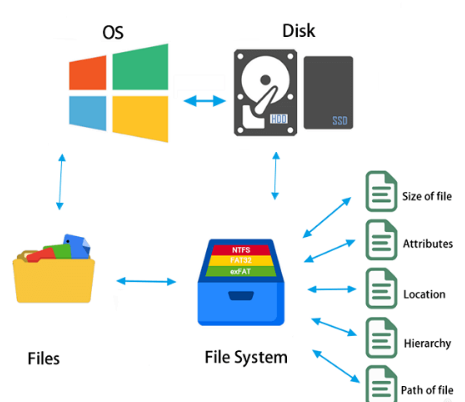
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

---o0o---

**A blue and white logo

Description automatically generated HỆ ĐIỀU HÀNH – CSC10007**

**QUẢN LÝ HỆ THỐNG TẬP TRÊN WINDOWS**

Ôn Gia Bảo (22127026)

Trần Anh Minh (22127275)

Đoàn Đặng Phương Nam (22127280)

Bùi Nguyễn Lan Vy (22127465)

Diệp Gia Huy (22127475)

**GROUP 11 – 22CLC02**

**MỤC LỤC**

# Bảng phân công công việc

|  |  |  |  |
| --- | --- | --- | --- |
| ***MSSV*** | ***Họ và tên*** | ***Công việc*** | ***Tỷ lệ hoàn thành*** |
| 22127026 | Ôn Gia Bảo |  | 100% |
| 22127275 | Trần Anh Minh |  | 100% |
| 22127280 | Đoàn Đặng Phương Nam |  | 100% |
| 22127465 | Bùi Nguyễn Lan Vy |  | 100% |
| 22127475 | Diệp Gia Huy |  | 100% |

# Đánh giá mức độ hoàn thành

### FAT32

|  |  |
| --- | --- |
| Nội dung | Đánh giá |
| Đọc thông tin phân vùng FAT32 | 100% |
| Hiển thị được cây thư mục gốc và các cây thư mục con | 100% |
| Hiển thị nội dung tập tin đối với tập tin có phần mở rộng là txt | 100% |

### NTFS

|  |  |
| --- | --- |
| Nội dung | Đánh giá |
| Đọc thông tin phân vùng NTFS | 100% |
| Hiển thị được cây thư mục gốc và các cây thư mục con | 100% |
| Hiển thị nội dung tập tin đối với tập tin có phần mở rộng là txt | 100% |

# Cách đọc ổ đĩa định dạng FAT32

## Cấu trúc FAT 32

A diagram of a cluster

Description automatically generated

Gồm 2 phần:

* Vùng hệ thống:
  + Boot Sector: 512-byte đầu tiên.
  + Vùng FAT
* Vùng Data
  + RDET – Root Directory Entry Table
  + Dữ liệu tập tin/thư mục

## Bước 1: Đọc phân vùng

* Lấy tên ổ đĩa và cho chương trình đọc file hệ thống của ổ đĩa đó.
* Nếu không tồn tại thì thoát.

## Bước 2: Đọc Boot Sector

* Đọc 512-byte đầu tiên tương ứng với Boot Sector
* Và lưu cái thông số chính cần quan tâm:

|  |  |  |
| --- | --- | --- |
| **Offset (Hex)** | **Số byte** | **Ý nghĩa** |
| 0x03 | 8 | OME\_ID : Nơi sản xuất-version (MSDOS5.0) |
| 0x0B | 2 | Số byte trên Sector |
| 0x0D | 1 | SC: số secter trên cluster |
| 0x0E | 2 | SB: số sector thuộc vùng Bootsector |
| 0x10 | 1 | NF: số bảng FAT |
| 0x20 | 4 | Sv: kích thước volume |
| 0x24 | 4 | SF: kích thước mỗi bảng FAT |
| 0x2C | 4 | Cluster bắt đầu của RDET |
| 0x52 | 8 | Loại FAT (chuỗi “FAT32”) |

### Các thông số tính toán được:

* + Sector bắt đầu của bảng FAT = SB
  + Sector bắt đầu của RDET: SRDET = SB + SF \* NF
  + Sector bắt đầu vùng Data = SRDET + SRDET \* kích thước RDET (= 0 trong FAT32)
  + i = SB + SF \* NF +SRDET + (k – 2) \* SC
    - i: sector thứ i trên phân vùng
    - k: cluster thứ k trên vùng Data

