TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM
KHOA CÔNG NGHỆ THÔNG TIN


# ĐỒ ÁN MÔN HỌC

# HỆ ĐIỀU HÀNH

## Đồ án 2: Tìm hiểu system calls - các thao tác với files

Lớp: 20TN

Họ và tên các thành viên:

| | |
|---|---|
| 1. Nguyễn Văn Lộc | MSSV: 20120131 |
| 2. Võ Trọng Nghĩa | MSSV: 20120536 |
| 3. Nguyễn Kiều Minh Tâm | MSSV: 20120572 |

GVLT: Thầy TS. Trần Trung Dũng
GVHDTH: Thầy Nguyễn Thanh Quân
Thầy Lê Giang Thanh

# Mục lục



# Danh sách hình vẽ

# Danh sách bảng

# 1 Thông tin của nhóm – Phân chia công việc

| **MSSV** | **Họ và tên** | **Công việc** | **Đánh giá mức độ hoàn thành** |
|---|---|---|---|
| 20120131 | Nguyễn Văn Lộc | • System call Create<br>• System call Remove<br>• Viết chương trình createfile<br>• Viết chương trình delete<br>• Viết báo cáo cho đồ án | 100% |
| 20120536 | Võ Trọng Nghĩa | • System call Read<br>• System call Write<br>• System call Seek<br>• Viết chương trình cat<br>• Viết chương trình copy<br>• Viết chương trình concatenate<br>• Tester chính của đồ án<br>• Viết báo cáo cho đồ án | 100% |
| 20120572 | Nguyễn Kiều Minh Tâm | • System call Open<br>• System call Close<br>• Viết báo cáo cho đồ án | 100% |

Bảng 1: Bảng thông tin thành viên

# 2 Đanh giá mức độ hoàn thành

| Phần | Câu | Ghi chú | Đánh giá |
|---|---|---|---|
| 1 | 1 | Create | 100% |
| | 2 | Open | 100% |
| | 3 | Close | 100% |
| | 4 | Read | 100% |
| | 5 | Write | 100% |
| | 6 | Seek | 100% |
| | 7 | Remove | 100% |
| | 8 | Sao chép vùng nhớ kernel - user | 100% |
| 2 | 1 | createfile | 100% |
| | 2 | cat | 100% |
| | 3 | copy | 100% |
| | 4 | delete | 100% |
| | 5 | concatenate | 100% |
| | | Không để user làm sụp hệ điều hành | 100% |
| | | Báo cáo | 100% |

Bảng 2: Đánh giá mức độ hoàn thành

# 3 Môi trường làm việc

NachOS được biên dịch và cài đặt trên Ubuntu 18.04 Image của Docker.

# 4 Tổng quan

## 4.1 Giá trị của các thanh ghi

- Mã system call được đưa vào thanh ghi r2.
- Tham số thứ 1 được đưa vào thanh ghi r4.
- Tham số thứ 2 được đưa vào thanh ghi r5.
- Tham số thứ 3 được đưa vào thanh ghi r6.
- Tham số thứ 4 được đưa vào thanh ghi r7.
- Kết quả thực hiện của system call được đưa vào thanh ghi r2.

## 4.2 Các bước viết một system call

Muốn phục vụ người dùng có thể thực thi được các công việc khác nhau, người lập trình hệ điều hành phải xây dựng một bộ các system call đủ để phục vụ các yêu cầu này. System call cũng là hàm xử lý nhưng ở *kernel mode*, khác với một hàm của chương trình người dùng ở *user mode*. Sau đây là các bước viết một system call.
**Bước 1:** Trong tập tin */code/userprog/syscall.h* , thêm dòng khai báo một syscall mới.
**Bước 2:** Thêm các dòng định nghĩa vào tập tin */code/test/start.c* và */code/test/start.s*.
**Bước 3:** Sửa điều kiện **if...** thành **switch... case** trong tập tin *code/userprog/exception.cc*. Trong phần xử lý cho các syscall, tạo tập tin có sử dụng hàm **System2User()** và **User2System()** được hướng dẫn.
**Bước 4:** Viết chương trình ở mức người dùng để kiểm tra.
**Bước 5:** Thêm đoạn code vào */code/test/Makefile*.
**Bước 6:** Biên dịch lại NachOS.
**Bước 7:** Thực thi chương trình test, nếu chương trình không báo lỗi thì xem như thành công.

## 4.3 Chuyển vùng nhớ giữa kernel space và user space

Mã nguồn cho phần chuyển vùng nhớ từ kernel space sang user space và ngược lại được thực hiện như trong các tài liệu hướng dẫn đồ án được giảng viên hướng dẫn thực hành cung cấp.

