Media Descriptor:** Đây là một byte mô tả loại phương tiện lưu trữ (ví dụ: 0xF8 cho ổ đĩa cứng).
- **Sectors Per Track:** Số sector trên một track.
- **Number of Heads:** Số đầu đọc/ghi trên đĩa.
- **Hidden Sectors:** Số sector được ẩn khỏi hệ thống tệp.

- Total Sectors: Tổng số sector trên phân vùng.
- MFT Location: Vị trí bắt đầu của Master File Table (MFT).
- MFT Mirror Location: Vị trí bắt đầu của bản sao MFT.
- Cluster Per MFT Record: Số cụm được sử dụng cho một bản ghi MFT.
- Cluster Per Index Record: Số cụm được sử dụng cho một bản ghi chỉ mục.
- Volume Serial Number: Số định danh duy nhất của phân vùng.
- Checksum: Giá trị kiểm tra tính toàn vẹn của Partition Boot Sector NTFS.

Các thông tin này cung cấp cho hệ thống thông tin về cấu trúc của phân vùng NTFS và giúp hệ thống truy cập vào các tệp tin và thư mục được lưu trữ trên phân vùng.

3.4 Thiết kế chi tiết thuật toán

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

Hàm readSector sử dụng hàm CreateFile để mở tệp và truy cập đến ổ đĩa được chỉ định. Sau đó, nó sử dụng hàm SetFilePointer để di chuyển con trỏ tệp đến vị trí bắt đầu đọc sector trên đĩa. Cuối cùng, nó sử dụng hàm ReadFile để đọc dữ liệu sector từ ổ đĩa và lưu trữ vào vùng nhớ được chỉ định bởi con trỏ sector. Nếu quá trình đọc sector thành công, hàm trả về 1. Nếu xảy ra lỗi, hàm trả về 0 và in ra thông báo lỗi tương ứng.

3.4.1 FAT32

Đọc các thông tin chi tiết của một phân vùng

```
1 string getLittleEdianHex(BYTE sector[512], int offset, int byte)
```

Hàm `getLittleEdianHex` được sử dụng để chuyển đổi các giá trị byte trong sector từ định dạng little-endian sang định dạng hexa. Hàm này nhận vào 3 tham số: sector là một mảng 512 byte chứa sector cần xử lý, offset là vị trí byte đầu tiên của giá trị cần chuyển đổi trong sector, và byte là số lượng byte của giá trị cần chuyển đổi.

```
1 void getBootSectorInformation(BYTE sector[512])
```

Hàm `getBootSectorInformation` được sử dụng để đọc các thông tin quan trọng trong Boot Sector của phân vùng. Các thông tin đọc được sẽ được lưu trữ trong các biến thành viên của một đối tượng có kiểu struct được định nghĩa trước đó.

Các thông tin đọc được bao gồm:

- `bytesPerSector`: số byte trên một sector
- `sectorsPerCluster`: số sector trên một cluster
- `reservedSectors`: số sector được dành riêng cho Boot Sector và các khu vực khác
- `numFATs`: số lượng bảng FAT
- `totalSectors`: tổng số sector trên phân vùng
- `sectorsPerFAT`: số sector được dành riêng cho mỗi bảng FAT
- `clusterNumberRDET`: số cluster đầu tiên của RDET (Resident Data Entry Table)
- `sectorNumberFileSystem`: số sector đầu tiên của hệ thống tệp tin
- `sectorNumberBackupBoot`: số sector đầu tiên của bản sao lưu của Boot Sector
- `firstSectorOfRDET`: số sector đầu tiên của RDET được lưu trữ trên ổ đĩa

Các giá trị này được đọc từ các vị trí cụ thể trong Boot Sector bằng cách sử dụng hàm `getLittleEdianHex` để chuyển đổi giá trị byte sang định dạng hexa và sau đó sử dụng hàm `strtol` để chuyển đổi định dạng hexa sang kiểu

số nguyên. Sau khi đọc các thông tin cần thiết, các giá trị này được lưu trữ vào các biến thành viên trong đối tượng curr được định nghĩa trước đó.

Hàm strtol là một hàm trong thư viện tiêu chuẩn C, được sử dụng để chuyển đổi một chuỗi ký tự đại diện cho một số sang kiểu số nguyên dạng long.

Cú pháp của hàm như sau [9]:

```
1 long int strtol(const char* str, char** endptr, int base)
```

Trong đó:

- 'str' là con trỏ tới chuỗi cần chuyển đổi.
- 'endptr' là một con trỏ tới một chuỗi ký tự, được sử dụng để lưu giữ địa chỉ của ký tự đầu tiên không thuộc phạm vi số hợp lệ (nếu có).
- 'base' là hệ số cơ số của số được đại diện bởi chuỗi (ví dụ: hệ cơ số 16 cho số hexa, hệ cơ số 10 cho số thập phân, ...).

