Đại học quốc gia thành phố Hồ Chí Minh

Trường Đại học Khoa Học Tự Nhiên

Hệ điều hành

Hệ thống tập tin FAT cho hệ điều hành nachos

Người thực hiện:

Nguyễn Văn Phước 1612523

Giáo viên:

Phạm Tuấn Sơn

Mục Lục

I. Giới thiệu: 3
II. Xây dựng: 3
1. Cấu trúc BootSector: 3
2. Cấu trúc bảng FAT: 4
3. Cấu trúc bảng thư mục: 4
III. Cài đặt: 6
1. Lớp BootSector: 6
2. Lớp FATTable: 6
a. Xin cấp phần tử FAT khi tạo file: 6
b. Xóa tập tin: 6
3. Lớp directory: 6
a. Thêm một tập tin mới vào bảng thư mục: 7
b. Xóa một tập tin: 7
4. Lớp OpenFile: 7
a. Đọc dữ liệu từ tập tin: 7
b. Ghi dữ liệu xuống file: 9
c. Dịch con trỏ đọc ghi file: 9
IV. Kiểm thử: 9
1. Thêm vào nachos: 9
2. Tạo mới một file: 10
3. Hiển thị danh sách tập tin: 11
4. Xóa một tập tin: 11
5. Sao chép tập tin: 12
6. Đọc nội dung file: 13

I. Giới thiệu:

Với đồ án này có mục tiêu là xây dựng và triển khai hệ thống tập tin FAT trên hệ điều hành nachos. Ở hệ điều hành nachos hiện tại vẫn có hệ thống tập tin riêng nhưng dựa vào cấu trúc tập tin của linux để xây dựng.

II. Xây dựng:

1. Cấu trúc BootSector:

- Ta xây dựng một lớp BootInfo để lưu trữ thông tin của bootsector như sau:

```
class BootInfo{

public:
        char bootStep[3]; // nhảy đến đầu đoạn mã boot
        char nameCompany[8]; // tên hệ điều hành

        short byteOfSector;// số byte của một sector
        char sectorOfCluster; // số sector của một cluster
        short numSectorPreFat; // số sector trước bảng FAT
        char numFat; // số lượng bảng FAT, thường là 2 nhưng ở đây chỉ dùng 1
        short numEntry; // số entry của RDET
        short numSectorVolumeL; // số sector của ổ đĩa, lưu giá trị khi số sector
                                //nhỏ hơn 2^16 - 1 = 65536

        char symVolume; // ký hiệu loại volume

        short numSectorFAT; // số sector của bảng FAT

        short numSectorTrack; //số sector của track
        short numReadHead; //số lượng đầu đọc
        int numBytes; // khoảng cách từ nơi mô tả vol đến đầu vol

        int numSectorVolumeH;// số sector của ổ đĩa, lưu giá trí lớn

        char type;// lưu loại ổ đĩa: mềm 0, cứng 80h
        char nu;// ký hiệu dành riêng
        char signOS; // dấu hiện nhận diện hệ điều hành
        int serialNumber;// serial Number của volume
        char volumeLabel[11]; // Volume Label
        char typeFAT[8]; //loại FAT
        char pushOS[SectorSize - 70]; // đoạn mã nạp hệ điều hành
        short signEndBootInfo ; // mã kết thức AA55h
};
```

- Ta sẽ vẫn tuân theo cấu trúc của bootsector của hệ thống tập tin FAT ở đây ta dùng FAT16

2. Cấu trúc bảng FAT:

- Đầu tiên ta sẽ định nghĩa một số thông tin cho bảng FAT như sau:

```
#define NUMFAT 1
#define NUMSECTORFAT 1
#define SECTOROFCLUSTER 2
#define SECTORSTARTFAT 1
```

  - NUMFAT là số lượng bảng FAT thông thường sẽ là 2 nhưng ở đây ta chỉ cần 1 để cho việc xử lý trở nên đơn giản hơn.
  - NUMSECTOFAT là số lượng sector mà bản FAT sẽ chiếm.
  - SECTOROFCLUSTER là số sector của một cluster.
  - SECTORSTARTFAT là vị trí sector bắt đầu của bảng FAT.
- Với thông tin bảng FAT là một dãy các phần từ 2 byte nên ta sẽ dùng một mảng kiểu short để lưu trữ:

```
short fatItems [FATSIZE]
```