### 4.3.1 Chuyển vùng nhớ từ kernel space sang user space

Dùng phương thức *WriteMem* của *machine*, viết lần lượt từng ký tự của *buffer* vào user space.

### 4.3.2 Chuyển vùng nhớ từ user space sang kernel space

Dùng phương thức *ReadMem* của *machine*, viết lần lượt từng ký tự của *buffer* vào kernel space.

# 5 Cài đặt các system calls thao tác với files

## 5.1 Cài đặt syscall *int Create(char* name)*

**Các file được chỉnh sửa**

- userprog/ksyscall.h
- userprog/exception.cc
- filesys/filesys.h

**Chức năng:** Tạo một file mới.
**Cách thức thực hiện**

- Sửa lại class FileSystem trong **filesys.h**, thêm tham số *int initialSize* cho hàm *Create*.
- Khi Syscall được gọi, thực hiện chuyển vùng nhớ từ User sang Kernel để lấy tên file. Sau đó gọi phương thức Create vừa tạo từ FileSystem (kernel->fileSystem->Create). Kết quả trả về sẽ lưu vào thanh ghi r2.

## 5.2 Cài đặt syscall *int Open(char* name, int accessType)*

**Các file được chỉnh sửa**

- userprog/ksyscall.h
- userprog/exception.cc
- filesys/filesys.h

**Chức năng:** Mở một file đã có sẵn (với trường hợp chỉ đọc và đọc&ghi) hoặc file chưa có sẵn (với trường hợp đọc&ghi) từ đường dẫn cho trước.
**Cách thức thực hiện**

- Sửa lại class FileSystem trong **filesys.h**, thêm thuộc tính mới là **fdTable**, là con trỏ cấp 2 để lưu danh sách các object OpenFile từ **openfile.h**.
- Sửa lại constructor FileSystem() trong **filesys.h** để khởi tạo **fdTable** gồm 20 phần tử rỗng, với phần tử 0 và 1 được khởi tạo trước (do đề bài không yêu cầu nên 2 phần tử này sẽ được bỏ qua). Đồng thời cũng sửa lại destructor $\sim$ FileSystem() để thu hồi bộ nhớ.

- Thêm phương thức **Open(name, accessType)**. Từ **lib/sysdep.cc** với hàm **OpenForWrite** và **OpenForReadWrite**, dùng hàm **open** để tạo ra linux's file descriptor. Sau đó khởi tạo OpenFile object với tham số là file descriptor này và thêm vào trong **fdTable**, sau đó trả về vị trí được thêm. Nếu bảng đầy thì thực hiện đóng file descriptor, trả về -1. Tương tự trả về -1 khi file descriptor trả về từ open là -1.
- Khi Syscall được gọi, thực hiện chuyển vùng nhớ từ User sang Kernel để lấy tên file, và lấy dữ liệu về loại file cần mở từ thanh ghi. Sau đó gọi phương thức Open vừa tạo từ FileSystem (kernel->fileSystem->Open). Kết quả trả về sẽ lưu vào thanh ghi r2.

## 5.3 Cài đặt syscall *int Read(char* buffer, int size, int ID)*

**Các file được chỉnh sửa**

- userprog/ksyscall.h
- userprog/exception.cc
- filesys/filesys.h
- filesys/openfile.h

**Chức năng:** Đọc nội dung từ file vào vùng nhớ, với kích thước cho trước.
**Cách thức thực hiện**

- Thêm phương thức **get(ID)** ở **filesys.h** để lấy object OpenFile từ **fdTable** với ID cho trước.
- Thêm phương thức **isReadable** ở **openfile.h** để kiểm tra file có thể đọc được không thông qua hàm **fcntl**.
- Khi Syscall được gọi, lưu thông tin về địa chỉ ảo, kích thước cần đọc và file ID từ thanh ghi
- Cấp phát vùng nhớ trong kernel với kích thước cần đọc.
- Nếu ID là 0, thực hiện gọi GetChar() từ lớp SynchConsoleInput (kernel->synchConsoleIn) để lấy ký tự từ console lưu vào vùng nhớ được cấp phát ở trên.
- Nếu ID lớn hơn 1, thực hiện lấy object OpenFile từ phương thức Get thông qua tham số ID, kiểm tra file có thể đọc và gọi phương thức Read của object OpenFile vừa lấy được để lưu dữ liệu của file vào vùng nhớ được cấp phát.
- Nếu ID khác 2 trường hợp trên, thực hiện hủy vùng nhớ và trả về -1 vào thanh ghi r2.
- Chuyển vùng nhớ về User space.
- Hủy vùng nhớ và trả về số ký tự đọc được vào thanh ghi r2.