Hàm strtol sẽ trả về giá trị số nguyên tương ứng với chuỗi được chuyển đổi. Nếu không thể chuyển đổi được, hàm sẽ trả về giá trị 0.

Trong hàm getBootSectorInformation, hàm strtol được sử dụng để chuyển đổi chuỗi ký tự hexa sang giá trị số nguyên. Tham số thứ hai và thứ ba của hàm đều được đặt là NULL và 16 tương ứng, cho biết số được đại diện bởi chuỗi là số hexa (với hệ cơ số là 16).

Hiển thị thông tin cây thư mục của phân vùng

```
1 void readRDET(LPCWSTR drive, int SRDET)
```

Hàm readRDET được sử dụng để đọc bảng MFT (Master File Table) từ ổ đĩa. Bảng MFT chứa thông tin về các tệp trên ổ đĩa, bao gồm tên tệp, kích thước tệp, trạng thái và vị trí trên ổ đĩa.

Hàm nhận hai tham số đầu vào là đường dẫn đến ổ đĩa và vị trí bắt đầu của bảng MFT (SRDET). Trong khi đọc bảng MFT, hàm sẽ chạy vòng lặp vô hạn và đọc từng sector 512 byte của bảng MFT. Sau đó, hàm sử dụng các hàm phụ trợ để trích xuất thông tin về các tệp từ sector đó.

Trong vòng lặp, hàm sử dụng biến `k` để theo dõi vị trí của sector đang được đọc. Biến `temp` lưu trữ tên tệp, `count` đếm số sector đã đọc và `checkend` kiểm tra xem đã đọc hết bảng MFT hay chưa.

Trong mỗi sector, hàm sử dụng một vòng lặp để trích xuất thông tin về các tệp. Các trường thông tin được trích xuất bao gồm tên tệp, kích thước tệp, trạng thái và vị trí trên ổ đĩa.

Đối với các tệp là thư mục, hàm sẽ đệ quy để đọc các tệp con trong thư mục đó. Nếu tệp là tệp văn bản, hàm sẽ in ra nội dung của tệp đó. Nếu tệp là một loại tệp khác, hàm sẽ in ra thông báo về cách mở tệp đó.

Hàm sử dụng các hàm phụ trợ như `ReadSector`, `toString`, `toNumber`, `decToBinary`, `readatr`, `readFile` để trích xuất thông tin từ sector và in ra kết quả tương ứng.

Các hàm phụ trợ này có các chức năng sau:

- `ReadSector`: đọc dữ liệu từ sector trên ổ đĩa.
- `toString`: chuyển đổi các byte thành chuỗi ký tự.
- `toNumber`: chuyển đổi các byte thành giá trị số.
- `decToBinary`: chuyển đổi giá trị

```
1 void readSDET(LPCWSTR drive, int SRDET)
```

Hàm `readSDET` được sử dụng để đọc và truy xuất thông tin về các tệp tin và thư mục trong một phân vùng FAT32. Đầu vào của hàm bao gồm `drive`, đây là tên ổ đĩa cần đọc (VD: `L"C:`) và `SRDET`, đây là sector đầu tiên của SDET (Short Directory Entry Table). SDET chứa các bản ghi về các tệp tin và thư mục trong phân vùng FAT32, tương tự như bảng FAT.

Các biến `k`, `temp` và `count` được sử dụng trong quá trình đọc SDET. Biến `k` được sử dụng để lưu vị trí sector hiện tại trong phân vùng FAT32. Biến `temp` là một chuỗi được sử dụng để lưu tên tệp tin hoặc thư mục, biến `count` được sử dụng để đếm số lượng các tệp tin và thư mục đã đọc.

Trong vòng lặp `while (true)`, một biến kiểm tra `checkend` được sử dụng để xác định xem đã đọc hết SDET hay chưa. Mỗi lần lặp, hàm `ReadSector` được gọi để đọc một sector từ phân vùng FAT32 tại vị trí `SRDET + k`.

Sau đó, vòng lặp for được sử dụng để duyệt qua các bản ghi trong sector đã đọc.

