BÁO CÁO HỆ ĐIỀU HÀNH
PROJECT 02
Tìm hiểu system calls - các thao tác với files

Sinh viên thực hiện: Nguyễn Đức Việt

MSSV: 20120401

Thông tin thành viên

MSSV	Họ và tên	Email
20120401	Nguyễn Đức Việt	20120401@student.hcmus.edu.vn

Đôi lời:

Vì trong khoảng thời gian làm đồ án 1 em bị ốm và nhập viện (gần 2 tuần), em không thể tìm thấy nhóm để làm bài cùng. Do đó, đến khi thực hiện đồ án 2, khi nhóm các bạn đã vào form, em vẫn tiếp tục không thể tìm thấy nhóm cho mình. Vì vậy em đã thực hiện đồ án này cá nhân, mong các thầy cô thông cảm.

1. **Cài đặt các lớp quản lý file**
    - **FileSystem**. Địa chỉ: *filesys/filesys.h*. Lớp này để quản lý các hệ thống file. Trong lớp này có các phương thức hỗ trợ việc quản lý bao gồm: tạo, xóa, mở 1 file dựa vào tên file đó.
    - **OpenFile**. Địa chỉ:*filesys/openfile.h*. Lớp này dùng để thực thi các phép toán (operator) trên 1 file, bao gồm đọc, ghi, lấy độ dài Phương thức Close 1 file chính là destructor.

    Trong class FileSystem ta đặt mảng openf dùng để quản lý 20 phần tử OpenFIle. Mỗi giá trị của OpenFileID chính là chỉ mục (index) của biến đó trong mảng ta tạo ra.

    Trong đó:

    - openf[0]: stdin
    - openf[1]: stdout
    - openf[2]: read only
    - openf[3]: read and write
2. **Cài đặt các syscall**
    2.1. **Cài đặt syscall SC_CreateFile: int CreateFile(char *fileName)**
    - Create system call sẽ sử dụng Nachos **FileSystem Object** để tạo một file rỗng . System call này trả về 0 nếu thành công và -1 nếu có lỗi.
    - Ta đọc địa chỉ của tham số fileName từ thanh ghi r4, sau đó thực hiện chép giá trị ở r4 từ vùng nhớ User sang System bằng hàm **User2System()**. Giá trị chép được là tên file. Ta tiếp tục kiểm tra nếu tên file là **NULL** hoặc không tạo được file thì trả về -1. Nếu tạo file thành công thì trả về 0, ngược lại thì trả về -1 vào thanh ghi r2.

    2.2. **Cài đặt syscall SC_OpenFile: OpenFileID Open(char *name, int type)**
    - Chương trình người dùng có thể mở 2 kiểu file, file chỉ có thể đọc (*read only*), và file có thể đọc và viết (*read and write*). Mỗi tiến trình sẽ được cấp một descriptor table với kích thước cố định. Kích thước của descriptor table có thể lưu được đặc tả của 20 files. 2 phần tử đầu tiên 0 và 1 sẽ không được sử dụng, cho mục đích console input và console output tương ứng.
    - System call sẽ chịu trách nhiệm chuyển đổi buffer của user space khi cần thiết và cấp pháp tương ứng dưới kernel. Sử dụng file system objects được cung cấp trong filesys folder.
    - Hàm system call "**Open**" sẽ trả về file descriptor **id** (**OpenFileID** là một số nguyên integer), hoặc -1 nếu lỗi.
    - Mở file có thể bị lỗi như trường hợp là không tồn tại tên file hay không đủ ô nhớ trong descriptor table. Tham số **type** = 0: chạy **stdin**, **type** = 1: chạy **stdout**, type = 2: chỉ đọc, **type** = 3: mở file đọc và ghi. Nếu tham số truyền bị sai thì system call phải báo lỗi.
    - Ta đọc địa chỉ của tham số name từ thanh ghi **r4** và tham số type từ thanh ghi **r5** sau đó kiểm tra type có hợp lệ hay không (0 <= type <= 3), kiểm tra **index** của file nó nằm trong descriptor table hay không. Nếu hai điều kiện trên hợp lệ thì thì thực hiện chép giá trị ở **r4** từ phía User sang System bằng hàm **User2System()**. Giá trị chép được là tên file. Ta tiếp tục kiểm tra tên file, nếu tên file là **stdin** và **stdout** thì trả về thanh ghi r2 0 và 1 tương ứng. Nếu file được mở thành công, giá trị trả về sẽ là index của file.

    2.3. **Cài đặt syscall SC_CloseFile: Close(OpenFileID id)**
    - Hàm system call "**Close**" sẽ truyền vào **OpenFileID** và sẽ trả về -1 nếu lỗi và 0 nếu thành công.

- Đọc tham số **m_index** của file từ thanh ghi **r4**, sau đó kiểm tra file đó có nằm trong descriptor table không. Nếu có thì tiến hành đóng file, nếu không thì trả về -1.