## Bước 3: Đọc bảng FAT

Cấu trúc bảng FAT trong FAT 32:

* Thứ tự lưu trữ: Little endian
* Kích thước mỗi phần tử: 4 bytes

A table with text and numbers

Description automatically generated

Thuật toán đọc bảng FAT

* **Bước 1**: Nhảy đến vị trí bắt đầu của bảng FAT theo [công thức (Các thông số tính toán được - 1.)](#_Các_thông_số).
* **Bước 2**: Tạo một mảng để lưu kết quả.
* **Bước 3**: Đọc 4 bytes tiếp theo và chuyển thành giá trị (số nguyên).
* **Bước 4**: Nếu giá trị nằm thuộc vùng **IN USE** trong bảng trạng thái thì lưu vào mảng kết quả và đi lại bước 3.
* **Bước 5**: Kết thúc thuật toán và lưu mảng kết quả vào hệ thống.

### Truy xuất chuỗi cluster:

* + Lấy cluster bắt đầu i tra vào bảng FAT, gán clus = I và tạo mảng lưu các cluster đã thăm.
  + Nếu clus ≠ EOF, thêm clus vào mảng và cập nhật clus = FAT[clus].
  + Khi clus = EOF, mảng là chuỗi cluster.

## Bước 4: RDET – Root Directory Entry Table

### **Cấu trúc RDET trong FAT 32**

* Gồm các Entry kích thước 32 bytes chứa thông tin về tập tin/ thư mục
* Có 2 loại Entry:
  + Entry chính: chứa metadata của tập tin
  + Entry phụ: chứa tên của tập tin
* Các Entry phụ đánh số từ 1 và nằm ngay phía trên Entry chính mà nó lưu tên

|  |
| --- |
| Entry phụ N |
| … |
| Entry phụ 2 |
| Entry phụ 1 |
| Entry chính |

### **Entry chính**

Cấu trúc Entry chính – Các vùng quan trọng

|  |  |  |
| --- | --- | --- |
| **Offset (Hex)** | **Số byte** | **Ý nghĩa** |
| 0x00 | 8 | Tên – mã ASCII |
| 0x08 | 3 | Tên mở rộng – mã ASCII |
| 0x0B | 1 | Thuộc tính |
| 0x14 | 2 | Cluster bắt đầu – Word cao |
| 0x1A | 2 | Cluster bắt đầu – Word thấp |
| 0x1C | 4 | Kích thước |

Byte tại vị trí 0x00: Trạng thái của Entry

* 0xE5: Đã xóa
* 0x00: Trống
* 0x05: Thay thế tên có kí tự bắt đầu là 0xE5

Tên file gồm 11-bytes gồm: 8-bytes tên chính + 3-byte phần mở rộng (8.3 format name) gọi là SFN (Short File Name)

* Tất cả kí tự của SFN được lưu ở dạng in hoa
* Dấu “.” chia cắt 2 phần bị xóa
* Các kí tự dấu cách (0x20h) được thêm vào nếu SFN không đủ format 8.3
* Các kí tự được phép sử dụng:

0～9 A～Z ! # $ % & ' ( ) - @ ^ \_ ` { } ~

* Các kí tự không được phép:
  + \ / : \* ? " < > |
  + Tên chỉ có phần mở rộng (“.ext”, “.txt”,…)
* Các tên vượt quá format 8.3 sẽ bị rút ngắn:
  + Tên chính: rút thành tối đa 6 kí tự + ~N
  + Phần mở rộng: rút ngắn thành 3 kí tự đầu
* Ví dụ về tên

|  |  |  |
| --- | --- | --- |
| **Tên thư mục** | **Tên được lưu** | **Lý do** |
| Hello.txt | HELLO TXT | Lắp đầy format 8.3 bằng dấu cách |
| LongFileName.docx | LONGFI~1DOC | Tên vi phạm 8.3 format |
| VeryLongText.txt | VERYLO~1TXT | Tên vi phạm 8.3 format |
| VeryLongFolder.txt | VERYLO~2TXT | Tên rút ngắn bị trùng |
| .ext | EXT~1 | Tên vi phạm 8.3 format |
| a+b=c | A\_B\_C~1 | Tên vi phạm 8.3 format |