Trong vòng lặp for, các câu lệnh điều kiện được sử dụng để kiểm tra các bản ghi đã đọc. Nếu bản ghi là một bản ghi đã bị xóa (0xe5), hoặc là bản ghi trống (0x00), thì vòng lặp for tiếp tục duyệt qua các bản ghi khác. Nếu bản ghi là một bản ghi đại diện cho một thư mục (0x0f), thì tên thư mục được đọc và lưu trữ trong biến temp, và biến count được tăng lên một. Nếu bản ghi là một bản ghi đại diện cho một tập tin, thì thông tin về tập tin được in ra màn hình. Thông tin về một tập tin bao gồm tên tập tin, kích thước, trạng thái (có phải là tập tin ẩn, hệ thống, chỉ đọc, có nhân ổ đĩa, hoặc là một thư mục), vị trí cluster bắt đầu và các sector được chiếm bởi tập tin đó. Nếu tệp được đọc là một thư mục, hàm đệ quy sẽ được gọi lại để đọc các tệp con bên trong thư mục đó. Nó được sử dụng để truy xuất và hiển thị nội dung của toàn bộ cấu trúc thư mục và tệp tin trong hệ thống tập tin.

3.4.2 NTFS

```
1 void Read_BPB(BYTE* sector, LPCWSTR disk)
```

Hàm Read_BPB đọc thông tin trong bảng phân vùng NTFS. Trong hàm, nó sử dụng hàm Get_Bytes để đọc các giá trị cần thiết từ sector được truyền vào, như số byte mỗi sector, số sector mỗi cluster, số sector trên mỗi track, tổng số sector và các vị trí bắt đầu của MFT và MFTMirror. Sau đó, in các giá trị này ra màn hình để hiển thị thông tin bảng phân vùng NTFS cho người dùng.

```
1 void read_MFT(unsigned int MFTStart, unsigned int sectors_per_cluster,
    LPCWSTR disk)
```

Hàm read_MFT có chức năng đọc thông tin trong MFT (Master File Table) của ổ đĩa. Đầu vào của hàm bao gồm vị trí bắt đầu của MFT, số lượng sector trên mỗi cụm (clusters), và tên của đĩa. Hàm khởi tạo một bộ đệm 512 byte để lưu trữ MFT và nhân MFTStart với sector_per_cluster để chuyển đổi số sector thành byte offset. Sau đó, nó sử dụng hàm trợ giúp ReadSect để đọc MFT từ đĩa vào bộ đệm.

Hàm sau đó trích xuất thông tin từ MFT, bao gồm:

- Số sector mà đầu vào \$INFORMATION bắt đầu.
- Độ dài của đầu vào \$INFORMATION.
- Số sector mà đầu vào \$FILENAME bắt đầu.
- Độ dài của đầu vào \$FILENAME.
- Số sector mà đầu vào \$DATA bắt đầu.
- Độ dài của đầu vào \$DATA.
- Số sector được chiếm bởi MFT.

```
1 void ReadSect(LPCWSTR disk, BYTE*& DATA, unsigned int _nsect)
```

Hàm ReadSect được sử dụng để đọc một sector từ ổ đĩa cứng hoặc ổ đĩa mềm. Hàm nhận vào ba đối số: con trỏ đến một chuỗi ký tự wide-character (LPCWSTR) đại diện cho đường dẫn tới thiết bị lưu trữ mà sector cần được đọc, con trỏ đến một mảng BYTE, sẽ được sử dụng để lưu trữ nội dung của sector được đọc và một số nguyên dương đại diện cho số sector cần đọc trên thiết bị lưu trữ. Hàm CreateFile() được sử dụng để mở ổ đĩa mềm, với các tham số được chỉ định cho quyền truy cập, chế độ chia sẻ, và tùy chọn bổ sung. Hàm SetFilePointerEx() được sử dụng để đặt con trỏ đọc tại sector được chỉ định. Sau đó, hàm ReadFile() được sử dụng để đọc dữ liệu từ sector được chỉ định và lưu trữ vào biến DATA. Cuối cùng, hàm CloseHandle() được sử dụng để đóng tệp tin ổ đĩa mềm được mở.

```
1 void TreeFolder(unsigned int len_MFT, unsigned int MFTStart, LPCWSTR disk)
```

Hàm TreeFolder có chức năng đọc và hiển thị cấu trúc thư mục dựa trên các thông tin được trích xuất từ Master File Table (MFT) của một ổ đĩa. Hàm sử dụng hàm ReadSect để đọc dữ liệu từ ổ đĩa và các hàm khác như LittleEndianConvert, Get_Bytes, Read_Info_Entry, Read_Filename_Entry, Read_Data_Entry, Print_TreeFolder để trích xuất các thông tin cần thiết từ MFT. Hàm sẽ đọc dữ liệu từ ổ đĩa vào biến currentEntry bằng cách gọi hàm ReadSect, kiểm tra xem sector đó có phải là một file hay không và lấy ra ID của file đó. Nếu ID của file lớn hơn 38, tức là file đó là một thư mục, hàm sẽ trích xuất các thông tin liên quan đến thư mục đó từ currentEntry,

bao gồm thông tin về các file con bên trong thư mục đó, và lưu trữ ID của thư mục đó vào vector fileID.