  - FATSIZE sẽ được tính thông qua công thức:

```
#define FATSIZE (NUMSECTORFAT*SectorSize)/2
```

  - SectorSize là số byte của một sector đã được định nghĩa sẵn ở trong disk.h

3. Cấu trúc bảng thư mục:

- Đầu tiên ta cần định nghĩa một số thông số mặc định như sau:

```
#define E5 229
#define FILETYPE 0x20
#define DIRSIZE 10
#define ENTRYSIZE 32
#define SECTORSTARTDIR 2
#define MAXLENGHTFILENAME 12
#define MAXLENGHTSHORT 6
#define MAXLENGHTEXTENSION 3
```

  - E5 dấu hiệu đánh dấu xóa file
  - FILETYPE ký hiệu đánh dấu là tập tin.
  - DIRSIZE số lượng tập tin có thể chứa trong bảng thư mục
  - ENTRYSIZE kích thước của một entry
  - SECTORSTARTDIR vị trí sector bắt đầu của bảng thư mục
  - MAXLENGHTFILENAME chiều dài tối đa mà tên của một file có thể có (đã bao gồm phần mở rộng và dấu chấm)

- MAXLENGHTSHORT là số ký tự mà tập tin tên dài được giữ lại ở phần tên.
- MAXLENGHTEXTENSION là số ký tự tối đa mà phần mở rộng của một file có thể có.

- Xây dựng class lưu trữ thông tin của một Entry như sau:

```
class DirectoryEntry {
public:
    char name [MAXLENGHTFILENAME]; // tích hợp cả phần
                                   // mở rộng
    unsigned char type; // trạng thái và loại tập tin
    //unsigned char C;// phần dành riêng vì ko sử dụng nên
                      //chuyển 1 byte lên name để lưu dấu chấm
    char hour[3]; // giờ tạo
    unsigned short date; // ngày tạo
    unsigned short lastDateAccess;// ngày truy cập gần nhất
    unsigned short hStartCluster; // cluster bắt đầu (byte cao)
    unsigned short lastHourAccess; // giờ truy cập gần nhất
    unsigned short lastUpdate; // ngày cập nhật gần nhất
    unsigned short lStartCluster; // cluster bắt đầu (byte thấp)
    unsigned int size; // kích thước tập tin
};
```

- Với bảng này thì số lượng của Entry chỉ bằng 10 và ta sẽ sử dụng mảng 10 phần tử để lưu cái entry này.

III. Cài đặt:

- Cách cài đặt chi tiết đã có code đi kèm, ở báo cáo này chỉ trình bày lại các chức năng chính của các lớp được khai báo và cách xử lý các vấn đề trong việc thao táo với tập tin.

1. Lớp BootSector:

- Nhiệm vụ: quản lý đối tượng BootInfo và có nhiệm vụ format và kiểm tra format ổ đĩa theo định dạng FAT

2. Lớp FATTable:

- Nhiệm vụ: quản lý mảng các phần tử FAT, cung cấp một số phương thức để giao tiếp với bảng FAT như sau:

a. Xin cấp phần tử FAT khi tạo file:

- Khi thêm một file mới thì ta sẽ yêu cầu cấp các phần tử FAT để lưu trữ file đó, ở đây ta sẽ chỉ sử dụng các phần tử trống liên tiếp nhau mà thôi không dùng lại các lỗ trống trong bảng FAT.
- Để xin cấp phần tử FAT ta cần cho biết dung lượng của tập tin là bao nhiêu để tính toán số cluster sẽ cấp cho file đó.
- Trong trường hợp nếu không còn đủ số lượng phần tử FAT để cấp cho tập tin đó thì trả về -1.

b. Xóa tập tin:

- Khi nhận yêu cầu xóa tập tin, thì người dùng sẽ truyền vào vị trí cluster bắt đầu và chương trình sẽ tự động đi đến từng phần tử FAT nối với nhau và chuyển về 0.

