TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

THÀNH PHỐ HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

Học phần: HỆ ĐIỀU HÀNH

BÁO CÁO ĐỒ ÁN

Tên đồ án: Syscall For FileSystem
Lớp: 20CLC11
Giảng viên: Phạm Tuấn Sơn

MỤC LỤC

I. Thông tin nhóm 3

II. Nội dung đồ án HĐH NachOS 3

1) Cài đặt xử lý các exceptions: 3

2) Cài đặt IncreasePC(): 4

3) Cài đặt System Call int Create(char* name): 4

4) Cài đặt SC OpenFileID Open(char* name, int type) và int Close(OpenFileID id): 4

a) System Call OpenFileID Open(char* name, int type): 4

b) System Call int Close(OpenFileID id): 5

5) Cài đặt System Call int Read(char* buffer, int charcount, OpenFileID id) và int Write(char* buffer, int charcount, OpenFileID id): 5

a) System Call int Read(char* buffer, int charcount, OpenFileID id): 6

b) System Call int Write(char* buffer, int charcount, OpenFileID id): 7

6) Cài đặt System Call int Seek(int pos, OpenFileID id): 7

7) Cài đặt System Call int Delete(char* name): 8

III. Nguồn tham khảo 8

I. Thông tin nhóm

MSSV	Họ và tên	Email	Chức vụ
20127610	Trương Samuel	20127610@student.hcmus.edu.vn	Nhóm trưởng
20127001	Hà Quốc Anh	20127001@student.hcmus.edu.vn	Thành viên
20127298	Nguyễn Trần Minh Quang	20127298@student.hcmus.edu.vn	Thành viên

II. Nội dung đồ án HĐH NachOS

1) Cài đặt xử lý các exceptions:

Trong thư mục code/machine/machine.h ta có danh sách các exceptions được liệt kê. Hầu hết các exception trong này là run-time errors, khi các exception này xảy ra thì user program không thể được phục hồi.

```
enum ExceptionType { NoException,           // Everything ok!
                     SyscallException,      // A program executed a system call.
                     PageFaultException,    // No valid translation found
                     ReadOnlyException,     // Write attempted to page marked
                                            // "read-only"
                     BusErrorException,     // Translation resulted in an
                                            // invalid physical address
                     AddressErrorException, // Unaligned reference or one that
                                            // was beyond the end of the
                                            // address space
                     OverflowException,     // Integer overflow in add or sub.
                     IllegalInstrException, // Unimplemented or reserved instr.

                     NumExceptionTypes
};
```

Sau đó ta vào file code/userprog/exception.cc để cài đặt các trường hợp lỗi này theo các Exception Type. Mỗi exception chúng em đều có thông báo lỗi cho người dùng và sử dụng lệnh Interupt→Halt() để tắt hệ thống.

Đối với no exception, đây là trường hợp đặc biệt duy nhất, nó sẽ trả quyền điều khiển về cho hệ điều hành.

Cuối cùng, SyscallException sẽ có các trường hợp được xử lý bằng các hàm chúng ta viết cho user system calls, chúng em sẽ thông tin chi tiết hơn ở phần sau.

2) Cài đặt IncreasePC():

Tất cả các system calls (không phải Halt()) sẽ yêu cầu Nachos tăng program counter trước khi thực system call trả kết quả về.

- Bước 1: Đọc địa chỉ của lệnh hiện tại, sử dụng biến counter.
- Bước 2: Ghi giá trị counter vào thanh ghi trước.
- Bước 3: Đọc giá trị của thanh ghi kế tiếp.
- Bước 4: Nạp giá trị biến counter cho thanh ghi hiện tại.
- Bước 5: Ghi địa chỉ của lệnh kế tiếp cho thanh ghi tiếp theo. Vì mỗi thanh ghi có 4 bytes, nên ta cộng giá trị thanh ghi là 4 để có thể qua được lệnh kế tiếp.

3) Cài đặt System Call int Create(char* name):

Create system call sẽ sử dụng NachOS FileSystem Object để tạo một file rỗng. Ban đầu chuỗi nam đang ở trong user space, do đó buffer phải lưu lại giá trị chuỗi name này trong system space. System call Create sẽ trả về -1 nếu không thể tạo file và trả về 0 nếu thành công.

- Bước 1: Đọc địa chỉ của tham số name từ thanh ghi r4
- Bước 2: Sao chép giá trị ở r4 từ user space sang system space bằng hàm User2System().
- Bước 3: Kiểm tra tên file có phải là giá trị NULL hay không. Nếu tên file là NULL thì thông báo lỗi chương trình cho người dùng và trả về -1 vào thanh ghi r2, ngược lại nếu create file thành công thì trả về 0.

4) Cài đặt SC OpenFileID Open(char* name, int type) và int Close(OpenFileID id):

a) System Call OpenFileID Open(char* name, int type):

User program có thể mở 2 loại file, file chỉ đọc và file đọc-ghi. Mỗi tiến trình sẽ được cấp một bảng mô tả file với kích thước cố định. Kích thước của bảng mô tả file là có thể lưu được đặc tả của 10 files. Trong đó, 2 phần tử đầu, ô 0 và ô 1 để dành cho console input và console output.