## 5.4 Cài đặt syscall *int Write(char* buffer, int size, int ID)*

**Các file được chỉnh sửa**

- userprog/ksyscall.h
- userprog/exception.cc
- filesys/openfile.h

**Chức năng:** Ghi vào file nội dung từ vùng nhớ, với kích thước cho trước.
**Cách thức thực hiện**

- Thêm phương thức **isWritable** ở **openfile.h** để kiểm tra file có thể ghi được không thông qua hàm **fcntl**.
- Khi Syscall được gọi, lưu thông tin về địa chỉ ảo, kích thước cần đọc và file ID từ thanh ghi
- Chuyển vùng nhớ từ User space sang Kernel space.
- Nếu ID là 1, thực hiện gọi PutChar() từ lớp SynchConsoleOutput (kernel->synchConsoleOut) để in ký tự từ vùng nhớ vào console.
- Nếu ID lớn hơn 1, thực hiện lấy object OpenFile từ phương thức Get thông qua tham số ID, kiểm tra file có thể ghi và gọi phương thức Write của object OpenFile vừa lấy được để ghi dữ liệu từ vùng nhớ vào file.
- Nếu ID khác 2 trường hợp trên, thực hiện hủy vùng nhớ và trả về -1 vào thanh ghi r2.
- Hủy vùng nhớ và trả về số ký tự ghi được vào thanh ghi r2.

## 5.5 Cài đặt syscall *int Seek(int position, int ID)*

**Các file được chỉnh sửa**

- userprog/ksyscall.h
- userprog/exception.cc
- filesys/openfile.h

**Chức năng:** Di chuyển con trỏ file tới vị trí cho trước (tính từ đầu file).

- Thêm phương thức **Seek** ở **openfile.h**. Nếu vị trí đưa vào là -1 thì dời offset về cuối file, trả về kích thước của file. Nếu vị trí đưa vào nằm trong đoạn từ 0 đến kích thước của file, dời offset đến vị trí mới và trả về offset đó. Nếu không thuộc cả 2 trường hợp trên thì coi như vị trí không phù hợp, trả về -1.
- Khi Syscall được gọi, lưu thông tin về vị trí, và file ID từ thanh ghi.
- Nếu ID nhỏ hơn 2, thông báo lỗi và trả về giá trị -1 vào thanh ghi r2.
- Ngược lại, thực hiện lấy object OpenFile từ phương thức Get thông qua tham số ID và gọi phương thức Seek của object OpenFile vừa lấy được.
- Trả về vị trí mới vào thanh ghi r2.

## 5.6 Cài đặt syscall *int Remove(char* name)*

**Các file được chỉnh sửa**

- userprog/ksyscall.h
- userprog/exception.cc
- filesys/filesys.h
- filesys/openfile.h

**Chức năng:** Xóa một file với tên cho trước.
**Cách thức thực hiện**

- Thêm phương thức *getINode* trong **openfile.h**, sử dụng cấu trúc **stat** của C++ để lấy giá trị INode number của tập tin hiện tại.
- Sửa lại phương thức *Remove(name)* trong **filesys.h**: mở file cần xóa ra (để kiểm tra tên file có tồn tại không), lấy giá trị INode, sau đó so sánh với giá trị INode của các file đang mở trong file descriptor table, để kiểm tra xem file cần xóa có đang được mở không, nếu không thì xóa, nếu đang mở thì báo lỗi và return -1.
- Khi Syscall được gọi, thực hiện chuyển vùng nhớ từ User sang Kernel để lấy tên file. Sau đó gọi phương thức Remove từ FileSystem (kernel->fileSystem->Remove). Kết quả trả về sẽ lưu vào thanh ghi r2.

## 5.7 Cài đặt syscall *int Close(int ID)*

**Các file được chỉnh sửa**

- userprog/ksyscall.h
- userprog/exception.cc
- filesys/filesys.h

**Chức năng:** Đóng một file đã mở với ID cho trước.
**Cách thức thực hiện**

- Sửa lại phương thức Close(ID) trong **filesys.h**: Nếu file có ID không phù hợp (ID nằm ngoài đoạn 2 đến 19) hoặc đã được đóng hoặc chưa được mở (fdTable[ID] == NULL) thì trả về -1. Ngược lại xóa object đó đi và gán lại NULL trong fdTable, trả về 0
- Khi Syscall được gọi, lấy thông tin về ID file cần đóng thông qua thanh ghi. Sau đó gọi phương thức Close từ FileSystem (kernel->fileSystem->Close). Kết quả trả về sẽ lưu vào thanh ghi r2.