3. Lớp directory:

- Nhiệm vụ: quản lý tất cả các entry, cung cấp các phương thức để làm việc với cây thư mục như: tìm file, xóa file, xem danh sách các tập tin, …

a. Thêm một tập tin mới vào bảng thư mục:

- Ta sẽ cung cấp tên file và dung lượng của tập tin.
- Kiểm tra xem tập tin đó đã tồn tại trong cây thư mục hay chưa, nếu đã tồn tại thì báo lỗi.
- Sau đó tìm vị trí trống trong mảng các entry để lưu trữ thông tin của tập tin, nếu trường hợp không còn chỗ trống, thì ta sẽ tìm tiếp xem có entry nào được đánh dấu xóa hay không, nếu cũng không có luôn thì báo lỗi.
- Tiếp theo ta sẽ xin cấp phát các phần tử FAT thông qua đối tượng lớp FATTable, và lấy cluster bắt đầu.
- Lưu thông tin của tập tin vào entry.
- Với tên tập tin quá dài thì ta sẽ sử dụng cách mà window lưu trữ là giữ lại 6 ký tự đầu của tên file và thêm vào chỉ số là số lần xuất hiện của các file có 6 ký đầu của tên giống nhau.

b. Xóa một tập tin:

- Truyền vào tên file cần xóa, sau đó kiểm tra xem file có tồn tại trong cây thư mục hay không, nếu không thì báo lỗi.
- Nếu tìm thấy thì bắt đầu việc xóa bảng FAT, thông qua đối tượng FATTable.
- Sau đó đánh dấu xóa E5 ở tên của tập tin đó. Và giữ lại vùng data.

4. Lớp OpenFile:

- Nhiệm vụ: lưu trữ thông tin và cung cấp các phương thức để làm việc với file như: read, write, seek.

a. Đọc dữ liệu từ tập tin:

- Cung cấp các phương thức Read hoặc ReadAt, với Read sẽ đọc nội dung file theo vị trí con trỏ của file và sẽ được tang sau mỗi lần đọc, ReadAt sẽ đọc file tại vị trí do người dùng truyền vào.
- Bản chất của hàm Read là sử dụng lại hàm ReadAt.
- Vì hàm đọc của SynchDisk của nachos chỉ cho phép đọc ghi trên một sector, nên ta cần tính toán một số thông số về số cluster sẽ đọc, và vị trí dữ liệu mà người dùng cần lấy như sau:
  - Đầu tiên ta cần tìm vị trí cluster bắt đầu việc đọc, ta không cần phải đọc hết toàn bộ file lên, mà chỉ cần đọc các cluster cần đọc mà thôi.
  - Ta tính vị trí cluster bắt đầu bằng công thức:

Pos = position / clusterSize

Dựa vào công thức trên thì ta có thể tìm được khoảng cách từ cluster bắt đầu tới vị trí cluster ta cần đọc là Pos cluster. Vì chưa chắc các cluster của vùng data là nối tiếp nhau nên ta cần nhờ đối tượng FATTable để tìm chính xác vị trí cluster bắt đầu đọc.

- Ta tính tiếp số cluster phải đọc bằng công thức:

  numCluster = divRoundUp (position%clusterSize + numBytes, clusterSize)

  - Với divRoundUp là macro giúp ta làm tròn lên kết quả chia, ví dụ ta có kết quả chia của 9/2 = 4.5 thì ta sẽ nhận được giá trị là 5.
  - Ở đây ta dùng position%clusterSize + numBytes để chia cho clusterSize, vì ta đã tính được vị trí cluster bắt đầu đọc nên ta cần chuyển vị trí đọc về lại vị trí đọc tính từ cluster bắt đầu và ở đây ta sẽ cộng thêm cho số byte cần đọc để xác định chính xác số cluster ta cần đọc. Ví dụ: vị trí ta cần đọc là 10, 1 cluster thì có 2byte và ta cần đọc 10byte thì ta sẽ có vị trí cluster ta sẽ đọc sẽ bắt đầu đọc từ cluster thứ 5 của file và số cluster ta cần đọc tính từ vị trí cluster bắt đầu là 5.
- Sau đó ta cấp phát vùng nhớ buf có kích thước đúng bằng kích thước của các cluster mà ta sẽ đọc.
- Ta tiến hành đọc từng cluster một, ta sẽ duyệt qua các sector của của các cluster để đọc hết dữ liệu của sector đó lên, làm cho tới khi đã đọc hết tất cả các cluster.
- Và copy vùng dữ liệu mà người dùng cần đọc vào data để chuyển về cho người dùng.