System Call mở file phải làm nhiệm vụ chuyển đổi địa chỉ buffer trong user space khi cần thiết và viết hàm xử lý phù hợp trong kernel. Dùng đối tượng filesystem trong thư mục filesys. System Call Open sẽ trả về id của file (OpenFileID = một số nguyên), hoặc là -1 nếu bị lỗi.

Mở file có thể bị lỗi như trường hợp là không tồn tại tên file hay không đủ ô nhớ trong bảng mô tả file. Tham số type = 0 cho mở file đọc và ghi, = 1 cho file chỉ đọc. Nếu tham số truyền bị sai thì system call phải báo lỗi. System call sẽ trả về -1 nếu bị lỗi và 0 nếu thành công.

Mục đích cài đặt SC_Open: mở một file với tham số truyền vào gồm tên file và chế độ mở file.

- Bước 1: Đọc tham số thứ 1 (name) từ thanh ghi r4.
- Bước 2: Đọc tham số thứ 2 (type) từ thanh ghi r5.
- Bước 3: Kiểm tra type có hợp lệ hay không (0 <= type <= 2), nếu không hợp lệ thì thông báo cho người dùng và trả về -1 vào thanh ghi r2.
- Bước 4: Tìm ô còn trống (biến int freeSlot) trong bảng đặc tả file. Mỗi tiến trình Read/Write sẽ được cấp một bảng đặc tả file có kích thước là 10 (từ 0 → 9).
- Bước 5: Nếu freeSlot có giá trị khác -1, ta thực hiện kiểm tra fileSystem→table[freeSlot] có khác NULL không, nếu khác thì ghi OpenFileID vào thanh ghi r2, ngược lại trả về -1 vào thanh ghi r2. Nếu freeSlot có giá trị -1 thì cũng trả về -1 vào thanh ghi r2.

b) System Call int Close(OpenFileID id):

Mục đích cài đặt SC_Close: Đóng file với tham số truyền vào là ID của file.

- Bước 1: Đọc địa chỉ tham số file id từ thanh ghi r4.
- Bước 2: Kiểm tra xem file id có tồn tại không (trong đoạn [2, 9]) và file id có nằm ngoài bảng mô tả file hay không. Nếu file id hợp lệ thì delete fileSystem→table[fileID] và gán lại = NULL, sau đó trả về 0 vào thanh ghi r2. Ngược lại nếu file id không hợp lệ sẽ báo lỗi cho người dùng và trả về -1 vào thanh ghi r2.

5) Cài đặt System Call int Read(char* buffer, int charcount, OpenFileID id) và int Write(char* buffer, int charcount, OpenFileID id):

Các system call đọc và ghi vào file với id cho trước. Phải chuyển vùng nhớ giữa user space và system space, và cần phải phân biệt giữa Console IO (OpenFileID 0, 1) và File.

Lệnh Read và Write sẽ làm việc như sau: Phần console read và write, bạn sẽ sử dụng lớp SynchConsole. Được khởi tạo qua biến toàn cục gSynchConsole(bạn phải khai báo biến này). Sử dụng các hàm mặc định của SynchConsole để đọc và ghi, tuy nhiên phải chịu trách nhiệm trả về đúng giá trị cho user. Đọc và ghi với Console sẽ trả về số bytes đọc và ghi thật sự, chứ không phải số bytes được yêu cầu. Trong trường hợp đọc hay ghi vào console bị lỗi thì trả về -1. Nếu đang đọc từ console và chạm tới cuối file thì trả về -2. Đọc và ghi vào console sẽ sử dụng dữ liệu ASCII để cho input và output, (ASCII dùng kết thúc chuỗi là NULL (\0)). Phần đọc, ghi vào file, bạn sẽ sử dụng các lớp được cung cấp trong file system. Sử dụng các hàm mặc định có sẵn của filesystem và thông số trả về cũng phải giống như việc trả về trong synchconsole. Cả read và write trả số kí tự đọc, ghi thật sự. Cả Read và Write trả về -1 nếu bị lỗi và -2 nếu cuối file. Cả Read và Write sử dụng dữ liệu binary.

a) System Call int Read(char* buffer, int charcount, OpenFileID id):

Mục đích: Đọc file với tham số là buffer, số ký tự cho phép charcount và id của file.