# 6 Các chương trình người dùng

*Phần này chỉ nhắc đến các syscall được cài đặt trong đồ án này.*

## 6.1 Chương trình *createfile*

**Các syscall sử dụng**: Create.
**Nội dung chương trình**

- Đọc tên file cần tạo từ console.
- Tạo file băng cách gọi syscall Create.
- Nếu kết quả trả về 0 thì in ra thông báo đã tạo thành công, ngược lại báo lỗi.

## 6.2 Chương trình *cat*

**Các syscall sử dụng**

- Open
- Close
- Read
- Write

**Nội dung chương trình**

- Đọc đường dẫn tới file cần đọc từ console.
- Mở file chỉ đọc thông qua syscall Open, nếu có lỗi thì thông báo và kết thúc chương trình.
- Thực hiện đọc file theo từng Block 256 ký tự với syscall Read.
- Thực hiện in Block lên màn hình thông qua syscall Write để đảm bảo nội dung của file (in cả những kí tự không in được).
- Thực hiện lại 2 bước trên cho đến khi hết file.
- Đóng file thông qua syscall Close.

## 6.3 Chương trình *copy*

**Các syscall sử dụng**

- Open
- Close
- Read
- Write

**Nội dung chương trình**

- Đọc đường dẫn tới file nguồn cần đọc từ console.

- Đọc đường dẫn tới file đích cần ghi từ console.
- Mở file nguồn chỉ đọc và file đích đọc & ghi thông qua syscall Open, nếu có lỗi thì thông báo và kết thúc chương trình.
- Thực hiện đọc file nguồn theo từng Block 256 ký tự với syscall Read.
- Thực hiện ghi Block vào file đích thông qua syscall Write.
- Thực hiện lại 2 bước trên cho đến khi hết file nguồn.
- Đóng file nguồn và đích thông qua syscall Close.

## 6.4 Chương trình *concatenate*

**Các syscall sử dụng**

- Open
- Close
- Read
- Write
- Seek

**Nội dung chương trình**

- Đọc đường dẫn tới file nguồn 1 từ console.
- Đọc đường dẫn tới file nguồn 2 (nội dung nối vào sau file nguồn 1) từ console.
- Mở file nguồn 1 đọc&ghi và file nguồn 2 chỉ đọc thông qua syscall Open, nếu có lỗi thì thông báo và kết thúc chương trình.
- Thực hiện chuyển vị trí con trỏ file nguồn 1 về cuối file thông qua syscall Seek.
- Thực hiện đọc file nguồn 2 theo từng Block 256 ký tự với syscall Read.
- Thực hiện ghi Block vào file nguồn 1 thông qua syscall Write.
- Thực hiện lại 2 bước trên cho đến khi hết file nguồn 2.
- Đóng file nguồn 1 và file nguồn 2 thông qua syscall Close.

## 6.5 Chương trình *delete*

**Các syscall sử dụng**: Remove.
**Nội dung chương trình**

- Đọc tên file cần xóa từ console.
- Sử dụng syscall Remove, kiểm tra các điều kiện cần để xóa, nếu thỏa mãn thì tiến hành xóa file.
- Nếu kết quả trả về 0 thì in ra thông báo đã xoá thành công, ngược lại báo lỗi.

# Lời cảm ơn

Chúng em xin chân thành cảm ơn thầy Trần Trung Dũng, giảng viên lý thuyết, và thầy Nguyễn Thanh Quân, thầy Lê Giang Thanh là giảng viên hướng dẫn thực hành đã tạo điều kiện cho chúng em được tìm hiểu thêm về cách cài đặt các system call, cùng với đó để chúng em hiểu thêm về cách thức hệ điều hành hoạt động khi thực hiện các thao tác trên tập tin. Tuy vậy, trong quá trình thực hiện đồ án, chúng em khó tránh khỏi thiếu sót do chưa hiểu đúng vấn đề, chúng em hi vọng được thầy hướng dẫn cũng như góp ý để chúng em có thể hiểu hơn nữa về các vấn đề liên quan đến đồ án lần này.

# Tài liệu tham khảo chủ yếu


[1] *Các tài liệu hướng dẫn thực hành đồ án.*

[2] Nguyễn Thành Chung. *Các video hướng dẫn lập trình NachOS*. URL: youtube.com/watch?v=t0jtY1C129s&list=PLRgTVtca98hUgCN2_2vzsAAXPiTFbvHp0.