**TRƯỜNG ĐẠI HỌC ĐÀ LẠT**

**KHOA CÔNG NGHỆ THÔNG TIN**

🙢❖🙠

**BÁO CÁO MÔN HỌC**

TÊN ĐỀ TÀI: “XUẤT THÔNG TIN CỦA THƯ MỤC TRÊN ĐĨA MỀM BAO GỒM KÍCH THƯỚC THẬT VÀ KÍCH THƯỚC TRÊN ĐĨA”

Giảng viên hướng dẫn: Đặng Thanh Hải

Sinh viên thực hiện: 1610207 - La Quốc Thắng

**Đà Lạt, tháng 12 năm 2018**

# Mục lục

[I. Mục lục 2](#_Toc532303212)

[II. Tổng quan 4](#_Toc532303213)

[1. Khái quát về hệ điều hành 4](#_Toc532303214)

[2. Hệ thống quản lý tập tin 5](#_Toc532303215)

[3. Cấu tạo bộ nhớ lưu trữ 6](#_Toc532303216)

[4. Giới thiệu về chương trình 7](#_Toc532303217)

[III. Cài đặt chương trình 8](#_Toc532303218)

[1. Bảng tham số đĩa 8](#_Toc532303219)

[2. Cấu trúc FAT 9](#_Toc532303220)

[3. Entry của thư mục, tập tin 10](#_Toc532303221)

[4. Định nghĩa danh sách liên kết 11](#_Toc532303222)

[5. Biến toàn cục 12](#_Toc532303223)

[6. Hàm chức năng 12](#_Toc532303224)

[a) Chèn một nút vào danh sách 12](#_Toc532303225)

[b) Xóa danh sách 12](#_Toc532303226)

[c) So sánh 2 chuỗi 13](#_Toc532303227)

[d) Phân tích đường dẫn 13](#_Toc532303228)

[e) Tìm kiếm thư mục 14](#_Toc532303229)

[f) Đọc đĩa 15](#_Toc532303230)

[g) Đọc bảng tham số đĩa 15](#_Toc532303231)

[h) Đổi sector logic sang sector vật lý 16](#_Toc532303232)

[i) Đọc đĩa theo từng sector logic 16](#_Toc532303233)

[j) Đọc bảng FAT 16](#_Toc532303234)

[k) Tìm cluster kế tiếp của file trong FAT 17](#_Toc532303235)

[l) Lấy các cluster ở thư mục gốc 17](#_Toc532303236)

[m) Lấy các cluster dựa vào cluster bắt đầu 17](#_Toc532303237)

[n) Lấy các entry từ danh sách cluster 18](#_Toc532303238)

[o) Lấy kích thước trên đĩa 19](#_Toc532303239)

[p) In các thông tin của thư mục lên màn hình 19](#_Toc532303240)

[q) Gọi các hàm 19](#_Toc532303241)

[IV. Thực thi chương trình 20](#_Toc532303242)

[V. Đánh giá 24](#_Toc532303243)

[1. Thuận lợi 24](#_Toc532303244)

[2. Hạn chế 24](#_Toc532303245)

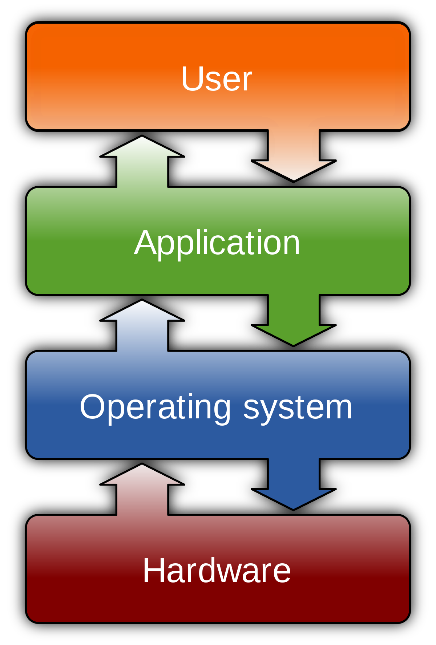
[3. Phương hướng phát triển 24](#_Toc532303246)

[VI. Tài liệu tham khảo 26](#_Toc532303247)

# Tổng quan

## Khái quát về hệ điều hành

Hệ điều hành là một chương trình được xem như trung gian giữa người sử dụng máy tính và phần cứng máy tính với mục đích thực hiện các chương trình giúp cho người dùng sử dụng máy tính dễ dàng hơn, sử dụng phần cứng một cách có hiệu quả.



Hệ điều hành cung cấp một môi trường làm việc cho các chương trình thi hành. Nó cung cấp các dịch vụ cho người sử dụng, giao tiếp với người sử dụng. Các thành phần bên trong của hệ điều hành bao gồm: hệ thống quản lý tiến trình, quản lý bộ nhớ chính, quản lý tập tin, quản lý hệ thống nhập xuất, quản lý hệ thống lưu trữ phụ, hệ thống bảo vệ, hệ thống dòng lệnh.