- Bước 1: Đọc địa chỉ tham số thứ nhất (buffer) từ thanh ghi r4.
- Bước 2: Đọc địa chỉ tham số thứ 2 (charcount) từ thanh ghi r5.
- Bước 3: Đọc địa chỉ tham số thứ 3 (id) từ thanh ghi r6.
- Bước 4: Kiểm tra id của file truyền vào có nằm trong bảng mô tả file không (trong đoạn [0, 9]), nếu id không hợp lệ thì thông báo lỗi cho user và trả về -1 vào thanh ghi r2.
- Bước 5: Kiểm tra file có tồn tại không, nếu không tồn tại thì thông báo lỗi cho user và trả về -1 vào thanh ghi r2.
- Bước 6: Nếu bước 4 và 5 không xảy ra lỗi (tham số truyền vào hợp lệ) thì copy chuỗi từ vùng nhớ user space sang system space.
- Bước 7: Kiểm tra fileID:
  - Nếu fileID = 0 (console input), ta dùng hàm Read của SynchConsole để đọc từ console output. Nếu đọc đến cuối file thì ta chuyển chuỗi đọc được từ console output sang cho user.
  - Nếu fileID = 1 (console output), ta thông báo lỗi cho user (vì ta không thể đọc được console output) và trả về -1 vào thanh ghi r2.
  - Nếu là file bình thường thì ta lấy số byte thực sự bằng hàm fileSystem→table[fileID]→Read(buffer, charcount). Nếu đọc đến cuối file (số byte < charcount) thì trả về -2 vào thanh ghi r2. Nếu không thì chuyển chuỗi đọc được từ console output sang cho user.

b) System Call int Write(char* buffer, int charcount, OpenFileID id):

Mục đích: Ghi file với tham số là buffer, số ký tự cho phép charcount) và id của file.

- Bước 1, 2, 3, 4, 5 và 6: Tương tự SC_Read.
- Bước 7: Kiểm tra file ID:
  - Nếu fileID = 0 (console input), ta thông báo lỗi cho user (vì không thể viết được console input) và trả về -1 vào thanh ghi r2.
  - Nếu fileID = 1 (console output), ta dùng hàm Write của SynchConsole để viết console input. Sau đó lấy số byte trả về, và ghi vào thanh ghi r2.
  - Nếu là file bình thường, ta chia thành 2 trường hợp con:
    + Đối với file chỉ đọc: Ta không thể đọc được file chỉ đọc, do đó thông báo lỗi cho user và trả về -1 vào thanh ghi r2.
    + Đối với file đọc-ghi: Ta dùng hàm Write của SynchConsole để viết console input. Sau đó lấy số byte trả về, và ghi vào thanh ghi r2.

6) Cài đặt System Call int Seek(int pos, OpenFileID id):

Seek sẽ phải chuyển con trỏ tới vị trí thích hợp. pos lưu vị trí cần chuyển tới, nếu pos = -1 thì di chuyển đến cuối file. Trả về vị trí thực sự trong file nếu thành công và -1 nếu bị lỗi. Gọi Seek trên console phải báo lỗi.

Mục đích: Di chuyển con trỏ đến vị trí thích hợp trong file với tham số là vị trí cần dịch chuyển và id của file.

- Bước 1: Đọc địa chỉ tham số pos từ thanh ghi r4.
- Bước 2: Đọc địa chỉ file id từ thanh ghi r5.
- Bước 3: Kiểm tra pos phải lớn hơn 0. Nếu pos < 0, ta thông báo lỗi cho user và trả về -1 vào thanh ghi r2.
- Bước 4: Kiểm tra file id có nằm ngoài bảng mô tả file không (trong đoạn [0, 9]). Nếu không hợp lệ, thông báo lỗi cho user và trả về -1 vào thanh ghi r2.
- Bước 5: Kiểm tra file id nếu = 0 (console input) hoặc = 1 (console output), ta cũng thông báo lỗi cho user và trả về -1 vào thanh ghi r2.
- Bước 5: Kiểm tra sự tồn tại của file. Nếu file không tồn tại, thông báo cho user và trả về -1 vào thanh ghi r2.

- Bước 6: Nếu tất cả tham số truyền vào hợp lệ và không báo lỗi ở các bước trên, ta tiến hành gán độ dài của file bằng phương thức Length() của lớp fileSystem.
- Bước 7: Kiểm tra nếu pos = -1 hoặc pos > endPos thì seek đến cuối file. Nếu không thì ta seek, truyền vào tham số pos để dịch chuyển con trỏ đến vị trí mong muốn và trả về vị trí dịch chuyển cho r2.

7) Cài đặt System Call int Delete(char* name):

Delete System Call sẽ sử dụng NachOS FileSystem Object để xóa file.

- Bước 1: Đọc địa chỉ của tham số name từ thanh ghi r4
- Bước 2: Sao chép giá trị ở r4 từ user space sang system space bằng hàm User2System().
- Bước 3: Kiểm tra file có đang mở không, sử dụng vòng lặp for để kiểm tra từ bảng mô tả file. Nếu file đang mở, thông báo lỗi cho user và trả về -1 vào thành ghi r2. Sau đó tăng giá trị program counter bằng hàm increasePC().
- Bước 4: Nếu xóa thành công, thông báo cho user và trả về 0 vào thanh ghi r2.
- Bước 5: Nếu file không tồn tại, thông báo cho user và trả về -1 vào thanh ghi r2.

III. Nguồn tham khảo

Lập trình Nachos HCMUS - YouTube

https://github.com/nguyenthanhchungfit/Nachos-Programing-HCMUS