1. Boot Sector

- Vùng Boot sector bao gồm một số sector đầu tiên của phân vùng (partition), trong đó:

+ sector đầu tiên (Boot Sector), Boot Sector của FAT32 thường có kích thước là 512 byte, chứa thông tin, thông số quan trọng của phân vùng và chứa một đoạn mã chương trình nhỏ để nạp hệ điều hành khi khởi động máy. Các thông tin quan trọng Boot sector lưu trữ bao gồm:

+Mã máy tính cơ bản: Boot sector chứa các hướng dẫn cho máy tính về cách bắt đầu quá trình khởi động. Đoạn code này có nhiệm vụ tải hệ điều hành để máy tính có thể chạy

+Thông tin về hệ điều hành: Boot sector bao gồm các thông tin liên quan đến hệ điều hành được cài đặt trên thiết bị lưu trữ. Điều này giúp máy tính xác định và tương tác với hệ điều hành một cách chính xác

+Kiểm tra tính toàn vẹn: Trong boot sector, có các thông tin được sử dụng để kiểm tra tính toàn vẹn của chính nó và của thiết bị lưu trữ. Nếu boot sector bị hỏng, máy tính có thể gặp khó khăn trong quá trình khởi động, làm ảnh hưởng đến sự ổn định của hệ thống.

+Các sector còn lại: chứa thông tin hỗ trợ việc xác định tổng số cluster trống, chứa một sector bản sao của Boot sector.

2. Cấu trúc của Boot Sector:



*Nguồn: slide tài liệu của thầy Lê Viết Long*

-File bootSector.h

A screenshot of a computer program

Description automatically generated

-Lớp BootSector bao gồm các thuộc tính:, Sc: số sector trên cluster, Sb: số sector thuộc vùng boot sector, Nf: số lượng bảng FAT, Sv: kích thước volume, Sf: kích thước mỗi bảng FAT, startClusterRDET: cluster bắt đầu của RDET, numOfSectorExtra: sector chứa thông tin phụ, numOfSectorSave: sector chứa bản lưu của Boot sector, typeFAT: loại FAT

-Các phương thức (được định nghĩa trong file readBootSector.cpp):

+BootSector(): set thông tin các thuộc tính đều bằng 0

+readBootSector(Byte sector[512]): hàm này dùng để đọc các thông tin của một boot sector thông qua một mảng 512 byte

A screenshot of a computer

Description automatically generated

* Sc (Sectors per cluster) từ offset 0x0D trong boot sector.
* Sb (Sector of boot sector) từ offsets 0x0E và 0x0F trong boot sector, sử dụng 2 bytes.
* Nf (Number of FATs) từ offset 0x10 trong boot sector.
* Sv (Number of sectors in the volume) từ offsets 0x20 đến 0x23 trong boot sector, sử dụng 4 bytes.
* Sf (Sectors per FAT) từ offsets 0x24 đến 0x27 trong boot sector, sử dụng 4 bytes.
* startClusterRDET (Start cluster of RDET) từ offsets 0x2C đến 0x2F trong boot sector, sử dụng 4 bytes.
* numOfSectorExtra (Number of sectors for extra data) từ offsets 0x30 và 0x31 trong boot sector, sử dụng 2 bytes.
* numOfSectorSave (Number of sectors for save data) từ offsets 0x32 và 0x33 trong boot sector, sử dụng 2 bytes.
* typeFAT (Type of FAT) từ offsets 0x52 đến 0x59 trong boot sector, sử dụng 8 bytes.

+kĩ thuật được dùng trong hàm này là phép dịch bit sang trái. Phép dịch bit sang trái di chuyển giá trị của một biểu thức hoặc một giá trị sang phía trái một số lượng bit cụ thể. Ví dụ:

* Sb = sector[14] + (sector[15] << 8); Dấu "<< 8" được sử dụng để dịch giá trị của sector[15] sang trái 8 bit. Điều này có nghĩa là giá trị của sector[15] được nhân với 2^8 (256). Sau đó, giá trị của sector[14] được cộng vào kết quả. Cách này được sử dụng để kết hợp giá trị của hai byte liên tiếp trong mảng thành một giá trị 16-bit. Tương tự, phép dịch bit sang trái có thể được sử dụng với số lượng bit khác nhau tùy thuộc vào cách dữ liệu được tổ chức trong boot sector.

