BÀI TẬP VỀ NHÀ
HỆ ĐIỀU HÀNH

Phạm Văn Thông

1 Cấu trúc "hình học của đĩa"

Đĩa cứng gồm vài đĩa có phủ lớp từ tính, nằm trên cùng một trục và quay với vận tốc lớn. Đọc/Ghi dữ liệu ra đĩa được thực hiện bởi các đầu đọc nằm giữa các đĩa này, di chuyển từ tâm đĩa ra ngoài của đĩa. Vòng tròn đầu đọc vẽ ra trên các đĩa khi quay quanh chúng gọi là *rãnh* (track), còn tập hợp các rãnh nằm chồng lên nhau gọi là cylinder. Mỗi rãnh lại chia thành các sector, và có thể ghi vào mỗi sector 512 byte thông tin. Vì thế đặc điểm của một ổ đĩa thường là tập hợp ba *số cylinder-số rãnh trong cylinder-số sector trên rãnh trên cylinder*, còn được viết tắt là C-H-S. Ba số này gọi là cấu trúc "*Cấu trúc hình học của đĩa*". Đĩa với cấu trúc hình học C-H-S có dung lượng C*H*S*512 byte.

Đĩa cứng là các thiết bị khối, tức là đọc và ghi thông tin theo các khối, và kích thước nhỏ nhất của khối bằng một sector (512 byte). Để có thể ghi thông tin lên đĩa, cần đặt đầu đĩa đúng vị trí, tức là chỉ cho controller biết cần ghi thông tin này vào sector nào. Sector được địa chỉ theo số thứ tự cylinder, số thứ tự đầu đọc và số thứ tự sector trên rãnh (*địa chỉ 1 chiều của sector*).

2 Phân vùng và bảng phân vùng của đĩa

Trong các hệ thống Intel ổ đĩa thường được chia thành các phân vùng. Rất có thể nguyên nhân của việc phân vùng là nguyên nhân lịch sử: các phiên bản MS-DOS đầu tiên không thể sử dụng được các đĩa lớn, mà dung lượng đĩa lại phát triển nhanh hơn khả năng của DOS. Khi đó đã nghĩ đến việc chia ổ đĩa thành các phân vùng.Để làm được điều này, trong sector số 0 của đĩa (sector đầu tiên trong cylinder số 0) ghi nhớ *bảng chia ổ đĩa thành các phân vùng* (*partition table*). Mỗi phân vùng được dùng như một đĩa vật lý riêng rẽ. Một trường hợp nói riêng đó là trong các phân vùng khác nhau có thể cài đặt các hệ điều hành khác nhau.

Bảng phân vùng chứa 4 bản ghi 16byte cho 4 phân vùng chính. Mỗi bản ghi có cấu trúc như sau:

```
struct partition {
char active;//0x80: phân vùng kích hoạt, 0: không kích hoạt
```

```
char begin[3]; //CHS đầu tiên
char type; //Loại phân vùng (VD: 83 -- LINUX_NATIVE)
char end[3]; //CHS cuối cùng
int start;  //số của sector đầu tiền (32bit, tính từ 0)
int length; //Số sector trong phân vùng (32bit)
};
```

DOS sử dụng trường begin và end của bảng phân vùng và *interupt 13h* của BIOS để truy cập tới đĩa, vì thế không thể sử dụng đĩa có dung lượng lớn hơn 8.4GB ngay cả các BIOS mới, còn phân vùng thì không thể lớn hơn 2.1GB (nhưng đây là hạn chế của hệ thống tập tin FAT16).

Linux thì chỉ sử dugnj trường *start* và *length* của bảng phân vùng đĩa và hỗ trợ các phân vùng chứa đến 232 sector, tức là dung lượng có thể đạt 2TB.

Vì trong bảng chia ổ đĩa chỉ có 4 dòng cho các phân vùng, số phân vùng chính trên đĩa ngay từ đầu đã hạn chế: không thể lớn hơn 4. Khi mà 4 phân vùng trở thành ít, thì người ta sáng chế ra phân vùng *logic*. Một trong số các phân vùng chính trở thành *mở rộng* (loại phân vùng -5 hay F hay 85 trong hệ cơ số mười sáu). Và trong phân vùng mở rộng người ta tạo ra các *phân vùng logic*. Phân vùng mở rộng không được sử dụng trực tiếp mà chỉ được dùng để tạo ra các *phân vùng logic*.

