ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN



HỆ ĐIỀU HÀNH

ĐỒ ÁN 1: SYSTEM CALL

*Giảng viên hướng dẫn:* **thầy Phạm Tuấn Sơn**

*Nhóm thực hiện*: **18120078 – Ngô Phù Hữu Đại Sơn**

**18120211 – Võ Thế Minh**

**18120227 – Phạm Văn Minh Phương**

*Lớp*: **HĐH 18\_4**

*Khóa*: **2018**

MỤC LỤC

[A. Thông tin khái quát 2](#_Toc55056361)

[I. Thông tin nhóm 2](#_Toc55056362)

[II. Bảng phân công công việc 2](#_Toc55056363)

[B. Nội dung 3](#_Toc55056364)

[I. Môi trường phát triển 3](#_Toc55056365)

[II. Mục tiêu của đồ án 3](#_Toc55056366)

[III. Ý tưởng thiết kế 3](#_Toc55056367)

[IV. Triển khai 4](#_Toc55056368)

[V. Kiểm thử 5](#_Toc55056369)

[C. TỔNG KẾT 6](#_Toc55056370)

[VI. Đánh giá đồ án 6](#_Toc55056371)

[1. Mức độ hoàn thành của các thành viên 6](#_Toc55056372)

[2. Mức độ hoàn thành đồ án: 6](#_Toc55056373)

[VII. Nguồn tham khảo 6](#_Toc55056374)

# Thông tin khái quát

## I. Thông tin nhóm

|  |  |  |
| --- | --- | --- |
| MSSV | HỌ TÊN | VAI TRÒ |
| 18120211 | Võ Thế Minh | Trưởng nhóm |
| 18120078 | Ngô Phù Hữu Đại Sơn | Thành viên |
| 18120227 | Phạm Văn Minh Phương | Thành viên |

## II. Bảng phân công công việc

|  |  |
| --- | --- |
| MSSV | CÔNG VIỆC PHỤ TRÁCH |
| 18120078 | Cài đặt nachOS |
| 18120078 | Tái cấu trúc lớp FileSystem |
| 18120078 | Tái cấu trúc lớp OpenFile |
| 18120078 | Cài đặt + kiểm thử system call Create |
| 18120078 | Cài đặt + kiểm thử system call Open |
|  | Cài đặt + kiểm thử system call Read |
| 18120078 | Cài đặt + kiểm thử system call Seek |
|  | Cài đặt + kiểm thử system call Write |
| 18120078 | Cài đặt + kiểm thử system call Close |
|  | Cài đặt + kiểm thử chương trình Echo |
|  | Cài đặt + kiểm thử chương trình Cat |
|  | Cài đặt + kiểm thử chương trình Copy |

# Nội dung

## Môi trường phát triển

* *Máy ảo*: nachOS 3.4
* *Hệ điều hành:* Ubuntu 14.04-i386
* *Compiler*: gcc - 3.4.6
* *Cross-compiler*: gcc - 2.95.3
* *Binutils*: 2.11.2

## Mục tiêu của đồ án

1. Tìm hiểu cách cài đặt hệ điều hành giả lập nashOS
2. Tìm hiểu cách thức giao tiếp giữa HĐH nashOS và chương trình người dùng
3. Tìm hiểu cách viết các system call cho HĐH nashOS.

## Ý tưởng thiết kế

* 1. **Chuyển đổi dữ liệu liệu giữa System space và User space**
  2. **Từ System space đến User space**

Sử dụng hàm WriteMem (có chức năng ghi 1, 2 hoặc 4 bytes từ System space sang User space) đã được định nghĩa sẵn của lớp Machine, xây dựng phương thức public System2User cho lớp Machine để chép dữ liệu từ 1 vùng nhớ ở System space sang User space

* 1. **Từ User space đến System space**

Sử dụng hàm ReadMem (có chức năng đọc 1, 2 hoặc 4 bytes từ User space sang System space) đã được định nghĩa sẵn của lớp Machine, xây dựng phương thức public User2System cho lớp Machine để chép dữ liệu từ 1 vùng nhớ ở User space sang System space

* 1. **Chén lớp SynchConsole và nashOS để đọc ghi từ thiết bị xuất nhật chuẩn**

HĐH nashOS không cung cấp cho ta các phương thức xuất nhập từ thiết bị xuất/nhập chuẩn nên ta cần phải chèn lớp SynchConsole vào nachos để xây dựng được các system call có sử dụng dữ liệu từ console. Đồng thời cũng dung để xuất các thông báo cho chương trình người dung.