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

Hàm Get_Bytes là một hàm đọc dữ liệu và trả về một số nguyên (kiểu dữ liệu int64_t) được lưu trữ trong một mảng các byte.

```
1 string LittleEndianConvert(BYTE* DATA, int offset, int number)
```

Hàm LittleEndianConvert là một hàm đọc và chuyển đổi một chuỗi ký tự được lưu trữ dưới dạng Little-Endian (thứ tự byte thấp nhất đến byte cao nhất) trong một mảng các byte. Kết quả hàm trả về giá trị của chuỗi str, là chuỗi ký tự đã được chuyển đổi từ dạng Little-Endian sang dạng chuỗi ký tự thông thường.

```
1 void Read_Data_Entry(BYTE* Entry, int start)
```

Hàm này đọc thông tin về thuộc tính \$DATA của file đó, bao gồm:

- Kích thước của thuộc tính (bao gồm cả header).
- Kích thước của file.
- Loại của thuộc tính, có phải là resident hay không.
- Nếu thuộc tính là resident, hàm sẽ đọc thông tin về kích thước và vị trí của phần nội dung của file trong thuộc tính, sau đó sử dụng hàm LittleEndianConvert để chuyển đổi các byte dữ liệu đó thành chuỗi ký tự và xuất ra màn hình.

```
1 int Read_Info_Entry(BYTE* Entry, int start)
```

Hàm Read_Info_Entry có nhiệm vụ đọc và hiển thị thông tin về thuộc tính STANDARD_INFORMATION của một file trong hệ thống tập tin NTFS. Trong hàm, đầu tiên nó sẽ đọc và hiển thị độ dài của thuộc tính và trạng thái của file dưới dạng chuỗi nhị phân. Sau đó, hàm sử dụng một vòng lặp để duyệt qua các bit của trạng thái và hiển thị các thuộc tính tương ứng với các bit bật.

Nếu bit đầu tiên của trạng thái là 1, thì file đó là một file ẩn. Nếu bit thứ hai của trạng thái là 1, thì file đó là một hệ thống tập tin. Trong trường hợp này, hàm sẽ trả về giá trị -1 để cho biết file đó không phải là một file thông thường.

Nếu không phải là file ẩn hoặc hệ thống tập tin, hàm sẽ hiển thị thông tin về các thuộc tính của file như chỉ đọc, thư mục, đánh dấu lần đầu và tập tin nén. Cuối cùng, hàm trả về độ dài của thuộc tính để sử dụng cho việc đọc các thuộc tính tiếp theo của file.

```
1 int Read_Filename_Entry(BYTE* Entry, int start, int ID)
```

Hàm `Read_Filename_Entry` thực hiện đọc và phân tích các thông tin liên quan đến tên và định dạng của một tập tin được lưu trữ trong hệ thống tập tin NTFS.

Các thông tin cụ thể được đọc bao gồm:

- Chiều dài của thuộc tính `$FILE_NAME`
- Mã số của thư mục cha
- Độ dài của tên file
- Tên của tập tin
- Định dạng của tập tin (dựa trên phần mở rộng)

Sau khi đọc được các thông tin này, hàm sẽ in ra các thông tin này để người dùng có thể hiểu được nội dung của tập tin cụ thể đang được phân tích. Hàm cũng thực hiện một số kiểm tra định dạng của tập tin và in ra thông báo hướng dẫn người dùng cần sử dụng phần mềm nào để mở tập tin đó. Cuối cùng, hàm trả về kích thước của thuộc tính `$FILE_NAME` để được sử dụng bởi các hàm khác.

```
1 void printSector(BYTE* sector)
```

Hàm này có chức năng in ra các byte của một sector trong ổ đĩa. Mỗi sector có độ dài 512 byte và được lưu trữ liên tục trong ổ đĩa. Hàm sẽ in ra giá trị hex của từng byte, bắt đầu từ byte đầu tiên đến byte cuối cùng trong sector.

```
1 void Print_TreeFolder(int a, int slash, int pos)
```