Sector đầu tiên của phân vùng mở rộng ghi nhớ bảng phân vùng với bốn đầu vào: một dùng cho phân vùng logic, một cho phân vùng mở rộng khác còn lại 2 cái không được sử dụng. Mỗi phân vùng mở rộng có một bảng chia của mình, trong bảng này cũng giống như trong phân vùng mở rộng chính, chỉ sử dụng có hai dòng để đưa ra một chuỗi các mắt xích tử bảng phân vùng, mắt xích đầu tiên mô tả ba phân vùng chính, và mỗi mắt xích tiếp theo - một phân vùng logic và vị trí của bảng tiếp theo. Số phân vùng logic theo nguyên tắc là không hạn chế, vì mỗi phân vùng logic có thể chứa bảng phân vùng và các phân vùng logic của mình. Tuy nhiên trên thức tế vẫn có những hạn chế. Ví dụ, Linux không thể làm việc với hơn 15 phân vùng trên các đĩa SCSI và hơn 63 phân vùng trên đĩa IDE.

Phân vùng mở rộng trên một đĩa vâ vật lý hay một phân vùng mở rộng chứa nó chỉ có thể là một. Không một chương trình phân chia ổ đĩa nào có thể tạo thêm một phân vùng mở rộng thứ hai.

Ổ đĩa trên Linux nói riêng (ổ đĩa vật lý) được truy cập qua tên của thiết bị, ví dụ: */dev/hda, /dev/hdb, /dev/sda,...* Các phân vùng chính có thêm số 1-4 trong tên thiết bị */dev/hda1, /dev/hda2, /dev/sda3, ...* còn phân vùng logic thì có các tên */dev/hda5, /dev/hda6,....* Từ những gì đề cập ở trên có thể suy ra tai sao có thể bỏ qua các tên như */dev/hda3* hay */dev/hda4* (đơn giản là phân vùng chính thứ ba và thứ tư không được tạo ra) và ngay sau */dev/hda2* là */dev/hda5*, và sau đó việc đánh số lại theo thứ tự thông thường.

Trong windows các phân vùng logic được nhận tên chữ cái, bắt đầu từ chữ cái cuối cùng dành cho phân vùng chính. Ví dụ nếu một đĩa cứng có hai phân vùng chính (C: và D:) và một phân vùng mở rộng, trong phân vùng mở rộng tạo ra hai phân vùng lôgíc, thì những phân vùng lôgíc này sẽ được đặt tên E: và F:. Xin nói thêm, trong Windows NT và 2000/XP có thể thay đổi tên của các phân vùng đĩa.

3 Chương trình đọc bảng phân vùng của đĩa

3.1 Sử dụng

Chương trình sau đây thực hiện đọc các thông tin trong bảng phân vùng của ổ đĩa. Chương trình được viết cho hệ điều hành GNU/Linux. Nếu muốn đọc thông tin bảng phân vùng của ổ đĩa vật lý hoặc phân vùng mở rộng bất kì, bạn phải nhập vào biến môi trường tương ứng (ví dụ: */dev/hda, /dev/sdb, /dev/hda5,...*) Chương trình đòi hỏi quyền root để có thể thực thi. Câu lệnh bạn nhập vào sẽ có dạng:

```
sudo ./partition /dev/sdx
```

hoặc

```
sudo ./partition
```

Trong trường hợp thứ 2, chương trình tự hiểu rằng bạn đang yêu cầu xem thông tin của ổ đĩa */dev/sda*.