### Thuộc tính (0.0.A.D.V.S.H.R)

* Tại offset 0x0B đọc 1-byte và tra bảng trạng thái sau:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **0** | **Archive** | **Directory** | **Volume label** | **System** | **Hidden** | **Read-only** |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

* Nếu bit nào bật 1 thì Entry có trạng thái đó.
* Đặc biệt: 0x0F dấu hiệu của entry phụ.

Cluster bắt đầu.

* Gồm 4-byte ghép từ 2-byte Word cao và 2-byte Word thấp
* Ở mỗi phần đọc 2-byte và sắp sếp theo kiểu little endian
* Đặt 2-byte Word cao trước và 2-byte Word thấp sau -> 4-byte thể hiện cluster bắt đầu
* Để truy xuất chuỗi cluster chứa thông tin thư mục: đưa cluster bắt đầu vào công thức ([Truy xuất chuỗi cluster](#_Truy_xuất_chuỗi)).

### **Entry phụ - VFAT (Virtual FAT)**

* Chứa tên phần Entry chính không hỗ trợ.
* Lưu dưới dạng UTF-16
* Nhận dạng: tại offset 0x0Bh: 0x0Fh.

Cấu trúc Entry phụ

|  |  |  |
| --- | --- | --- |
| **Offset (Hex)** | **Số byte** | **Ý nghĩa** |
| 0x00 | 1 | Thứ tự entry (bắt đầu từ 1) |
| 0x01 | 10 | 5 ký tự UniCode – UTF16 |
| 0x0B | 1 | Dấu hiện nhận biết (0x0F) |
| 0x0E | 12 | 6 ký tự tiếp theo |
| 0x1C | 4 | 2 ký tự tiếp theo |

* Tối đa 255 kí tự
* Mỗi Entry phụ lưu được 13 ký tự unicode nếu hơn sẽ tiếp tục tạo thêm Entry phụ
* Tên lưu đúng như hiển thị
* Hiển thị được các kí tự (+, [,], =,…) mà format 8.3 không hỗ trợ

## Bước 5 : Đọc RDET và tạo cây thư mục

Cấu trúc cây thư mục :

* Mỗi phần tử là CFolder gồm có :
  + Tên, trạng thái, chuỗi cluster, kích thước
  + subItem: mảng chứa thư mục con là các Cfolder

Thuật toán tạo cây thư mục

Bước 1: Lấy offset bắt đầu của RDET từ công thức ([Các thông số tính toán được-2.](#_Các_thông_số)).

* startOffset = SRDET \* bytePerSector

Tạo biến CFolder “root” để thể hiện cây thư muc:

* Tên: tên phân vùng (A~Z)
* Trạng thái: Directory
* Chuỗi Cluster: Cluster bắt đầu của RDET
* Kích thước: 0
* subItem: rỗng

Bước 2: Từ offset bắt đầu và đưa vào hàm sau để có được thông tin về số Entry chính và Entry phụ:

**function** numberOfFile (interger *offset*) : **return** interger array

Entry *entry*

*vfat* ← 0;

*result* ← array of interger

**while not** entry.name[0] is 0 **do**

readEntryFromOffset(*entry*, *offset*)

*offset* ← *offset* + sizeof(*entry*)

**if** *entry*.attr is 0x0Fh

increase *vfat* by 1

**else**

add vfat to *result*

vfat ← 0

**return** *result*

Bước 3: Dùng mảng *result* trả về, tạo mảng Cfolder có kích thước bằng kích thước mảng *result*:

Bước 3.1: với mỗi phần tử trong *result* đọc 32-byte theo cấu trúc Entry phụ và chuyển hóa thành tên rồi lưu vào mảng tên dài và đánh dấu cờ có Entry phụ.

Bước 3.2: Đọc 32-byte kế tiếp và lưu theo cấu trúc Entry chính.

Bước 3.3: Nếu có Entry phụ thì thay thế tên của Entry chính thành Entry phụ.