b. Ghi dữ liệu xuống file:

- Cũng giống với đọc file thì lớp OpenFile cũng cung cấp các phương thức để ghi file là Write và WriteAt.
- Các bước thực hiện đều giống với đọc file, chỉ khác ta cần phải kiểm tra xem vị trí ghi file hiện tại là có phải đầu cluster hay không, nếu không phải đầu cluster, thì ra cần phải đọc dữ liệu của cluster lên, vì cũng giống như Read thì hàm Write của SynchDisk chỉ hỗ trợ ghi xuống 1 sector, nên nếu ta không được dữ liệu trước đó thì khi ta ghi sẽ mất vùng dữ liệu đó.

c. Dịch con trỏ đọc ghi file:

- Dịch chuyển vị trí đọc ghi file khi sử dụng hàm Read, Write. Vị trí con trỏ sẽ được tự động tăng lên sau mỗi lần đọc ghi, ta sẽ sử dụng hàm Seek để đưa về vị trí ta mong muốn.

IV. Kiểm thử:

1. Thêm vào nachos:

- Ta sẽ copy các file ta định nghĩa lại vào thư mục filesys của nachos, ta bao gồm các file cặp file sau:
    - filesys.h / filesys.cc
    - bootsector.h / bootsector.cc
    - FATTable.h / FATTable.cc
    - directory.h / directory.cc
    - openfile.h / openfile.cc
- Ta sẽ ghi đè lên các file có sẵn của nachos, và sửa lại trong Makefile.common như sau:

```
FILESYS_H =../filesys/bootsector.h \
       ../filesys/directory.h \
       ../filesys/FATTable.h\
       ../filesys/filesys.h \
       ../filesys/openfile.h\
       ../filesys/synchdisk.h\
       ../machine/disk.h
FILESYS_C =../filesys/bootsector.cc \
       ../filesys/directory.cc\
       ../filesys/FATTable.cc\
       ../filesys/filesys.cc\
       ../filesys/fstest.cc\
       ../filesys/openfile.cc\
       ../filesys/synchdisk.cc\
       ../machine/disk.cc
```

```
FILESYS_O =bootsector.o directory.o FATTable.o filesys.o fstes
t.o openfile.o synchdisk.o\
      disk.o
```

- Và ta cũng chỉnh trong Makefile trong thư mục userprog để chuyển nachos từ sử dụng hệ thống tập tin của linux sang sử dụng hệ thống tập tin của ta định nghĩa bằng cách mở đoạn code sau và nhớ đóng lại đoạn code cũ:

```
# if file sys done first!
DEFINES = -DUSER_PROGRAM -DFILESYS_NEEDED -DFILESYS
INCPATH = -I../bin -I../filesys -I../userprog -I../threads -I../machine
HFILES = $(THREAD_H) $(USERPROG_H) $(FILESYS_H)
CFILES = $(THREAD_C) $(USERPROG_C) $(FILESYS_C)
C_OFILES = $(THREAD_O) $(USERPROG_O) $(FILESYS_O)
```