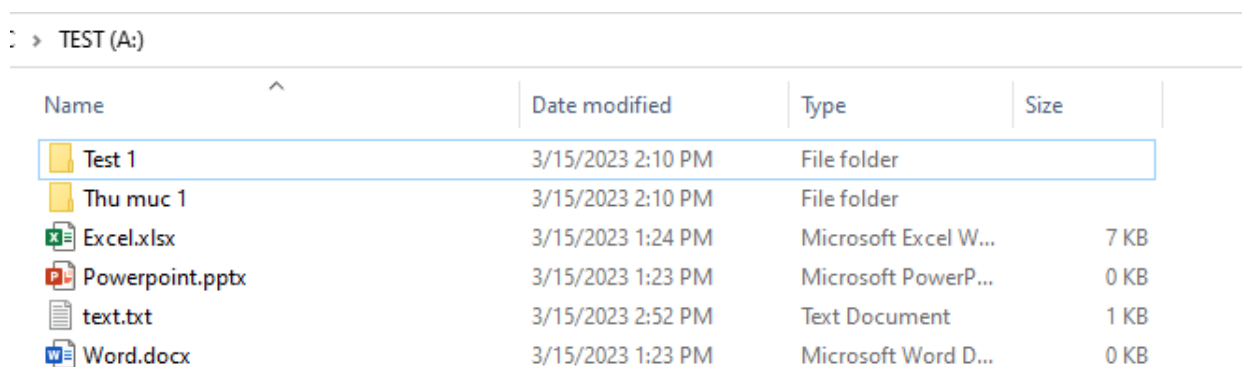
Hàm `Print_TreeFolder` được sử dụng để in ra cây thư mục bắt đầu từ một thư mục được chỉ định. Đầu vào của hàm là ba tham số: `a` - ID của thư mục được in ra, `slash` - số lượng dấu gạch chéo để in ra định dạng cây thư mục, và `pos` - vị trí của thư mục trong vector các ID thư mục và tên tập tin.

Chương 4

Giao diện chương trình

Lưu ý: phải chạy chương trình với quyền admin (Run as Administrator)

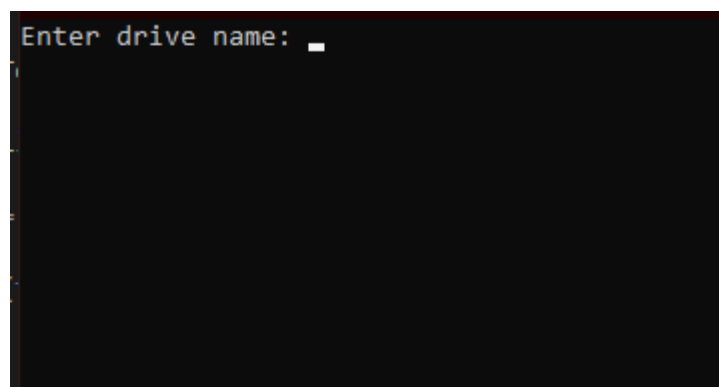
Ổ đĩa được truy xuất



TEST (A:)				
Name		Date modified	Type	Size
Test 1		3/15/2023 2:10 PM	File folder	
Thu mục 1		3/15/2023 2:10 PM	File folder	
Excel.xlsx		3/15/2023 1:24 PM	Microsoft Excel W...	7 KB
Powerpoint.pptx		3/15/2023 1:23 PM	Microsoft PowerP...	0 KB
text.txt		3/15/2023 2:52 PM	Text Document	1 KB
Word.docx		3/15/2023 1:23 PM	Microsoft Word D...	0 KB