**Các hệ điều hành phổ biến hiện nay**

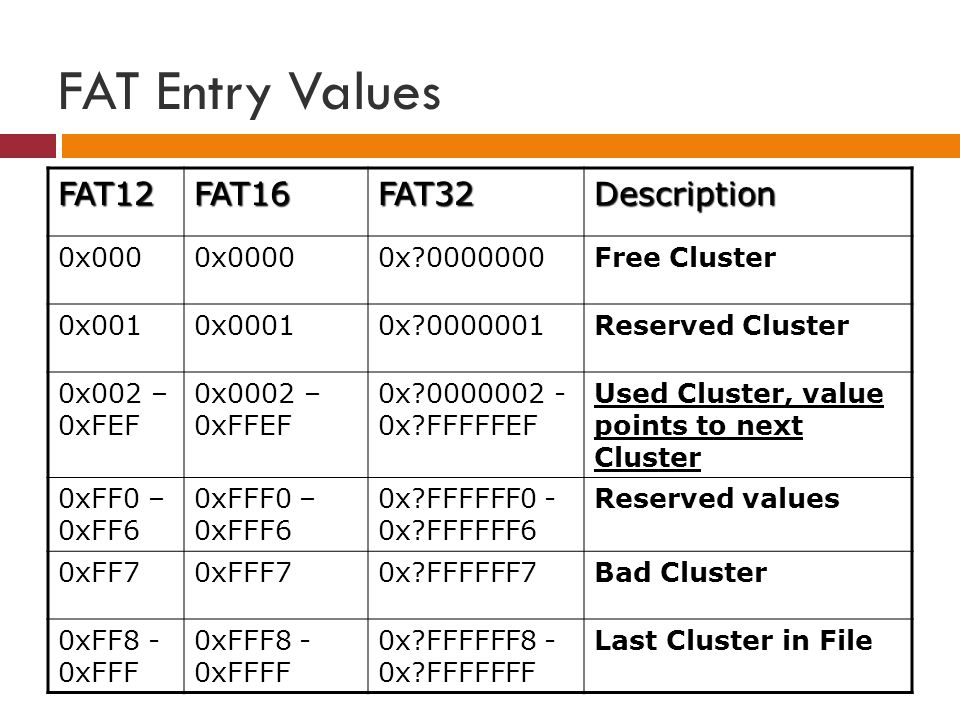
MS-DOS, Hệ điều hành đĩa từ Microsoft, là hệ điều hành đầu tiên chạy trên máy tính cá nhân, đã từng rất phổ biến trong suốt thập niên 1980, và đầu thập niên 1990, cho đến khi Windows 95 ra đời.



**Biểu tượng của MS-DOS**

## Hệ thống quản lý tập tin

Hệ thống quản lý tập tin được sử dụng trong các hệ điều hành đời đầu của Microsoft là FAT, viết tắt của “File Allocation Table”. FAT được giới thiệu lần đầu tiên vào năm 1977 với phiên bản FAT12. Sau đó là các phiên bản FAT16 và FAT32.



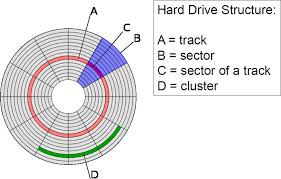
**Bảng so sánh 3 loại FAT**

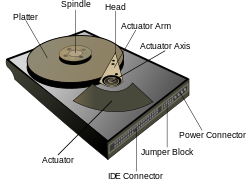
FAT12 được dùng cho ổ đĩa mềm, ổ đĩa có dung lượng từ 32MB trở xuống. FAT12 sử dụng 12 bit để đếm nên chỉ có khả năng quản lý các ổ đĩa có dung lượng thấp hơn 32MB với số lượng cluster thấp.

How the FAT file system organizes a volume

**Minh họa hệ thống tệp FAT trên một ổ đĩa**

## Cấu tạo bộ nhớ lưu trữ





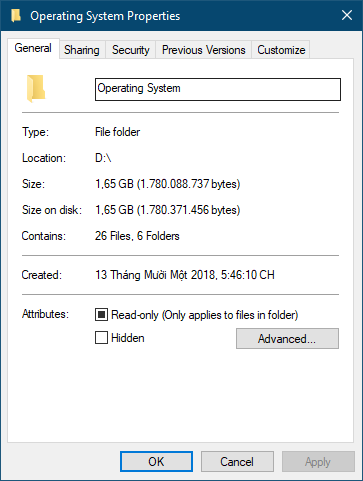
**Cấu tạo đĩa cứng**

Ổ đĩa cứng bao gồm nhiều đĩa từ, mỗi đĩa từ có thể sử dụng hai mặt, đĩa cứng có thể có nhiều đĩa từ, chúng gắn song song, quay đồng trục, cùng tốc độ với nhau khi hoạt động. Trên một mặt làm việc của đĩa từ chia ra nhiều vòng tròn đồng tâm thành các track. Trên track chia thành những phần nhỏ bằng các đoạn hướng tâm thành các sector. Tập hợp các track cùng bán kính (cùng số hiệu trên) ở các mặt đĩa khác nhau thành các cylinder.

Khi đĩa được format thì đơn vị nhỏ nhất trên đĩa là sector. Đối đĩa cứng lớn có nhiều sector mà DOS không thể quản lý được. Trong trường hợp này để giảm số sector cần quản lý bằng cách định nghĩa cluster là tập hợp các sector. Lúc này Dos chỉ quản lý cluster thay vì sector.

## Giới thiệu về chương trình

Chúng ta thường quen với việc xem thuộc tính của một tập tin hoặc một thư mục và ít nhiều gì cũng nhìn thấy 2 thuộc tính Size và Size on disk.