* 1. **Tái cấu trúc lớp FileSystem**

HĐH nashOS ban đầu chỉ cung cấp cho ta lớp FileSystem quản lý việc nhập/xuất của 1 file tại 1 thời điểm. Vì vậy cần tái cấu trúc lại lớp này để có thể quản lý nhiều file hơn tại 1 thời điểm. Sử dụng 1 mảng để quản lý các file đang mở (quản lý tối đa 10 file trong cùng 1 thời điểm). 2 phần tử đầu tiên của mảng này dùng để quản lý thiết bị xuất nhập chuẩn.

Lưu ý: Nachos sử dụng 2 hệ thống quản lý file của UNIX và của chính nashOS nên để dịch được và an toàn thì t phải chỉnh sử cả 2 lớp này.

* 1. **Tái cấu trúc lớp OpenFile**

Lớp OpenFile của nachos mặc định khi mở file thì luôn có thể dùng để đọc và ghi. Ta cần dùng 1 thuộc tính type để phân biệt các file mở chỉ để đọc, có khả năng đọc và ghi, thiết bị nhập chuẩn, thiết bị xuất chuẩn.

* 1. **Xây dựng system call Create**

Sao chép tên của tập tin muốn tạo từ User space sang System space và dùng phương thức Create của đối tượng fileSystem (đã được khai báo tĩnh tại threads/system.h và threads/system.cc) để tạo 1 file rỗng. Nếu thành công sẽ trả về 0 ngược lại trả về -1.

* 1. **Xây dựng system call Open**

Truyền vào 2 tham số là name và type với type là loại file muốn mở:

Type = 0 : File chỉ đọc

Type = 1: File có thể đọc và ghi

Đối với file có thể đọc và ghi, nếu file chưa tồn tại thì phải tạo file trước khi mở. File chỉ đọc nếu chưa tồn tại mở sẽ bị lỗi. Nếu mở file không bị lỗi sẽ thanh file mới vào danh sách các file đang quản lý (tối đa 10 file trong cùng 1 thời gian). Nếu mở file thành công trả về 0.

* 1. **Xây dựng system call Close**

Nếu file đang muốn đóng có nằm trong danh sách các file đang quản lý thì sẽ thực hiện xóa file này khỏi danh sách các file đang quản lý. Ngươc lại nếu không có sẽ trả về -1. Lưu ý không thể đóng 2 file đầu tiên của danh sách (stdin và stdout).

* 1. **Xây dựng system call Read**
  2. **Xây dựng system call Write**
  3. **Xây dựng system call Seek**

Sửa dụng hàm Seek có sẵn của lớp OpenFile, dịch chuyển offset hiện tại đến offset mong muốn. Vị trí mới không được vượt quá kích thước của file. Và chỉ có thể thực hiện Seek với các file đang được mở. Nếu vị trì mới là -1 thì dịch chuyển đến cuối file. Nếu Seek thành công trả về 0, ngược lại trả về -1.

* 1. **Xây dựng chương trình echo**
  2. **Xây dựng chương trình cat**
  3. **Xây dựng chương trình copy**

## Triển khai

* 1. **User2System (Đọc dữ liệu từ User space sang System Space)**

Đầu vào: virtualAddr: con trỏ đến vùng nhớ tại User space.

Limit: Số kí tự tối đa có thể đọc.