2.4. Cài đặt syscall SC_ReadFile và SC_WriteFile: int Read(char *buffer, int size, OpenFileID id) và Write(char *buffer, int size, OpenFileID id)

- Các system call đọc và ghi vào file với **id** cho trước. Cần phải chuyển vùng nhớ giữa user space và system space, và cần phải phân biệt giữa Console IO (**OpenFileID** 0, 1) và **File**.
- Trong trường hợp console là read và write console, sử dụng **SynchConsoleIn** class và **SynchConsoleOut** class cho việc read và write, đảm bảo việc trả đúng dữ liệu cho người dùng. Read và write vào console sẽ trả đúng số lượng ký tự được read hoặc write, không phải là charcount. Trong trường hợp read hoặc write bị lỗi, trả về -1.
- Đối với **SC_Read**:
    - Ta đọc địa chỉ của tham số **buffAddr** từ thanh ghi **r4**, tham số **length** từ thanh ghi **r5** và **m_index** của file từ thanh ghi **r6**, sau đó ta tiến hành kiểm tra **m_index** của file truyền vào có nằm ngoài descriptor table không, file cần đọc có tồn tại không và file cần đọc có phải là **stdout** với **type** = 1 không. Nếu vi phạm các điều kiện trên thì trả về -1 cho thanh ghi **r2** ngược lại là hợp lệ thì lấy vị trí con trỏ ban đầu trong file là **old_pos** bằng phương thức **GetCurrentPos()** của lớp **FileSystem** và thực hiện chép giá trị ở r4 từ phía User sang System bằng hàm **User2System()**. Xét trường hợp đọc file bình thường, thì ta lấy vị trí con trỏ hiện tại trong file bằng phương thức **GetCurrentPos()** của lớp **FileSystem** gọi là **new_pos**, trả về số byte thực sự đọc được cho thanh ghi **r2** bằng công thức: **new_pos – old_pos + 1** và cũng chép **buffAddr** từ phía System sang User bằng hàm **System2User()**.
- Đối với **SC_Write**:
    - Ta đọc địa chỉ của tham số **buffAddr** từ thanh ghi **r4**, tham số **length** từ thanh ghi **r5** và **m_index** của file từ thanh ghi **r6**, sau đó ta tiến hành kiểm tra **m_index** của file truyền vào có nằm ngoài descriptor table không, file cần ghi có tồn tại không và file cần ghi có phải là **stdin** với **type** = 0 hay là file chỉ đọc với **type** = 1. Nếu vi phạm các điều kiện trên thì trả về -1 cho thanh ghi **r2** ngược lại là hợp lệ thì lấy vị trí con trỏ ban đầu trong file bằng phương thức **GetCurrentPos()** của lớp **FileSystem** gọi là **old_pos** và thực hiện chép giá trị ở **r4** từ phía User sang System bằng hàm **User2System()**. Giá trị chép được là **buffAddr** chứa chuỗi kí tự. Xét trường hợp ghi file đọc và ghi với **type** = 3, thì ta lấy vị trí con trỏ hiện tại trong file bằng phương thức **GetCurrentPos()** của lớp **FileSystem** gọi là **new_pos**, trả về số byte thực sự ghi được cho thanh ghi **r2** bằng công thức**: new_pos – old_pos + 1**. Xét trường hợp ghi file **stdout** với **type** = 3, ta gọi phương thức **Write** của lớp **SynchConsoleOut** để ghi từng kí tự trong **buffer** và kết thúc là kí tự xuống dòng *'\n'*, trả về số byte thực sự ghi được cho thanh ghi **r2**.

2.5. Cài đặt syscall SC_Seek: int Seek(int position, OpenFileID id)

- Seek sẽ phải chuyển con trỏ tới vị trí thích hợp, position lưu vị trí cần chuyển tới, nếu **pos** = -1 thì di chuyển đến cuối file.
- Ta đọc tham số **pos** từ thanh ghi **r4** và **m_index** của file từ thanh ghi **r5**, sau đó ta tiến hành kiểm tra **m_index** của file truyền vào có nằm ngoài descriptor table không, file cần di chuyển con trỏ có tồn tại không và kiểm tra người dùng có gọi **Seek** trên console không. Nếu vi phạm các điều kiện trên thì trả về -1 cho thanh ghi **r2** ngược lại là hợp

lệ thì kiểm tra nếu **pos** = -1 thì gán pos bằng độ dài của file bằng phương thức **Length()** của lớp **FileSystem**. Gọi phương thức **Seek** của lớp **FileSystem** với tham số truyền vào là **pos** để dịch chuyển con trỏ đến vị trí mong muốn và trả về vị trí dịch chuyển cho **r2**.

2.6. Cài đặt syscall SC_Remove: int Remove(char *name)

- Remove system call sẽ sử dụng Nachos **FileSystem Object** để xóa file.
- Ta đọc tham số **buffer** từ thanh ghi **r4** sau đó sử dụng **User2System** để chép giá trị ở **r4** từ phía User sang System. Giá trị trả về chính là tên file cần xoá. Sau đó ta kiểm tra file có đang mở hay không. Nếu file đang mở, ta tiến hành đóng file, sau đó xoá file. Nếu xoá file thành công, trả về **r2** giá trị 0, nếu thất bại trả về -1.

3. Hướng dẫn chạy chương trình

Ở thư mục *./build.linux*:

Ví dụ:

- Chương trình createfile: `./nachos -x ../test/createfile hello.txt`
- Chương trình cat: `./nachos -x ../test/cat hello.txt`
- Chương trình copy: `./nachos -x ../test/copy a.txt b.txt`
- Chương trình delete: `./nachos -x ../test/delete hello.c`
- Chương trình concatenate: `./nachos -x ../test/concatenate a.txt b.txt`