+Hàm printBootSector trong lớp BootSector có nhiệm vụ in ra các thông tin quan trọng từ boot sector, giúp hiển thị chi tiết về cấu trúc và thuộc tính của hệ thống tập tin FAT32.

A computer screen shot of a computer code

Description automatically generated

* Phần chuyển typeFAT từ dạng số sang dạng chuỗi:
  + Khởi tạo một chuỗi rỗng “fatTypeString” để chứa kết quả cuối cùng
  + Vòng lặp for: for (int i = 0; i < 8; ++i) { ... }: Vòng lặp từ 0 đến 7 (8 lần) để xử lý từng byte của giá trị typeFAT.
  + Dịch bit và chuyển đổi:fatTypeString += static\_cast<char>((typeFAT >> (i \* 8)) & 0xFF);:
    - (typeFAT >> (i \* 8)): Dịch bit sang phải i \* 8 vị trí để lấy ra từng byte của typeFAT.
    - & 0xFF: Áp dụng phép AND với 0xFF để giữ lại chỉ 8 bit cuối cùng (0xFF có tất cả các bit ở 8 bit cuối là 1).
    - static\_cast<char>(...): Chuyển đổi giá trị kết quả thành kiểu char để thêm vào chuỗi fatTypeString.
  + Tạo chuỗi kết quả: Mỗi lần lặp, một byte của typeFAT được chuyển đổi thành ký tự và thêm vào chuỗi fatTypeString

II. Attribute Standard Information

1. Mô tả

Attribute $STANDARD\_INFORMATION xuất hiện trong mọi tập tin và thư mục, lưu trữ các thông tin quan trọng như thời gian, ngày tháng, quyền sở hữu, phân quyền sử dụng, và hạn ngạch đĩa. Mặc dù những thông tin này không đặc biệt quan trọng cho quá trình lưu trữ tập tin, nhưng chúng lại đóng vai trò quan trọng trong các ứng dụng của hệ điều hành Windows.

Mã loại (type ID) mặc định của attribute này là 16. Trong Windows 2000 và XP, kích thước của attribute là 72 byte, trong Windows NT là 48 byte, và trong Windows 7 là 96 byte.

2. Cấu trúc

|  |  |
| --- | --- |
| **Byte thứ** | **Mô tả** |
| 0 – 15 | Header của attribute $STANDARD\_INFORMATION. |
| 16 – 19 | Kích thước phần nội dung của attribute $STANDARD\_INFORMATION. 0x00000048 =  72 (byte). |
| 20 – 21 | Nơi bắt đầu (offset) của phần nội dung attribute $STANDARD\_INFORMATION. 0x0018 = 24 (byte thứ 24 tính từ đầu attribute). |

* Cấu trúc phần header:

|  |  |  |
| --- | --- | --- |
| **Byte thứ** | **Giá trị (Hệ 16 – Hệ 10)** | **Mô tả** |
| 0 – 3 | 0x00000010 – 16 | Mã loại là 16. |
| 4 – 7 | 0x00000060 – 96 | Kích thước của attribute $STANDARD\_INFORMATION là 96 byte. |
| 8 – 8 | 0x00 – 0 | Attribute thuộc kiểu resident. |
| 9 – 9 | 0x00 – 0 | Attribute này không được đặt tên, nên không có giá trị chiều dài của tên. |
| 10 – 11 | 0x0000 – 0 | Attribute này không được đặt tên, nên không có thông tin về vị trí của tên. |
| 12 – 13 | 0x0000 – 0 | Giá trị cờ báo. |
| 14 – 15 | 0x0000 – 0 | Định danh của attribute (attribute ID) $STANDARD\_INFORMATION là 0. |

-Cấu trúc của phần nội dung attribute $STANDARD\_INFORMATION được thể hiện ở bảng sau.