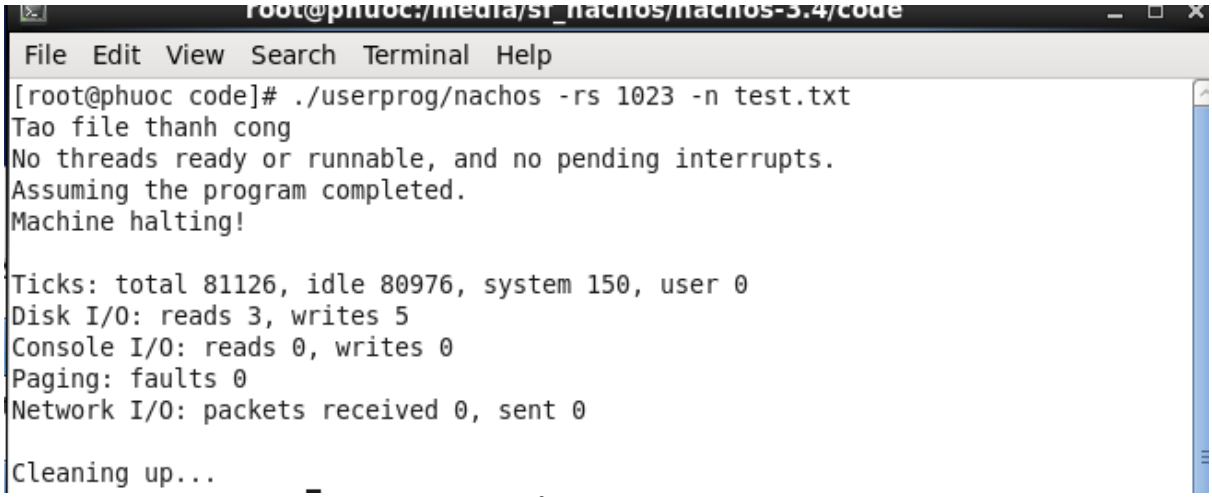
- Như vậy là ta đã hoàn thành việc thêm hệ thống tập tin FAT vào nachos.

2. Tạo mới một file:

- Ở đây có định nghĩa một lệnh mới dùng để tạo file là -n <Tên file>
- Ta gõ lệnh như sau:

  ./userprog/nachos -rs 1023 -n test.txt

- Lúc này sẽ có thông báo nếu tạo tập tin thành công sẽ có thông báo Tao tap tin thanh cong, ngược lại thì Tao tap tin that bai.



- Tình trạng cây thư mục dưới ổ đĩa như sau:

```
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 74 65 73 74  2E 74 78 74 00 00 00 00  ....test.txt....
20 00 00 00 00 00 00 00  00 00 00 00 00 00 02 00   ...............
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

- Tình trạng bảng FAT:

```
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 55 AA FE FF FE FF  FF FF 00 00 00 00 00 00  ..U.............
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

3. Hiển thị danh sách tập tin:

- Ta có hai lựa chọn hiển thị danh sách tập tin thư mục là:
- Chỉ hiển thị tên:

./userprog/nachos -rs 1023 -l

Sẽ có kết quả như sau:

```
[root@phuoc code]# ./userprog/nachos -rs 1023 -l
test.txt
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
```

- Hai là hiển thị tên tập tin và kích thước của tập tin đó:

./userprog/nachos -rs 1023 -D

Ta có kết quả như sau:

```
[root@phuoc code]# ./userprog/nachos -rs 1023 -D
name            size
test.txt                0
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
```

4. Xóa một tập tin:

./userprog/nachos -rs 1023 -r <Tên file>

- Ta thử xóa tập tin ta vừa mới tạo:

```
[root@phuoc code]# ./userprog/nachos -rs 1023 -r test.txt
Xoa file thanh cong
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
```

- Tình trạng bảng thư mục:

```
00 00 00 00 E5 65 73 74  2E 74 78 74 00 00 00 00  .....est.txt....
20 00 00 00 00 00 00 00  00 00 00 00 00 00 02 00   ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

Lúc này thì tên file đã được đánh dấu xóa bằng ký tự E5

- Tình trạng bảng FAT:

```
00 00 55 AA FE FF FE FF  00 00 00 00 00 00 00 00  ..U.............
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