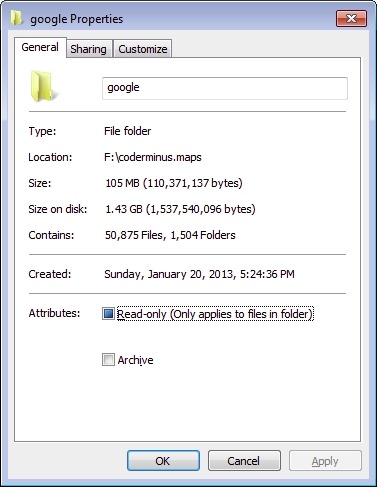


**Ví dụ về thuộc tính của một thư mục**

Như đã đề cập ở phần trên, đơn vị phân bổ (Allocation Units) chính là cluster, giá trị của cluster trải dài từ 512 bytes đến 32 Kbytes.

Một tập tin sẽ chiếm một số hữu hạn cluster riêng biệt và không có cluster nào dùng cho nhiều file. Nếu một tập tin cần 2 clusters để lưu trữ (mỗi cluster có kích thước là 1KB) thì kích thước trên đĩa của nó là 2KB (mặc dù kích thước thật chưa đến 2KB).

Một ví dụ khác, giả sử ta có 50.000 files, với kích thước cluster là 32 KB (giá trị tối đa của FAT32), mỗi file có kích thước chỉ 2KB. Vậy số cluster cần dùng là 50.000 clusters, tổng kích thước trên đĩa là 50.000 x 32.000 = 1.600.000.000 bytes (1,6GB). Nhưng kích thước thật của nó chỉ 50.000 x 2 = 100.000 KB (100 MB).



**Có một sự lãng phí bộ nhớ không hề nhỏ…**

Mục đích của chương trình này chính là xuất thông tin của các tập tin trên đĩa mềm, bao gồm kích thước trên đĩa mà MS-DOS chưa hỗ trợ, cao hơn cả là hiểu được cơ chế quản lý hệ thống tập tin của MS-DOS theo yêu cầu của môn học Hệ điều hành.

# Cài đặt chương trình

## Bảng tham số đĩa

|  |  |  |  |
| --- | --- | --- | --- |
| **Offset (bytes)** | **Tên** | **Kích thước (bytes)** | **Ý nghĩa** |
| 0 | JMP | 3 | Lệnh nhảy đến đọan mồi HĐH |
| 3 | Ver | 8 | Nhận diện OEM |
| 11 | SecSiz | 2 | Bytes/Sector |
| 13 | ClustSiz | 1 | Sectors/Cluster |
| 14 | ResSec | 2 | Số sector trước bảng FAT |
| 16 | FatCnt | 1 | Số bảng FAT trên đĩa |
| 17 | RootSiz | 2 | Số entry của thư mục tối đa trên root |
| 19 | TotSec | 2 | Tổng số sector trên đĩa nhỏ hơn 32MB |
| 21 | Media | 1 | Byte chỉ danh đĩa |
| 22 | FatSiz | 2 | Sector/FAT |
| 24 | TrkSec | 2 | Sector/Track |
| 26 | HeadCnt | 2 | Số head hoặc side |
| 28 | HidSec | 4 | Số sector ẩn |
| 32 | TotSec | 4 | Tổng số sector lớn hơn 32 MB |

Cài đặt:

typedef struct

{

char jmp[3]; //The first byte to JMP SHORT XX NOP

char Ver[8]; //OEM identifier

unsigned SecSiz; //Bytes/sector (in little-endian format)

char ClustSiz; //Sectors/cluster

unsigned ResSec; //Number of reserved sectors (included Boot Record Sectors)

char FatCnt; //Number of File Allocation Tables. Often 2

unsigned RootSiz; //Max # of dictionary entries

unsigned TotSec; //Total logical sectors

char Media; //Indicates the media descriptor type (FAT ID)

unsigned FatSiz; //Number of sectors per FAT

unsigned TrkSec; //Number of sectors per track

unsigned HeadCnt; //Number of heads or sides on the storage media

unsigned HidSec; //Number of hidden sectors (LBA)

} EntryBPB;

//Union between Sector array and BPB structure

typedef union {

char Sec[512];

EntryBPB Entry;

} UnionBPB;

## Cấu trúc FAT

|  |  |
| --- | --- |
| **Giá trị entry** | **Mô tả** |
| 0x?0000000 | Cluster còn trống |
| 0x?0000001 | Dành riêng cho mục đích nội bộ |
| 0x?0000002 - 0x?FFFFFEF | Chứa dữ liệu, giá trị của nó là vị trí của cluster tiếp theo |
| 0x?FFFFFF0 -0x?FFFFFF5 | Dành riêng |
| 0x?FFFFFF6 | Dành riêng, không sử dụng |
| 0x?FFFFFF7 | Bad sector |
| 0x?FFFFFF8 -0x?FFFFFFF | Cluster cuối của một tập tin |