Bước 3.4: Từ dãy binary chuyển hóa thành các trạng thái theo công thức ([Thuộc tính (0.0.A.D.V.S.H.R)](#_Thuộc_tính_(0.0.A.D.V.S.H.R))):

Bước 3.5: Từ cluster bắt đầu của Entry chính dò bảng FAT và tạo nên mảng các cluster liên tiếp theo công thức ([Truy xuất chuỗi cluster](#_Truy_xuất_chuỗi)).

Bước 3.5: Tạo Cfolder với các tham số: tên, trạng thái, mảng cluster liên tiếp, kích thước và đẩy vào mảng CFolder.

Bước 4: Cập nhật mảng subItem của root thành mảng Cfolder

Bước 5: Với mỗi phần tử trong subItem

Bước 5.1: Nếu trạng thái là thư mục quay lại bước 2 với các thông số sau:

* + root: phần tử hiện tại
  + offset: cluster đầu tiên trong mảng cluster \* sectorPerCluster \* bytePerSector.

Bước 5.2: Nếu không phải thư mục thì không làm gì

## Bước 6: Khai thác thông tin của thư mục

* Cây thư mục: lưu tên thư mục, trạng thái, cluster bắt đầu, kích thước.
* Nội dung thư mục/tập tin: Hiển thị metadata và:
  + Thư mục: cây thư mục từ subItem
  + Tập tin: đọc từng byte với mỗi cluster trong mảng cluster và in ra nội dung.

# Các bước đọc ổ đĩa dạng NTFS

## Bước 1: Lấy file cần đọc

* Đọc volume hệ thống tương ứng trong máy tính theo định dạng tên file truyền vào là [\\\\.\\X:](file:///\\\\.\\X:) với X là tên của ổ đĩa cần đọc.
* Chương trình cần phải chạy với phân quyền người quản trị để đọc được các thông tin ổ đĩa.
* Nếu không tồn tại thì thoát chương trình.

## Bước 2: Đọc Volume Boot Sector

* Đọc 512-byte đầu tương ứng với VBR (Volume Boot Record)
* Các thông tin chính cần quan tâm:

|  |  |  |
| --- | --- | --- |
| **Offset** | **Size** | **Thông tin trong VBR** |
| 0x3 | 8 | OEM ID (Sẽ là ‘NTFS ’) |
| 0xB | 2 | Bytes/sector |
| 0xD | 1 | Sectors/cluster |
| 0xE | 2 | Số sector dự trữ |
| 0x28 | 8 | Tổng số sector |
| 0x30 | 8 | Cluster bắt đầu của $MFT |
| 0x38 | 8 | Cluster bắt đầu của $MFTMirr |
| 0x40 | 1 | Kích thước của một $MFT record |

* Kích thước của mỗi một MFT record ở trên sẽ là một con số dạng bù 2, và sẽ cần phải lấy 2 lũy thừa với trị tuyệt đối của giá trị trên để có kích thước thực tế.

VD: *(Theo ví dụ tham khảo phần giải đáp của thầy Long)*

Tại 0x40 có là 0xF6, là một con số bù 2 nên quy đổi ra hệ thập phân là -10.

Do đó, kích thước là (byte)

* Để cho thuận tiện, cluster bắt đầu của MFT đầu tiên nên được quy đổi ra byte: (Cluster bắt đầu) \* (bytes/sector) \* (sectors/cluster)

## Bước 3: Đọc MFT

* Đưa con trỏ đọc đến vị trí bắt đầu đã tính ở trên.
* Đọc MFT đầu tiên để lưu giá trị tổng số sector để lưu các MFT record.  
  (Tại offset 0x118, độ dài là 8)
* Tính tổng số MFT record mà có trong ổ đĩa.  
  (Lấy tổng số sector đó chia cho sectors/record, thường mỗi record sẽ chiếm 1024 bytes)
* Lần lượt đọc qua các MFT record tiếp theo.