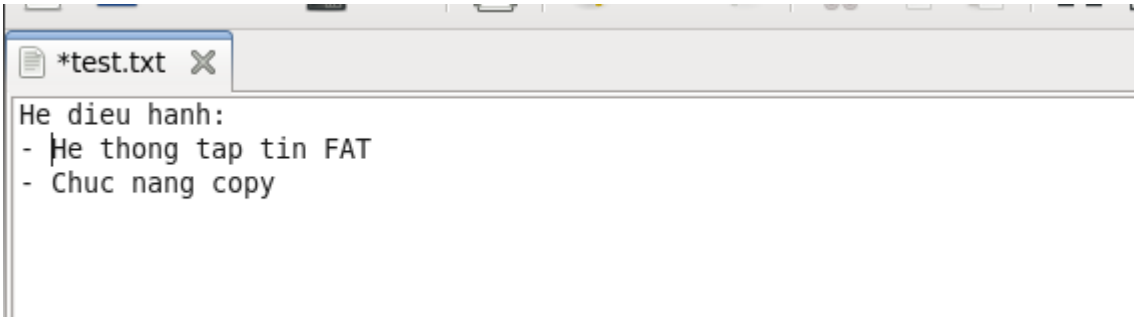
Các phần tử cấp phát đã bị xóa

5. Sao chép tập tin:

- Cho phép ta copy tập từ linux vào hệ thống tập tin nachos

  ./userprog/nachos -rs 1023 -cp <Tên tập ở linux> <Tên mới>

- Ta sẽ tạo thử một tập tin có nội dung như sau:



- Ta sẽ copy tập tin này vào nacho với tên mới là chucnangcopy.txt

  ./userprog/nachos -rs 1023 -cp test.txt chucnangcopy.txt

  Ta có kết quả như sau:

```
[root@phuoc code]# ./userprog/nachos -rs 1023 -cp test.txt chucnangcopy.txt
Sao chep thanh cong
No threads ready or runnable, and no pending interrupts.
```

- Dùng lệnh để xem bảng thư mục ta có kết quả:

```
[root@phuoc code]# ./userprog/nachos -rs 1023 -D
name            size
chucna~0.txt            57
No threads ready or runnable, and no pending interrupts.
Assuming the program completed
```

  Ở đây ta có tên file là tên dài nên sẽ được đổi tên lại có dạng như hình.

- Tình trạng bảng thư mục:

```
00 00 00 00 E5 65 73 74  2E 74 78 74 00 00 00 00  .....est.txt....
20 00 00 00 00 00 00 00  00 00 00 00 00 00 02 00   ................
00 00 00 00 63 68 75 63  6E 61 7E 30 2E 74 78 74  ....chucna~0.txt
20 00 00 00 00 00 00 00  00 00 00 00 00 00 02 00   ................
39 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  9...............
```

  Vẫn giữ lại entry của tập tin đã xóa trước đó, vì bảng thư mục vẫn còn chỗ trống.

- Tình trạng bảng FAT:

```
00 00 55 AA FE FF FE FF  FF FF 00 00 00 00 00 00  ..U.............
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

  Vì file chỉ có dụng lượn 57byte vẫn nằm trong 1 cluster, nên chỉ cấp một cluster để lưu trữ.

- Vùng data lưu trữ nội dung file:

```
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 48 65 20 64  69 65 75 20 68 61 6E 68  ....He dieu hanh
3A 0D 0A 2D 20 48 65 20  74 68 6F 6E 67 20 74 61  :..- He thong ta
70 20 74 69 6E 20 46 41  54 0D 0A 2D 20 43 68 75  p tin FAT..- Chu
63 20 6E 61 6E 67 20 63  6F 70 79 0D 0A 00 00 00  c nang copy.....
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

6. Đọc nội dung file:

./userprog/nachos -rs 1023 -p <Tên file>

- Ta sẽ đọc thử tập tin ta vừa copy sang:

./userprog.nachos -rs 1023 -p chucna~0.txt

- Ta nhận được kết quả như sau:

```
Cleaning up...
[root@phuoc code]# ./userprog/nachos -rs 1023 -p chucna~0.txt
Noi dung file chucna~0.txt la:
He dieu hanh:
- He thong tap tin FAT
- Chuc nang copy
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
```