|  |  |  |
| --- | --- | --- |
| **Byte thứ** | **Giá trị hệ 16 – Hệ 10** | **Mô tả** |
| 0 – 7 | 0x01CF 352F 00BB 73E4 -130381390209053668 | Thời gian tạo tập tin: Saturday, March 1, 2014 4:17:01AM UTC. |
| 8 – 15 | 0x01CF 352E EDCA 04D0 – 130381389891241168 | Thời gian thay đổi mới nhất nội dung hai attribute $DATA hoặc $INDEX: Saturday, March 1, 2014 4:16:29AM UTC. |
| 16 – 23 | 0x01CF 352F 0488 3354 – 130381390272803668 | Thời gian thay đổi mới nhất thông tin mô tả tập tin: Saturday, March 1, 2014 4:17:07AM UTC. |
| 24 – 31 | 0x01CF 352F 00BB 73E4 – 130381390209053668 | Thời gian truy cập nội dung tập tin mới nhất: Saturday, March 1, 2014 4:17:01AM UTC. |
| 32 – 35 | 0x00000020 | Giá trị cờ báo, tập tin được đánh dấu là archive. (xem bảng về các giá trị của cờ ở bên dưới). |
| 36 – 39 | 0x00000000 | Maximum number of versions |
| 40 – 43 | 0x00000000 | Version number |
| 44 – 47 | 0x00000000 | Class ID |
| 48 – 51 | 0x00000000 | Định danh sở hữu - Owner ID (từ phiên bản 3.0 về sau). |
| 52 – 55 | 0x000002D9 | Định danh bảo mật - Security ID (từ phiên bản 3.0 về sau). Lưu ý: đây không phải là SID trong Windows. |
| 56 – 63 | 0x00000000 00000000 | Thông tin về hạn ngạch - Quota charged (từ phiên bản 3.0 về sau) |
| 64 – 71 | 0x0000 0000 00BA EC10 | Giá trị của USN (update sequence number) (từ phiên bản 3.0 về sau) |

-file StandardInfo.h gồm:

+Class bao gồm các thuộc tính: flag: giá trị cờ, header: thuộc type NTFSAtrributeHeader (sẽ được giải thích ở phần sau), offset: nơi bắt đầu phần nội dung của thuộc tính, vector<string> entry

-Các method (được viết trong file StandardInfo.cpp):

A black and white screen

Description automatically generated

* this->entry = entry;: Lưu trữ vector chuỗi entry vào thành viên dữ liệu entry của đối tượng StandardInfo.
* this->header = NTFSAttributeHeader(entry);: Tạo một đối tượng NTFSAttributeHeader từ dữ liệu entry và lưu trữ nó trong thành viên dữ liệu header.

this->offset = convertHexadecimalToDecimal(convertStringToLittleEdian(getStringFromVector(this->entry, 20, 21 - 20 + 1)));

* Sử dụng hàm getStringFromVector để trích xuất một chuỗi từ vector <entry>, lấy giá trị tại byte 20-21.
* Chuyển đổi chuỗi hex thành số thập phân bằng convertHexadecimalToDecimal.
* Lưu kết quả vào offset để xác định vị trí bắt đầu phần nội dung

this->flag = convertHexadecimalToDecimal(convertStringToLittleEdian(getStringFromVector(entry, this->offset + 32, 35 - 32 + 1)));

* Lấy giá trị byte 32 đến 35 ở phần nội dung attribute

-Hàm này giúp hiển thị thông tin chi tiết và đọc được từ thuộc tính Standard Information, giúp người đọc dễ dàng hiểu và phân tích các thuộc tính của hệ thống tập tin NTFS.

A screen shot of a computer

Description automatically generated

* Sử dụng câu lệnh switch để xác định giá trị của flag và in ra mô tả tương ứng.
* Mỗi giá trị flag tương ứng với một trạng thái, ví dụ: "Read Only," "Hidden," "System," và các trường hợp khác.
* Nếu giá trị flag không khớp với bất kỳ trường hợp nào, in ra "Unknown".

NGUỒN:

* Slide tài liệu Hệ điều hành của thây Lê Viết Long, ĐHKHTN-ĐHQG HCM
* https://legiacong.blogspot.com/2014/05/he-thong-quan-ly-tap-tin-ntfs-8.html