Tạo kernelBuffer (1 vùng nhớ kiểu char với số lượng kí tự = limit + 1). Vì phải tính thêm kí tự kết thúc chuỗi.

Tạo oneChar (1 biến kiểu char) để đọc từng kí tự từ User space sang System space. Ta không thể đọc 1 lần hết tất cả kí tự từ User space vì:

* Chưa biết được số lượng kí tự thực sự phải đọc là bao nhiêu.
* Hàm ReadMem của lớp Machine có tó thể đọc 1,2 hoặc 4 bytes trong 1 lần đọc.

Việc đọc lặp lại và sẽ dừng nếu ta đọc được kí tự “\0” hoặc “\n” (thêm 1 ký tự “\0” phía sau “\n” trước khi kết thúc).

Kết quả trả về kernelBuffer.

* 1. **System2User (Ghi dữ liệu từ System space sang User space)**

Đầu vào: virtAddr: con trỏ đến vùng nhớ tại User space.

len: số lượng kí tự muốn ghi vào virtualAddr.

buffer: Vùng nhớ tại System space.

Nếu số lượng kí tự muốn ghi < 0 thì trả về -1.

Nếu số lương kí tự muốn ghi = 0 thì trả về 0 và thoát chương trình.

Tương tự như User2System, ta không biết trước được số kí tự thực sự phải ghi nên ta phải thực hiện thêm từng kí tự đến khi gặp kí tự kết thúc chuỗi hoặc số kí tự đã ghi được = len.

Kết quả trả về số ký tự thực sự đã ghi.

* 1. **Tái cấu trúc FileSystem**
* Thêm 2 thuộc tính cho lớp FileSystem: OpenFile \*\* openFiles : mảng các file

Size: số lượng file đang quản lý

* FileSystem:

Khởi tạo các phần trong openFiles = NULL.

Thêm 2 phần tử stdin và stdout vào đầu openFiles.

* Open:

Nếu file không tồn tại thì sẽ trả về -1

Nếu size = 10 thì trả về -1

Duyệt qua openFiles và mở file vào phần tử NULL đâu tiên trong mảng.

Trả về vị trí của file đang mở trong mảng.

* 1. **Tái cấu trúc OpenFile**

Thêm thuộc tính type thể hiện file mở có chức năng gì:

* + 1. chỉ đọc
    2. đọc và ghi
    3. là stdin
    4. là stdout
  1. **Xây dựng system call Create**

Đọc filename (tên file) từ User space sang System space.

Nếu filename = NULL thì trả về -1.

Dùng hàm Create của lớp FileSystem để tạo 1 file rỗng.

Nếu tạo thành công trả về 0, ngược lại trả về -1.

* 1. **Xây dựng system call Open**

Kiểm tra loại file là “chỉ đọc” hoặc “đọc và ghi”

Kiểm tra số lượng file đang quản lý đã đầy chưa

Đọc tên file và System space

Nếu loại file là “đọc và ghi” thì tạo file trước khi mở

Mở file và lấy id của file vừa mở, nếu id = -1 thì không mở dược file.

Trả về id của file vừa mở

* 1. **Xây dựng system call Close**

fileID là id của file muốn đóng.

Nếu file đang không đươc mở thì trả về -1.

Xóa file trong danh sách các file đang quản lý.

* 1. **Xây dựng system call Read**
  2. **Xây dựng system call Write**
  3. **Xây dựng system call Seek**

pos là vị trí offset muốn đến

fileID là id của file

Nếu file đang không mở thì trả về -1 (Không được seek trong stdin và stdout)