## Entry của thư mục, tập tin

|  |  |  |
| --- | --- | --- |
| **Offset (bytes)** | **Kích thước (bytes)** | **Ý nghĩa** |
| 0 | 8 | Tên file: 8 ký tự ASCII  Tên file có thể kết thúc bằng ký tự 0x00 hoặc 0x20  Nếu tên file bắt đầu là 0x00 thì entry trước chính là entry cuối cùng. Nếu là 0xE5 thì entry bị xóa/ không sử dụng. |
| 8 | 3 | Phần mở rộng: 3 ký tự ASCII Phần mở rộng có thể kết thúc bằng ký tự 0x00 hoặc 0x20 |
| 11 | 1 | Thuộc tính tập tin. Giá trị có thể là: READ\_ONLY=0x01, HIDDEN=0x02, SYSTEM=0x04, VOLUME\_ID=0x08, DIRECTORY=0x10, ARCHIVE=0x20, LFN=READ\_ONLY|HIDDEN|SYSTEM|VOLUME\_ID (LFN - long file name entry) |
| 12 | 1 | Dành riêng |
| 13 | 1 | Thời gian tạo trong 1/10 giây, giá trị từ 0-199 inclusive, Ubuntu dùng 0-100. |
| 14 | 2 | Thời gian tạo file, nhân số giây cho 2.  Bits 15 - 11: giờ. Bits 10 -5: phút. Bits 4 - 0: giây/2. |
| 16 | 2 | Ngày tạo file Bits 15 - 9: Năm (0 = 1980). Bits 8 - 5: Tháng. Bits 4 - 0: Ngày. |
| 18 | 2 | Ngày truy cập cuối cùng. Định dạng giống như ngày tạo |
| 20 | 2 | 16 bit cao của entry thuộc về số cluster đầu tiên của file |
| 22 | 2 | Giờ chỉnh sửa cuối cùng |
| 24 | 2 | Ngày chỉnh sửa cuối cùng |
| 26 | 2 | 16 bit thấp của entry thuộc về số cluster đầu tiên của file |
| 28 | 4 | Kích thước của file |

Cài đặt:

//The file time

typedef struct

{

unsigned S : 5; //Seconds

unsigned M : 6; //Minutes

unsigned H : 5; //Hours

} Time;

//Union between Creation time and Time structure

typedef union {

unsigned intTime;

Time T;

} UnionTime;

//The file date

typedef struct

{

unsigned D : 5; //Days

unsigned M : 4; //Months

unsigned Y : 7; //Years

} Date;

//Union between Creation date and Date structure

typedef union {

unsigned intDate;

Date Dat;

} UnionDate;

//Attribute Byte (8 bits)

typedef struct

{

char ReadOnly : 1; //Indicates that the file is read only

char Hidden : 1; //Indicates a hidden file

char System : 1; //Indicates a system file

char Volume : 1; //Indicates a special entry containing the disk's volume label

char SubDir : 1; //Describes a subdirectory

char Archive : 1; //This is the archive flag

char DR : 2;

} Attribute;

//Union between Attribute char and Attribute structure

typedef union {

char charAtt;

Attribute Attr;

} UnionAttribute;

//Regular Directory Entry

typedef struct

{

char FileName[8]; //The file name, the first byte indicates it's status

char Ext[3]; //The file name extension

UnionAttribute Att; //The file attributes

char DR[10];

UnionTime Tg; //The file time

UnionDate Ng; //The file date

unsigned Clust; //The starting cluster number

long FileSize; //The file size

} EntryDir;

//Union between Entry char array and Entry structure

typedef union {

char Entry[32];

EntryDir Entdir;

} UnionDir;

## Định nghĩa danh sách liên kết

//Define a node in linked list

typedef struct Node

{

void \*Data; //Content of node

Node \*Next; //The pointer of next node

} NodeType;

//Define a linked list

typedef NodeType \*PointerType;

## Biến toàn cục

char \*path; //Duong dan

char drive; //O dia

EntryBPB BPB; //Bang tham so dia

unsigned char \*FAT; //Noi dung bang FAT

long SecStart = 0; //Sector bat dau

## Hàm chức năng

### Chèn một nút vào danh sách

/\*Chen mot nut vao danh sach

- list: Danh sach lien ket

- lastItem: Phan tu cuoi cung cua danh sach

- item: Phan tu can chen

Tra ve:

- true: Thanh cong

- false: That bai

\*/

int InsertLast(PointerType &list, PointerType &lastItem, void \*item)

{

PointerType temp = new NodeType;

if (!temp)

{

cout << "\nLoi cap phat bo nho cho viec chen vao danh sach!";

return 0;

}

temp->Data = item;

temp->Next = NULL;

if (list == NULL)

list = temp;

else

lastItem->Next = temp;

lastItem = temp;

return 1;

}

### Xóa danh sách

/\*Xoa toan bo danh sach lien ket

- list: Danh sach can xoa

\*/

void DeleteList(PointerType &list)

{

PointerType p = list->Next;

while (p != NULL)

{

delete list;

list = p;

p = p->Next;

}

//delete list;

list = NULL;

}

### So sánh 2 chuỗi

/\*So sanh hai chuoi

- source: Chuoi ban dau

- dest: Chuoi thu hai

Tra ve:

- true: bang nhau

- false: khong bang nhau

\*/

int Compare(char \*source, char \*dest)

{

for (int i = 0; source[i] != '\0'; i++)

if (source[i] != dest[i])

return 0;

if (dest[i] == ' ')

return 1;

else return 0;

}

### Phân tích đường dẫn

/\*Lay cac thu muc xuat hien trong duong dan, ngoai tru o dia ra

- path: Chuoi duong dan ma nguoi dung nhap vao

Tra ve:

- NULL: Duong dan khong hop le

- !NULL: Danh sach duong dan

\*/

PointerType AnalysePath(char \*path) {

PointerType list = NULL, last = NULL;

char \*fileName;

drive = path[0];

if (drive >= 97 && drive <= 122)

drive -= 97; //Neu la chu thuong

else if (drive >= 65 && drive <= 90)

drive -= 65; //Neu la chu hoa

else

{

cout << "\nKhong ton tai o dia " + drive;

return NULL;