Hình 4.1: Các tập tin của ổ đĩa

Nhập tên ổ đĩa cần đọc



Hình 4.2: Màn hình input

4.0.1 Đối với FAT32

```
Enter drive name: A
FAT32 PARTITION Boot Sector:
Offset  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  .X.MSDOS5.0.....
00000000 eb 58 90 4d 53 44 4f 53 35 2e 30 00 02 10 f6 11 .....?.....7s
00000010 02 00 00 00 00 f8 00 00 3f 00 ff 00 00 d8 37 73 ..8...'.....
00000020 00 80 38 01 05 27 00 00 00 00 00 00 02 00 00 00 .....
00000030 01 00 06 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000040 80 00 29 9c 39 b0 da 4e 4f 20 4e 41 4d 45 20 20 ..).9..NO NAME
00000050 20 20 46 41 54 33 32 20 20 20 33 c9 8e d1 bc f4 FAT32 3.....
00000060 7b 8e c1 8e d9 bd 00 7c 88 56 40 88 4e 02 8a 56 {.....|.V@.N..V
00000070 40 b4 41 bb aa 55 cd 13 72 10 81 fb 55 aa 75 0a @.A..U..r...U.u.
00000080 f6 c1 01 74 05 fe 46 02 eb 2d 8a 56 40 b4 08 cd ...t..F..-.V@...
00000090 13 73 05 b9 ff ff 8a f1 66 0f b6 c6 40 66 0f b6 .s.....f...@f..
000000a0 d1 80 e2 3f f7 e2 86 cd c0 ed 06 41 66 0f b7 c9 ...?.....Af...
000000b0 66 f7 e1 66 89 46 f8 83 7e 16 00 75 39 83 7e 2a f..f.F...~..u9.~*
000000c0 00 77 33 66 8b 46 1c 66 83 c0 0c bb 00 80 b9 01 .w3f.F.f.....
000000d0 00 e8 2c 00 e9 a8 03 a1 f8 7d 80 c4 7c 8b f0 ac ..,.....}.|...
000000e0 84 c0 74 17 3c ff 74 09 b4 0e bb 07 00 cd 10 eb ..t.<.t.....
000000f0 ee a1 fa 7d eb e4 a1 7d 80 eb df 98 cd 16 cd 19 ...}....}.....
00001000 66 60 80 7e 02 00 0f 84 20 00 66 6a 00 66 50 06 f`~.... .fj.fP.
00001100 53 66 68 10 00 01 00 b4 42 8a 56 40 8b f4 cd 13 Sfh.....B.V@....
00001200 66 58 66 58 66 58 66 58 eb 33 66 3b 46 f8 72 03 fXfXfXfX.3f;F.r.
00001300 f9 eb 2a 66 33 d2 66 0f b7 4e 18 66 f7 f1 fe c2 ..*f3.f..N.f....
00001400 8a ca 66 8b d0 66 c1 ea 10 f7 76 1a 86 d6 8a 56 ..f..f....v....V
00001500 40 8a e8 c0 e4 06 0a cc b8 01 02 cd 13 66 61 0f @.....fa.
00001600 82 74 ff 81 c3 00 02 66 40 49 75 94 c3 42 4f 4f .t.....f@Iu..BOO
00001700 54 4d 47 52 20 20 20 20 00 00 00 00 00 00 00 00 TMGR .....
00001800 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00001900 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00001a00 00 00 00 00 00 00 00 00 00 00 00 00 0d 0a 44 69 .....Di
00001b00 73 6b 20 65 72 72 6f 72 ff 0d 0a 50 72 65 73 73 sk error...Press
00001c00 20 61 6e 79 20 6b 65 79 20 74 6f 20 72 65 73 74 any key to rest
00001d00 61 72 74 0d 0a 00 00 00 00 00 00 00 00 00 00 00 art.....
```

Hình 4.3: Volume Boot Sector

Sau đó, các thông tin trong Boot Sector sẽ được in ra

```

00001f0  00 00 00 00 00 00 00 00 ac 01 b9 01 00 00 55 aa  ....U.
FAT name: FAT32
Bytes Per Sector: 512
Sectors Per Cluster: 16
Reserved Sectors: 4598
Number of file allocation tables: 2
Number of Sectors in Partition: 20480000
Number of Sectors Per FAT: 9989
Cluster Number of RDET: 2
Sector Number of the File System Information Sector: 2
Sector Number of the Backup Boot Sector: 6
Number of sectors in RDET: 0 sector
Lenght volume: SV = 20480000 sector = 10000MB
First sector of FAT1: 4598
Starting sector of RDET: 24576
Starting sector of DATA: 24576

```

Hình 4.4: Thông tin trong Boot Sector

Và đồng thời hiện thông tin các tập tin cùng cây thư mục

```

TREE FOLDER

File name: TEST
Lenght: 0
Status: vollabel
Cluster bat dau: 0

File name: System Volume Information
Lenght: 0
Status: hidden system directory
Cluster bat dau: 3
Chiem cac sector: 4294942736 4294942737 4294942738 4294942739 4294942740 4294942741 4294942742 4294942743 4294942744 429
4942745 4294942746 4294942747 4294942748 4294942749 4294942750 4294942751

File name: Word.docx
Lenght: 0
Status: archive
Cluster bat dau: 0
=> Use Microsoft Office Word to open!

File name: Powerpoint.pptx
Lenght: 0
Status: archive
Cluster bat dau: 0
=> Use Microsoft Office PowerPoint to open!

File name: Excel.xlsx
Lenght: 6184
Status: archive
Cluster bat dau: 7
Chiem cac sector: 4294942800 4294942801 4294942802 4294942803 4294942804 4294942805 4294942806 4294942807 4294942808 429