## Bước 4: Đọc các MFT record header

* Quá trình đọc MFT record cần chú ý đọc qua 4 phần chính, phần header, attribute Standard Information, attribute File Name, attribute Data.
* Ở phần đầu tiên là header

|  |  |  |
| --- | --- | --- |
| **Offset** | **Size** | **Thông tin của MFT record header** |
| 0x0 | 4 | Dấu hiệu nhận biết MFT entry.  - “FILE”: MFT record bình thường.  - “BAAD”: MFT record lỗi. |
| 0x16 | 2 | Giá trị cờ báo  - 0x01: MFT record của một file.  - 0x02: MFT record của một thư mục. |
| 0x2C | 4 | Số hiệu của MFT record này. |
| 0x14 | 2 | Vị trí bắt đầu của attribute đầu tiên (Standard Information) |

* Các attribute sẽ được sắp xếp liên tiếp nhau nên vị trí bắt đầu của attribute kế tiếp sẽ là kết thúc của attribute hiện tại.
* Trong từng các attribute đều có phần header riêng của mình, các thông tin cần chú ý ở đây là:

|  |  |  |
| --- | --- | --- |
| **Offset** | **Size** | **Thông tin trong attribute header** |
| 0x0 | 4 | Mã của attribute |
| 0x4 | 4 | Kích thước của attribute này |
| 0x8 | 1 | Cờ báo giá trị resident  - 0x00: Resident  - 0x01: Non-resident |

## Bước 5: Standard Information

* Mã attribute: 0x10
* Attribute này sẽ luôn là resident.
* Chứa các thông tin liên quan đến thời gian, phiên bản, phân quyền, …

|  |  |  |
| --- | --- | --- |
| **Offset** | **Size** | **Thông tin** |
| 0x20 | 4 | Thông tin phân quyền |

* Các phân quyền cần lưu ý:

|  |  |
| --- | --- |
| **Giá trị** | **Thông tin** |
| 0x0001 | Chỉ đọc (read only) |
| 0x0002 | Ẩn (hidden) |
| 0x0004 | Hệ thống (system) |
| Ngoài ra còn các thông tin khác nữa. | |

Một file có thể có nhiều quyền, cần phải xem coi bit nào bật (phép ‘and’) để tương ứng với quyền đó (không nên dùng phép so sánh bằng).

## Bước 6: File Name

* Mã attribute: 0x30
* Attribute này sẽ luôn là resident.
* Chứa các thông tin liên quan đến tên của file, số hiệu của file cha, phân quyền, …

|  |  |  |
| --- | --- | --- |
| **Offset** | **Size** | **Thông tin** |
| 0x00 | 6 | Số hiệu của MFT record cha |
| 0x38 | 4 | Cờ báo phân quyền  Tương tự như attribute Standard Information, File Name sẽ có thêm:  - 0x10000000: thư mục (directory) |
| 0x40 | 1 | Độ dài của tên file (L) |
| 0x41 | 1 | Namespace của tên file, gồm các giá trị:  - 0: POSIX  - 1: Win32  - 2: DOS  - 3: Win32 & DOS  Từng định dạng namespace khác nhau sẽ có quy định về cách đặt tên và các kí tự đặc biệt khác nhau. |
| 0x42 | 2L | Tên – được định dạng theo chuẩn Unicode (UTF-16le)  (Một kí tự UTF-16 sẽ chiếm 2 bytes) |

* Ngoài ra, có thể sẽ có nhiều hơn một attribute File Name, điều này xảy ra khi namespace của tên file theo chuẩn MS DOS (chuẩn 8.3, còn gọi là tên ngắn, đã được trình bày ở phần FAT32 ở trên).
* Hệ thống nếu lưu tên ngắn sẽ thường đi kèm với tên dài sau đó, nằm trong 1 attribute File Name khác liền kề attribute File Name này.

## Bước 7: Data

* Mã attribute: 0x80
* Attribute này sẽ là resident khi dữ liệu nó chứa ít hơn 700B, sẽ là non-resident khi nhiều hơn.
* Chứa các dữ liệu mà file lưu trữ (là 0 đối với thư mục)
* Nếu MFT record này là resident:

|  |  |  |
| --- | --- | --- |
| **Offset** | **Size** | **Thông tin** |
| 0x10 | 4 | Kích thước phần nội dụng (N) |
| 0x14 | 2 | Offset bắt đầu nội dung (S) |
| S | N | Nội dung |