}

if (drive == 2)

drive = 0x80;

if (drive == 0 || drive == 1 || drive == (char)0x80) //Chi chap nhan o A, B, C

{

if (path[1] != ':' && path[2] != '\\')

{

cout << "\nDuong dan nhap sai";

return NULL;

}

if (path[3] == '\0')

{

cout << "\nDuong dan thieu thu muc";

return NULL;

}

int i = 3;

while (path[i] != '\0') //Doc den het duong dan

{

int j = 0;

fileName = new char[12];

if (!fileName)

{

cout << "\nLoi cap phat bo nho cho ten file";

return NULL;

}

while (path[i] != '\\' && path[i] != '\0') //Doc den het ten cua thu muc

{

fileName[j] = path[i];

++i;

++j;

}

fileName = strupr(fileName); //Doi sang chu hoa

fileName[j] = '\0';

j = 0;

while (fileName[j] != '.' && fileName[j] != '\0') //Lay ten cua thu muc (khong tinh phan mo rong)

++j;

if (fileName[j] == '.') //Neu no la tap tin

{

fileName[j] = ' ';

++j;

int k = 3;

while (fileName[j] != '\0')

{

fileName[11 - k] = fileName[j];

++j;

}

fileName[11] = '\0';

k = 7;

while (fileName[k] != ' ')

{

fileName[k] = ' ';

k--;

}

}

if (path[i] != '\0')

i++;

InsertLast(list, last, fileName);

}

return list;

}

else

{

cout << "\nDuong dan thieu o dia!";

return NULL;

}

}

### Tìm kiếm thư mục

/\*Kiem tra mot thu muc con co xuat hien trong thu muc duoc chi dinh

- listEntry: Danh sach entry cua mot thu muc (goc hoac con)

- fileName: Ten cua thu muc con

- dir: Thuc muc tim duoc (ket qua)

Tra ve:

- true: Tim thay

- false: Khong tim thay

\*/

int SearchDir(PointerType listEntry, char \*fileName, EntryDir &dir)

{

PointerType p = listEntry;

while (p != NULL) //Duyet toan bo entry cua thu muc

{

if (Compare(fileName, ((EntryDir \*)p->Data)->FileName))

{

dir = \*(EntryDir \*)p->Data;

return 1;

}

p = p->Next;

}

return 0;

}

### Đọc đĩa

/\*Doc dia va cho vao buffer

- buff: Du lieu cua o dia (ket qua)

- side: So mat cua dia

- track: So track cua dia

- sector: Vi tri sector

- number: So luong sector can doc

Tra ve:

- true: Thanh cong

- false: Khong thanh cong

\*/

int ReadDiskBIOS(char \*buff, unsigned side, unsigned track, unsigned sector, unsigned number)

{

union REGS u, v; //Thanh ghi da dung

struct SREGS s; //Thanh ghi doan

int i = 0, k; //i: So lan doc toi da

v.x.cflag = 1; //Co carry

while (i < 2 && v.x.cflag != 0)

{

u.h.ah = 0x2; //Doc dia - Neu la 0x1: Ghi dia

u.h.dl = drive; //O dia - Neu la 80: Dia cung

u.h.dh = side;

u.h.cl = sector; //SecTor DAU

u.h.ch = track; //track

u.x.cx = track; //Track va Sector bat dau

u.h.al = number; //So luong sector can doc

s.es = FP\_SEG(buff); //Thanh ghi doan cua buff

u.x.bx = FP\_OFF(buff); //Dia chi offset cua buff

int86x(0x13, &u, &v, &s); //Ham ngat 13h

i++; //Tang so lan doc

}

return !(v.x.cflag); //Kiem tra loi

}

### Đọc bảng tham số đĩa

/\*Doc bang tham so dia BPB va cho vao bien toan cuc BPB

\*/

void ReadBPB()

{

UnionBPB temp;

if (drive == 0 || drive == 1)

if (!ReadDiskBIOS(temp.Sec, 0, 1, 1, 1))

{

cout << "\nKhong doc duoc bang tham so dia";

return;

}

BPB = temp.Entry;

}

### Đổi sector logic sang sector vật lý

/\*Doi sector logic sang sector vat ly

- begin: Sector bat dau

- side: So mat cua dia

- track: So track cua dia

- sector: Vi tri sector

Luu y: Phai gan cho bien toan cuc BPB truoc

\*/

void Change(long begin, unsigned &side, unsigned &track, unsigned &sector)

{

unsigned x;

sector = (unsigned)(1 + begin % BPB.TrkSec);

side = (unsigned)((begin / BPB.TrkSec) % BPB.HeadCnt);

track = (unsigned)(begin / (BPB.TrkSec \* BPB.HeadCnt));

x = track;

x = x & 0xFF00;

x = x >> 2;

x = x & 0x00FF;

x = x | sector;

track = track << 8;

track = track | x;

}

### Đọc đĩa theo từng sector logic

/\*Doc dia theo tung sector logic

- buff: Du lieu cua o dia (ket qua)

- begin: Sector bat dau

- number: So sector can doc

Tra ve:

- true: Thanh cong

- false: That bai

\*/

int ReadDisk(char \*buff, long begin, int number)

{

unsigned side, track, sector;

Change(begin, side, track, sector);

if (ReadDiskBIOS(buff, side, track, sector, number))

return 1;

else return 0;