```

Hình 4.5: Thông tin cây thư mục của phân vùng

4.0.2 Đối với NTFS

```
Microsoft Visual Studio Debug Console
Enter drive name: a
Read successfully !

offset  0  1  2  3  4  5  6  7      8  9  A  B  C  D  E  F
0x000  EB 52 90 4E 54 46 53 20    20 20 20 00 02 08 00 00
0x010  00 00 00 00 00 F8 00 00    3F 00 FF 00 00 D8 37 73
0x020  00 00 00 00 80 00 80 00    FF 7F 38 01 00 00 00 00
0x030  00 00 0C 00 00 00 00 00    02 00 00 00 00 00 00 00
0x040  F6 00 00 00 01 00 00 00    C1 34 1F 88 47 1F 88 46
0x050  00 00 00 00 FA 33 C0 8E    D0 BC 00 7C FB 68 C0 07
0x060  1F 1E 68 66 00 CB 88 16    0E 00 66 81 3E 03 00 4E
0x070  54 46 53 75 15 B4 41 BB    AA 55 CD 13 72 0C 81 FB
0x080  55 AA 75 06 F7 C1 01 00    75 03 E9 DD 00 1E 83 EC
0x090  18 68 1A 00 B4 48 8A 16    0E 00 8B F4 16 1F CD 13
0x0A0  9F 83 C4 18 9E 58 1F 72    E1 3B 06 0B 00 75 DB A3
0x0B0  0F 00 C1 2E 0F 00 04 1E    5A 33 DB B9 00 20 2B C8
0x0C0  66 FF 06 11 00 03 16 0F    00 8E C2 FF 06 16 00 E8
0x0D0  4B 00 2B C8 77 EF B8 00    BB CD 1A 66 23 C0 75 2D
0x0E0  66 81 FB 54 43 50 41 75    24 81 F9 02 01 72 1E 16
0x0F0  68 07 BB 16 68 52 11 16    68 09 00 66 53 66 53 66
0x100  55 16 16 16 68 B8 01 66    61 0E 07 CD 1A 33 C0 BF
0x110  0A 13 B9 F6 0C FC F3 AA    E9 FE 01 90 90 66 60 1E
0x120  06 66 A1 11 00 66 03 06    1C 00 1E 66 68 00 00 00
0x130  00 66 50 06 53 68 01 00    68 10 00 B4 42 8A 16 0E
0x140  00 16 1F 8B F4 CD 13 66    59 5B 5A 66 59 66 59 1F
0x150  0F 82 16 00 66 FF 06 11    00 03 16 0F 00 8E C2 FF
0x160  0E 16 00 75 BC 07 1F 66    61 C3 A1 F6 01 E8 09 00
0x170  A1 FA 01 E8 03 00 F4 EB    FD 8B F0 AC 3C 00 74 09
0x180  B4 0E BB 07 00 CD 10 EB    F2 C3 0D 0A 41 20 64 69
0x190  73 6B 20 72 65 61 64 20    65 72 72 6F 72 20 6F 63
0x1A0  63 75 72 72 65 64 00 0D    0A 42 4F 4F 54 4D 47 52
0x1B0  20 69 73 20 63 6F 6D 70    72 65 73 73 65 64 00 0D
0x1C0  0A 50 72 65 73 73 20 43    74 72 6C 2B 41 6C 74 2B
0x1D0  44 65 6C 20 74 6F 20 72    65 73 74 61 72 74 0D 0A
0x1E0  00 00 00 00 00 00 00 00    00 00 00 00 00 00 00 00
0x1F0  00 00 00 00 00 00 8A 01    A7 01 BF 01 00 00 55 AA
```

Hình 4.6: Partition Boot Sector

```
|Bytes Per Sector : 512
|Sectors Per Cluster : 8
|Sectors Per Track : 63
|Total Sectors : 20479999
|Cluster start of MFT : 786432
|Cluster start of MFTMirror : 2

$INFORMATION Entry starts at: 56
Length of INFORMATION Entry: 72
$FILENAME Entry starts at: 128
Length of $FILENAME Entry: 104
DATA Entry starts at: 232
Length of DATA Entry: 393216
Number of sectors in MFT is: 1032
```

Hình 4.7: Thông tin chi tiết của phân vùng

```
                                TREE FOLDER
Test 1
    Test 2
        Nen.rar

Thu muc 1
    Thu muc 2
        Thu muc 3
            HCMUS.txt

Excel.xlsx
Powerpoint.pptx
text.txt
Word.docx
```

Hình 4.8: Cây thư mục gốc gồm tên tập tin / thư mục

```

File ID: 39
STANDARD_INFORMATION
- Length of attribute (header included): 96
- Status Attribute of File:

Attribute $FILE_NAME
- Length of attribute (header included): 104
- Parent file: 5
- Length of name file: 6
- Name of file: Test 1


File ID: 40
STANDARD_INFORMATION
- Length of attribute (header included): 96
- Status Attribute of File:

Attribute $FILE_NAME
- Length of attribute (header included): 104
- Parent file: 39
- Length of name file: 6
- Name of file: Test 2


Attribute $DATA
- Length of attribute (header included): 176
- Size of file: 144
    => Non-resident


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

Attribute $FILE_NAME
- Length of attribute (header included): 104
- Parent file: 40
- Length of name file: 7
- Name of file: Nen.rar


Attribute $DATA
- Length of attribute (header included): 48
- Size of file: 24
    => Non-resident


File ID: 42
STANDARD_INFORMATION
- Length of attribute (header included): 96
- Status Attribute of File:

Attribute $FILE_NAME
- Length of attribute (header included): 112
- Parent file: 5
- Length of name file: 9
- Name of file: Thu muc 1