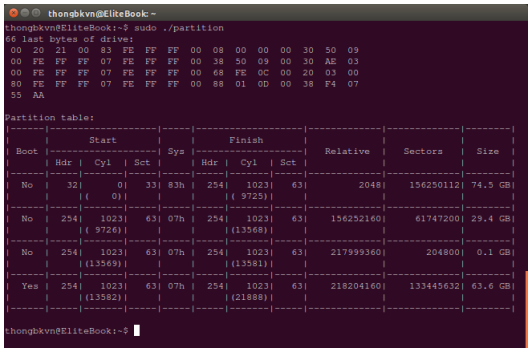
3.2 Source code

Đây là phần source code của chương trình. Xin lưu ý lại rằng chương trình được viết cho hệ điều hành GNU/Linux vì vậy bạn không thể cố gắng biên dịch nó trên Windows.

```
#include "stdio.h"
#include "stdlib.h"
#include "string.h"

void convert (unsigned char a, unsigned char b)
{
unsigned int c;
c=(b<<8)+a;
printf("%7d|%5d|",(c&65472)>>6,c&63); //c&1111111111000000b and c&111111b
}

int main(int argv, char **argc)
{
FILE *fp,*fl;
char path[30];
unsigned char buf[512];
void *data=malloc(512);
unsigned int i,j,*p;
if (argc[1])
strcpy(path,argc[1]);
```

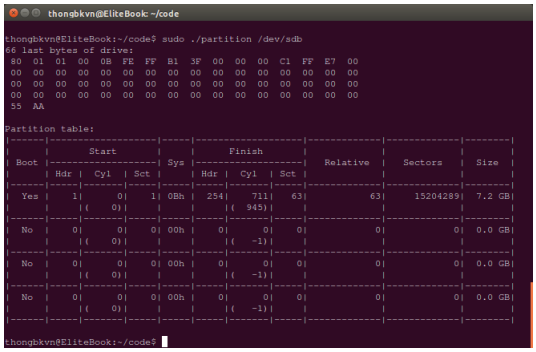
```
else
strcpy(path,"/dev/sda");
if (!(fp=fopen(path,"rb")))
printf("Error to open");
fread(buf,1,512,fp);
printf("66 last bytes of drive:\n");
for (i=446;i<512;i++)
{
printf(" %02X ",buf[i]);
if (i%16==13) printf("\n");
}
i=446;
printf("\n\nPartition table:\n");
printf("|------|-------------------|-----|-------------------|-------------|-------------|--------|\n");
printf("|      |       Start       |     |      Finish       |             |             |        |\n");
printf("| Boot |-------------------| Sys |-------------------|   Relative  |   Sectors   |  Size  |\n");
printf("|      | Hdr |  Cyl  | Sct |     | Hdr |  Cyl  | Sct |             |             |        |\n");
printf("|------|-----|-------|-----|-----|-----|-------|-----|-------------|-------------|--------|\n");
while (i<510)
{
if (buf[i]==0x80)            //Contain OS
printf("|  Yes |");
else
printf("|  No  |");
printf("%5d|",buf[i+1]);       //header
convert(buf[i+2],buf[i+3]);    //Cylinder and sector
printf(" %02Xh |",buf[i+4]);    //Disk format
printf("%5d|",buf[i+5]);       //header
convert(buf[i+6],buf[i+7]);    //Cylinder and sector
p=(unsigned int*)(buf+i+8);
printf("%13u|%13u|",*p,*(p+1)); //Relative and Sectors
printf("%5.1f GB|",*(p+1)*1.0/2/1024/1024); //size
printf("\n|      |     |(%5d)|     |     |     |(%5d)|     |             |             |        |"
,*p/255/63,(*p+*(p+1))/255/63-1);

i+=16;
printf("\n|------|-----|-------|-----|-----|-----|-------|-----|-------------|-------------|--------|\n");
}
printf("\n");
fcloseall();
return 0;
}
```

Sau đây là kết quả thu được sau khi chạy chương trình với lệnh sudo ./partition (giả sử chương trình được biên dịch vơi tên *partition*).



Hình 1: Bảng phân vùng ổ đĩa cứng /dev/hda

Và đây là bảng phân vùng của thiết bị nhớ flash được cắm qua cổng USB. Để xem bảng phân vùng, ta chạy với lệnh sudo ./partition /dev/sdb :



Hình 2: Bảng phân vùng /dev/sdb