}

### Đọc bảng FAT

/\*Doc noi dung bang FAT vao bien toan cuc

Luu y: Phai gan gia tri cho bien toan cuc BPB truoc

\*/

int ReadFAT()

{

FAT = new char[BPB.FatSiz \* BPB.SecSiz];

if (!FAT)

{

cout << "\nLoi cap phat bo nho";

return 0;

}

if (!ReadDisk(FAT, BPB.ResSec, BPB.FatSiz))

{

cout << "\nKhong doc duoc bang tham so dia!";

return 0;

}

return 1;

}

### Tìm cluster kế tiếp của file trong FAT

/\*Tim cluster ke tiep cua file trong FAT

- index: So hieu cluster

Tra ve: Vi tri cluster tiep theo cua index

Luu y: Phai gan ket qua cho bien toan cuc FAT truoc

\*/

unsigned NextEntry(unsigned index)

{

unsigned add, x, t;

if (drive == 0 || drive == 1)

{

add = index \* 3 / 2;

x = FAT[add];

t = FAT[add + 1];

t = t << 8;

x = x + t;

if ((index % 2) == 0)

x = x & 0x0FFF; //Neu la chan

else x = x >> 4; //Neu la le

}

else

{

add = (index \* 2) - 1;

x = FAT[add];

t = FAT[add + 1];

t = t << 8;

x = x + t;

}

return x;

}

### Lấy các cluster ở thư mục gốc

/\*Lay danh sach cluster cua thu muc goc

Tra ve: Danh sach cluster

Luu y: Phai gan ket qua cho bien toan cuc BPB truoc

\*/

PointerType GetClusterRoot()

{

PointerType listCluster = NULL, last = NULL;

unsigned rootSec, \*cluster, number;

rootSec = BPB.ResSec + BPB.FatSiz \* (int)BPB.FatCnt;

number = BPB.RootSiz \* 32 / 512;

for (int i = 0; i < number; i++, rootSec++)

{

cluster = new unsigned;

\*cluster = rootSec;

InsertLast(listCluster, last, (unsigned \*)cluster);

}

return listCluster;

}

### Lấy các cluster dựa vào cluster bắt đầu

/\*Lay danh sach cluster dua vao cluster bat dau

- begin: Vi tri cluster bat dau

Tra ve: Danh sach cluster

\*/

PointerType GetCluster(unsigned begin)

{

PointerType listCluster = NULL, last = NULL;

unsigned \*cluster, next = begin;

for (; next >= 2 && next < 0xFEF; next = NextEntry(\*cluster))

{

cluster = new unsigned;

\*cluster = next;

InsertLast(listCluster, last, (unsigned \*)cluster);

}

return listCluster;

}

### Lấy các entry từ danh sách cluster

/\*Doc tung entry ben trong thu muc va cho vao danh sach

- listCluster: Danh sach cluster

- flag: 0 : Thu muc goc

1 : Thu muc con

Tra ve:

- NULL: Khong thanh cong

- !NULL: Danh sach entry cua thu muc

Luu y: Phai gan gia tri cho bien toan cuc BPB, drive truoc

\*/

PointerType GetEntryDir(PointerType listCluster, char flag)

{

PointerType p = listCluster, listEntry = NULL, last = NULL;

unsigned size = BPB.ClustSiz \* 512, currentCluster, currentSector;

unsigned char \*buff = new unsigned char[size];

UnionDir \*dir;

while (p != NULL)

{

currentCluster = \*(unsigned \*)p->Data;

if (flag)

currentSector = BPB.ResSec + BPB.FatSiz \* BPB.FatCnt + (BPB.RootSiz \* 32) / 512 + (currentCluster - 2) \* BPB.ClustSiz;

else

currentSector = currentCluster;

if (!ReadDisk(buff, currentSector, BPB.ClustSiz))

{

cout << "\nKhong doc duoc dia voi danh sach cluster nay";

return NULL;

}

for (int i = 0; i < size;)

{

if (buff[i] != (char)0x00) //Entry khac trong

if (buff[i] != (char)0xE5) //Entry khong bi xoa

{

int j = 0;

dir = new UnionDir;

for (; j < 32; j++, i++)

dir->Entry[j] = buff[i];

InsertLast(listEntry, last, &dir->Entdir);

}

else i += 32;

else break;

}

p = p->Next;

}

delete buff;

return listEntry;

}

### Lấy kích thước trên đĩa

/\*Lay kich thuoc tren dia dua vao kich thuoc that

- realSize: Kich thuoc cua file

Tra ve:

- Kich thuoc tren dia

\*/

long GetSizeOnDisk(long realSize)

{

long size = (long)BPB.SecSiz \* BPB.ClustSiz; //Bytes/cluster

if ((realSize % size) != 0)

return ((realSize / size) + 1) \* size;

else

return realSize;

}

### In các thông tin của thư mục lên màn hình

/\*Hien thi noi dung cua thu muc len man hinh

- listEntry: Danh sach entry can hien thi thong tin

\*/

void PrintDir(PointerType listEntry)

{

PointerType p = listEntry;

EntryDir dir;

cout << '\n'

<< setiosflags(ios::left)

<< setw(8) << "Name"

<< setw(8) << "Ext"

<< setw(16) << "Size (bytes)"

<< setw(16) << "On disk (bytes)"

<< setw(8) << "Time"

<< setw(12) << "Date";

while (p != NULL)

{

dir = \*(EntryDir \*)p->Data;

if (dir.Att.Attr.Volume == 0)