* Nếu MFT record này là non-resident:

|  |  |  |
| --- | --- | --- |
| **Offset** | **Size** | **Thông tin** |
| 0x30 | 8 | Kích thước phần nội dụng (N) |
| 0x40 | - | Các data-runs (đề cập ở phần tiếp theo) |

## Bước 8: Xây dựng cây thư mục

* Ở từng MFT record đều có các số hiệu của MFT hiện tại và MFT cha.
* Từ đó có thể xác định thứ tự sắp xếp của các file, folder.
* Ngoài ra, có thể xác định được vị trí của MFT record gốc dựa trên mục số hiệu và số hiệu của cha là giống nhau, hoặc là đọc dựa trên attribute Index Root (mã attribute 0x90).

# Cách đọc các data-runs

* Các data-run được cấu trúc gồm 3 phần: Header – Length – Offset.
* Các data-run được xếp liền kề nhau, và phía sau data-run cuối cùng sẽ là 0x00, thông báo không còn data-run nào khác nữa.

Header có dạng 0xXY với:

* Y: số byte tiếp theo là của phần length (bắt đầu từ sau header)
* X: số byte tiếp theo của phần offset (bắt đầu sau phần length)

Ví dụ, giả sử chúng ta có các dãy byte sau:

11 03 07 32 AD 11 4C 1A 01 00 00 ….

Ở đây chúng ta có 2 data-runs:

* Data-run 1:
  + 1 byte cho phần length, 03h = 3 clusters.
  + 1 byte là phần offset, bắt đầu tại cluster thứ 07h = 7.
* Data-run 2:
  + 2 bytes cho phần length, AD 11, 11ADh = 4525 clusters.
  + 3 bytes offset, bắt đầu tại cluster 4C 1A 01, 011A4Ch = 72268.

Phía sau data-run thứ 2 là 00, tức là không còn data-run nào khác nữa.

# Demo

### A screenshot of a computer Description automatically generatedMàn hình chính

### Thông tin từng phân vùng

A screenshot of a computer

Description automatically generatedFAT32

A screenshot of a computer

Description automatically generatedNTFS

### Các câu lệnh cơ bản (dùng chung cho cả FAT32 và NTFS)

#### A screen shot of a computer Description automatically generatedhelp: In ra tất cả các lệnh được hỗ trợ

Ngoài ra, người dùng có thể nhập các lệnh “?”, “-h” hoặc “—help” với chức năng tương tự như lệnh “help”

#### info: In ra thông tin phân vùng của ổ đĩa

Như đã trình bày ở phần thông tin từng phân vùng.

#### A black screen with white text Description automatically generateddir: Liệt kê tất cả các thư mục và tập tin trong thư mục hiện tại

Hơn thế, chương trình cũng hỗ trợ thêm 1 tham số sau lệnh dir để hỗ trợ người dùng nếu họ có nhu cầu in ra thêm các file hoặc thư mục ẩn, file hệ thống hoặc in ra tất cả các file hoặc thư mục có thể có trong thư mục hiện tại.

* A black screen with white text

  Description automatically generatedTham số -h: In ra các file hoặc thư mục ẩn

*Note: HihiHaha là một thư mục ẩn*

* A black screen with white text

  Description automatically generatedTham số -s: In ra các file hệ thống

*Note: NAMRONA.LDO là một file hệ thống*

* A screenshot of a computer

  Description automatically generatedTham số -a: In ra tất cả các file hoặc thư mục có thể có trong ổ đĩa

Ngoài ra, người dùng có thể nhập “ls” với chức năng cùng các hỗ trợ tương tự như lệnh “dir”

#### A black screen with white text Description automatically generatedcd: Di chuyển đến thư mục khác

***2 trường hợp đặc biệt:***

*cd .*

A black screen with white text

Description automatically generated Giữ nguyên thư mục hiện tại

*cd ..*

A black screen with white text

Description automatically generatedQuay lại thư mục cha trực tiếp của thư mục hiện tại

Hơn thế, chương trình cũng hỗ trợ thêm một vài kiểu tham số khác sau lệnh cd để hỗ trợ người dùng.