```

Hình 4.9: Thông tin trên cây thư mục (1)


```

File ID: 43
STANDARD_INFORMATION
- Length of attribute (header included): 96
- Status Attribute of File:

Attribute $FILE_NAME
- Length of attribute (header included): 112
- Parent file: 42
- Length of name file: 9
- Name of file: Thu muc 2


File ID: 44
STANDARD_INFORMATION
- Length of attribute (header included): 96
- Status Attribute of File:

Attribute $FILE_NAME
- Length of attribute (header included): 112
- Parent file: 43
- Length of name file: 9
- Name of file: Thu muc 3


Attribute $DATA
- Length of attribute (header included): 184
- Size of file: 152
  => Non-resident


File ID: 45
STANDARD_INFORMATION
- Length of attribute (header included): 96
- Status Attribute of File: 100000
  => Archive

Attribute $FILE_NAME
- Length of attribute (header included): 112
- Parent file: 44
- Length of name file: 9
- Name of file: HCMUS.txt


Attribute $DATA
- Length of attribute (header included): 48
- Size of file: 20
  => Resident

Content: Day la noi dung file

File ID: 46
STANDARD_INFORMATION
- Length of attribute (header included): 96
- Status Attribute of File: 100000
  => Archive

Attribute $FILE_NAME
- Length of attribute (header included): 112
- Parent file: 5
- Length of name file: 10
- Name of file: Excel.xlsx

```

Hình 4.10: Thông tin trên cây thư mục (2)

```

- Parent file: 5
- Length of name file: 10
- Name of file: Excel.xlsx
  => Use Microsoft Excel to open this file

File ID: 47
STANDARD_INFORMATION
- Length of attribute (header included): 96
- Status Attribute of File: 100000
  => Archive

Attribute $FILE_NAME
- Length of attribute (header included): 120
- Parent file: 5
- Length of name file: 15
- Name of file: Powerpoint.pptx
  => Use Microsoft PowerPoint to open this file

File ID: 48
STANDARD_INFORMATION
- Length of attribute (header included): 96
- Status Attribute of File: 100000
  => Archive

Attribute $FILE_NAME
- Length of attribute (header included): 112
- Parent file: 5
- Length of name file: 8
- Name of file: text.txt

Attribute $DATA
- Length of attribute (header included): 48
- Size of file: 22
  => Resident

Content: Doc noi dung cua NTFSS

File ID: 49
STANDARD_INFORMATION
- Length of attribute (header included): 96
- Status Attribute of File: 100000
  => Archive

Attribute $FILE_NAME
- Length of attribute (header included): 112
- Parent file: 5
- Length of name file: 9
- Name of file: Word.docx
  => Use Microsoft Word to open this fille

```

Hình 4.11: Thông tin trên cây thư mục (3)

Tài liệu tham khảo

- [1] *File system*. Feb. 2023. URL: https://en.wikipedia.org/wiki/File_system.
- [2] *File system formats available in Disk Utility on mac*. URL: <https://support.apple.com/en-vn/guide/disk-utility/dsku19ed921c/mac>.
- [3] *Apple File System*. Jan. 2023. URL: https://en.wikipedia.org/wiki/Apple_File_System.
- [4] Kaplarevic, Vladimir. *Introduction to the linux file system EXT, MINIX, and alternatives*. Nov. 2022. URL: <https://phoenixnap.com/kb/linux-file-system>.
- [5] *What is ext4 (ext2, ext3) – linux file system*. URL: <https://recoverhdd.com/blog/the-ext-ext2-ext3-ext4-filesystem.html>.
- [6] Linda. *XFS vs ext4: Which one is better?* Dec. 2022. URL: <https://www.partitionwizard.com/partitionmanager/xfs-vs-ext4.html>.
- [7] Peng, Zewu et al. “A data recovery method for NTFS files system”. In: *Applications and Techniques in Information Security: 6th International Conference, ATIS 2015, Beijing, China, November 4-6, 2015, Proceedings*. Springer. 2015, pp. 379–386.
- [8] Minatu. *Khái quát Về Fat*. Nov. 2016. URL: <https://lazytrick.wordpress.com/2015/12/27/khai-quat-ve-fat/>.
- [9] *Strtol*. URL: <https://cplusplus.com/reference/cstdlib/strtol/>.