{

int i;

cout << "\n";

for (i = 0; i < 8; i++)

cout << dir.FileName[i];

for (i = 0; i < 3; i++)

cout << dir.Ext[i];

cout << '\t' << dir.FileSize

<< "\t\t" << GetSizeOnDisk(dir.FileSize)

<< "\t\t" << ((dir.Tg).T).H << ':' << ((dir.Tg).T).M

<< '\t' << ((dir.Ng).Dat).D << '/' << ((dir.Ng).Dat).M << "/" << ((dir.Ng).Dat).Y + 1980;

}

p = p->Next;

}

}

### Gọi các hàm

void ChayChuongTrinh()

{

path = new char[256];

cout << "\nNhap duong dan: ";

scanf("%s", path);

PointerType listPath = AnalysePath(path);

if (listPath == NULL)

return;

ReadBPB();

if (!ReadFAT())

return;

//Tai thu muc goc

PointerType listCluster = GetClusterRoot();

PointerType listEntry = GetEntryDir(listCluster, 0);

//Tai thu muc con

while (listPath)

{

EntryDir dir;

if (SearchDir(listEntry, (char \*)listPath->Data, dir))

{

listCluster = GetCluster(dir.Clust);

listEntry = GetEntryDir(listCluster, 1);

}

else

{

cout << "\nKhong tim thay thu muc " << (char \*)listPath->Data;

return;

}

listPath = listPath->Next;

}

PrintDir(listEntry);

}

int main()

{

clrscr();

ChayChuongTrinh();

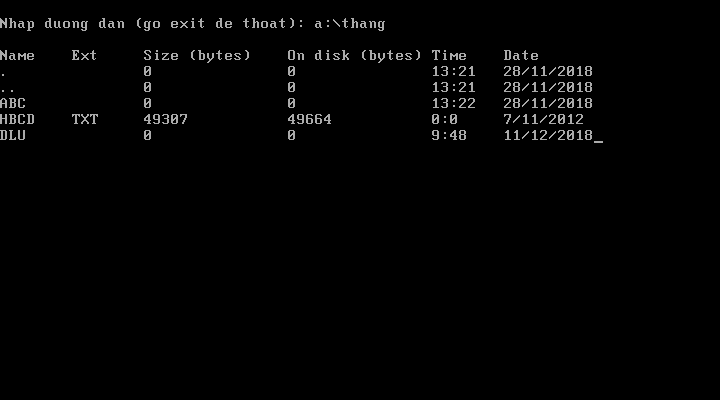
getch();

return 1;

}

# Thực thi chương trình

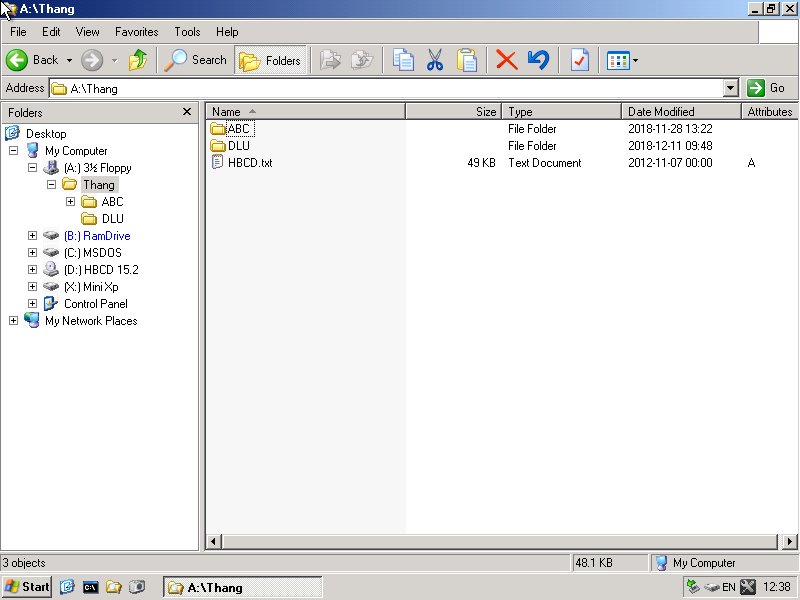
* Kết quả từ chương trình:

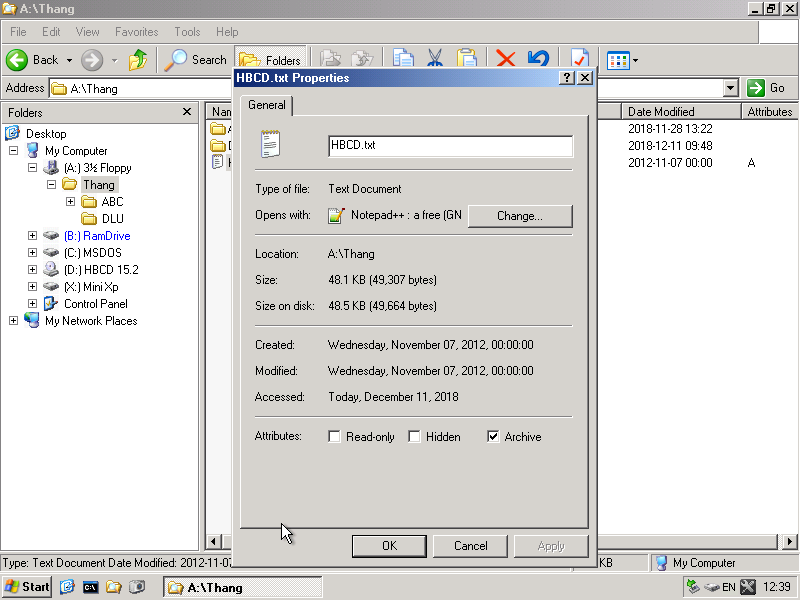


* So sánh với lệnh dir của MS-DOS:

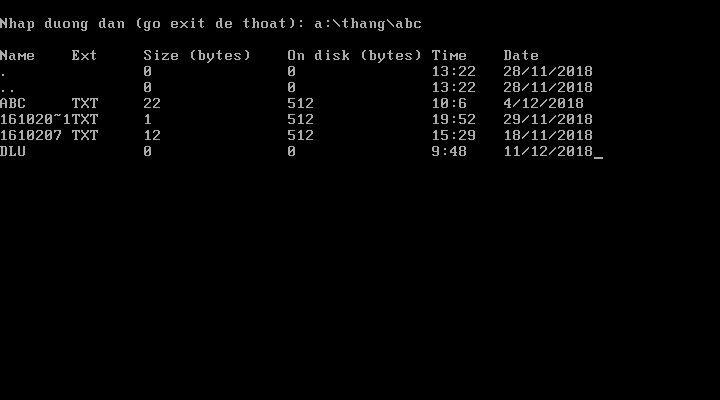


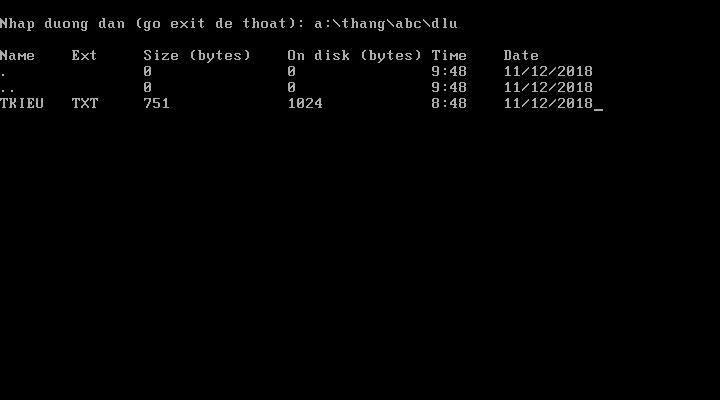
* So sánh với properties trên Windows XP:

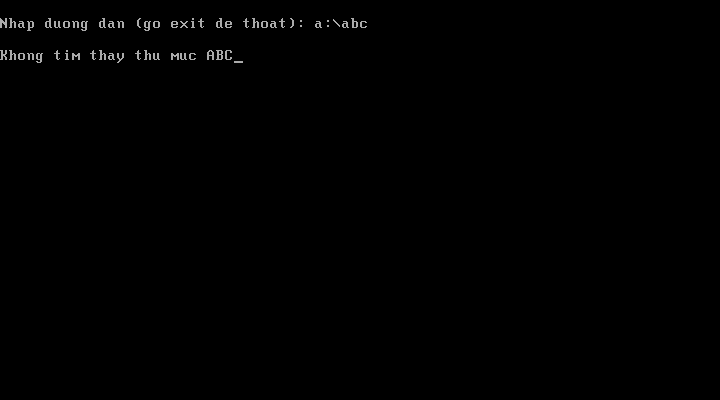




* Một số kết quả khác:







# Đánh giá

## Thuận lợi

Việc cài đặt chương trình không gặp quá nhiều khó khăn bởi vì các hàm trên hầu hết đã được nghiên cứu từ trước, bản thân người viết cũng có chỉnh sửa nhằm khắc phục các cảnh báo khi biên dịch chương trình bằng Borland C++ và để tối ưu cho chức năng chính.

## Hạn chế

Chương trình này hiện tại chỉ xuất được kích thước của các tập tin, chưa xuất được kích thước của thư mục.

## Phương hướng phát triển

Từ cơ sở của việc tính kích thước của các tập tin riêng lẻ, ta có thể vận dụng để tính kích thước của thư mục bằng đệ quy:

Hàm Lấy\_thông\_tin\_của\_thư\_mục (PointerType listEntry):

Biến size = 0, size\_on\_disk = 0;

Với mỗi entry trong listEntry:

Nếu entry này là của tập tin:

size += size của entry;

size\_on\_disk += size\_on\_disk của entry;

Ngược lại, nếu entry này là của thư mục:

Đệ quy hàm Lấy\_thông\_tin\_của\_thư\_mục (GetEntryDir (GetCluster (cluster bắt đầu của entry))) và cộng dồn vào 2 biến size và size\_on\_disk;

Đi tới entry tiếp theo trong danh sách;

Trả về 2 biến size và size\_on\_disk;

# Tài liệu tham khảo

[1] Giáo trình Hệ điều hành, Đặng Thanh Hải, Khoa Công nghệ thông tin, Trường Đại học Đà Lạt

[2] Ổ đĩa cứng, <https://en.wikipedia.org/wiki/Hard_disk_drive>

[3] MS-DOS, <https://vi.wikipedia.org/wiki/MS-DOS>

[4] fat-structs, <https://web.stanford.edu/class/cs140e/notes/lec7/fat-structs.pdf>

[5] Why is There a Big Difference Between ‘Size’ and ‘Size on Disk’?, <https://www.howtogeek.com/180369/why-is-there-a-big-difference-between-size-and-size-on-disk/>