A screenshot of a computer

Description automatically generated

* cd [Path]

Cho phép người dùng di chuyển đến các thư mục con nằm sâu bên trong nhiều thư mục con khác thông qua đường dẫn

A screenshot of a computer

Description automatically generated

* cd -i [ID]

(hoặc cd --index [ID])

Khi biết ID của 1 thư mục (biết được thông qua lệnh dir hoặc ls), người dùng có thể gán nó vào tham số [ID] của lệnh trên.

*Ghi chú: dùng cho trường hợp tên bị lỗi hiện thị do các kí tự Unicode không được hỗ trợ ở C++17*

#### A black screen with white text Description automatically generatedpwd: In ra đường dẫn của thư mục hiện tại

#### A black screen with white text Description automatically generatedtree: In cây thư mục của thư mực hiện tại

Tương tự như lệnh “dir” đã đề cập ở trên người dùng có thể thêm các tham số để chọn lọc nguồn được in ra.

* Tham số -h: In ra các file hoặc thư mục ẩn
* Tham số -s: In ra các file hệ thống
* Tham số -a: In ra tất cả các file hay thư mục có thể có trong thư mục hiện tại

#### read: Đọc thông tin thư mục hoặc tập tin

* *Thư mục*

Đối với đối tượng thư mục, chương trình cho phép người dùng chỉ nhập lệnh “read” để đọc thông tin của thư mực hiện tại, hoặc lệnh “read [Tên thư mục]” nếu muốn đọc thông tin của 1 thư mục con trong thư mục hiện tại.

Thông tin của thư mục được đọc bởi lệnh read chia làm 2 phần:

+ Info: Thông tin phân vùng

+ Content: Cây thư mụcA screenshot of a computer

Description automatically generated

*(Đọc thông tin thư mục hiện tại)*

*A screenshot of a computer

Description automatically generated(Đọc thông tin thư mục con)*

A screenshot of a computer

Description automatically generatedNgoài ra, giống như lệnh cd, lệnh read cũng được hỗ trợ tính năng đọc theo ID của chương trình, thông qua lệnh read -i [ID] (hoặc read --index [ID])

* *Tập tin*

Đối với đối tượng tập tin, chương trình cho phép người dùng nhập lệnh “read [Tên tập tin]” để đọc thông tin của tập tin muốn tìm, hoặc có thể “read -i [ID]” nếu người dùng biết ID của tập tin

Thông tin của tập tin được đọc bởi lệnh read chia làm 2 phần:

+ Info: Thông tin phân vùng

+ Content: Nội dung của tập tin (lưu ý: chỉ đọc được các tập tin txt, với các đuôi mở rộng khác, chương trình sẽ thông báo người dùng sử dụng phần mềm khác phù hợp để đọc)

A black screen with white text

Description automatically generatedread [Tên tập tin]

A screenshot of a computer

Description automatically generatedread -i [ID]

A screenshot of a computer

Description automatically generatedTuy nhiên, nếu tập tin được đọc không có đinh dạng text (đuôi txt), chương trình sẽ in ra thông báo yêu cầu dùng phần mềm khác để đọc được nội dung của tập tin này.

#### A black screen with white text Description automatically generatedexit: Thoát chương trình

Ngoài ra, chương trình còn hỗ trợ “quit” hoặc “bye” với chức năng tương tự như “exit”

#### cls: Xoá màn hình chương trình

A black screen with a black border

Description automatically generatedA computer screen shot of a black screen

Description automatically generatedNgoài ra, chương trình cũng hỗ trợ lệnh “clear” với chức năng tương tự như “cls”

# Các điểm hạn chế và hướng cải thiện

* Ngôn ngữ lập trình C, C++ phiên bản 2017 vẫn chưa hỗ trợ các kí tự Unicode nên khi in ra có thể sẽ xảy ra lỗi (Xuất hiện các kí tự khác).
* Chương trình cần đọc tất cả các Entry (đối với Fat32) và các MFT record (NTFS) nên khi ổ đĩa quá to, sẽ tốn một ít thời gian để chương trình có thể đọc xong.
* Đọc một cách Dynamic

# Tài liệu tham khảo

<https://flatcap.github.io/linux-ntfs/ntfs>

<https://sabercomlogica.com/en/ntfs-mft-metadata-files/>

<https://legiacong.blogspot.com/>