Kiểm tra fileID có hợp lệ không

Nếu pos = -1 thì seek đến cuối file

Nếu offset muốn đến nằm ngoài kích thước của file thì trả về -1

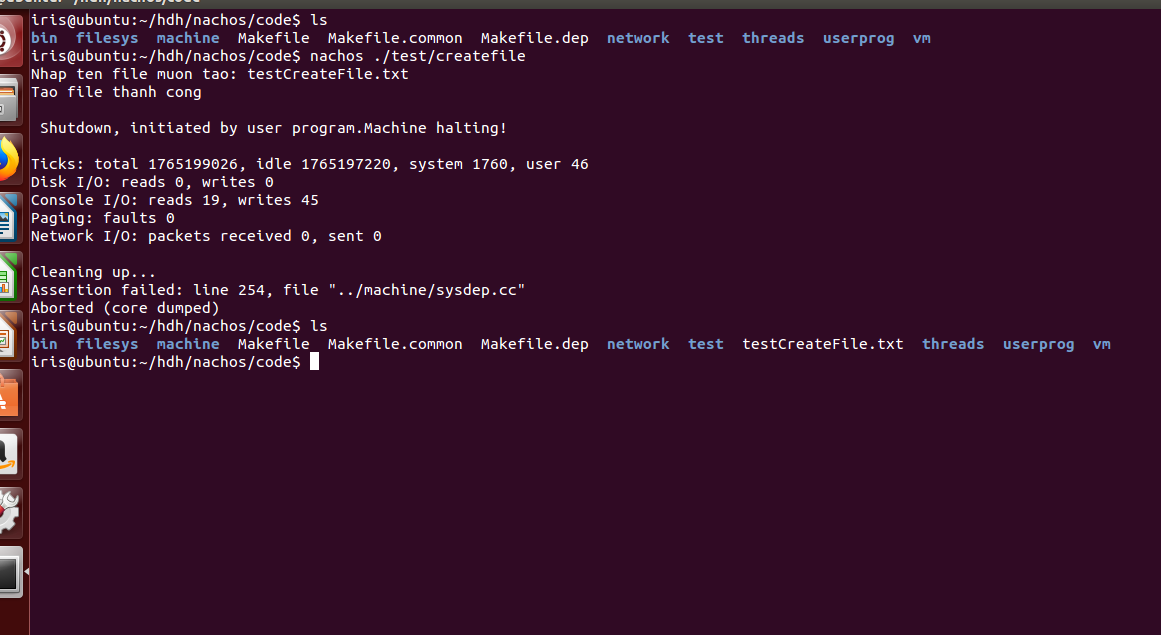
Di chuyển đến offset mới

Trả về vị trí offset vừa đến.

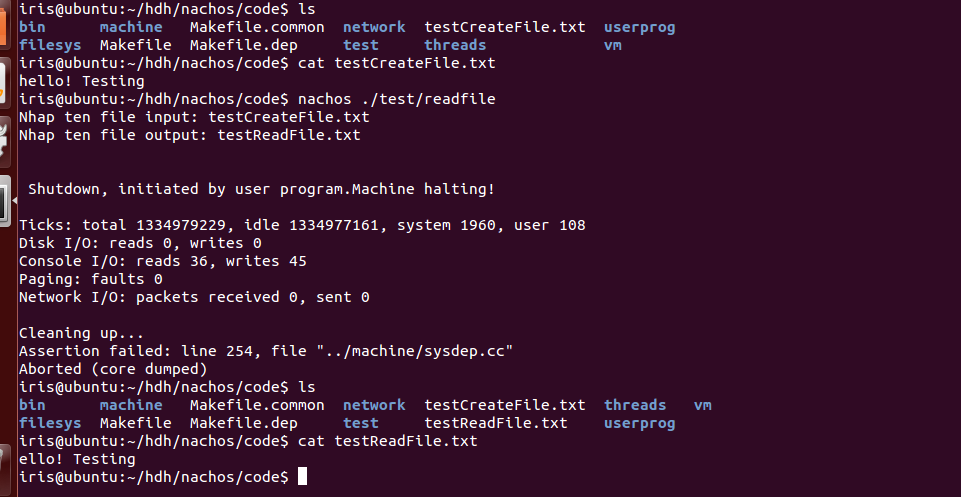
* 1. **Xây dựng chương trình echo**
  2. **Xây dựng chương trình cat**
  3. **Xây dựng chương trình copy**

## Kiểm thử

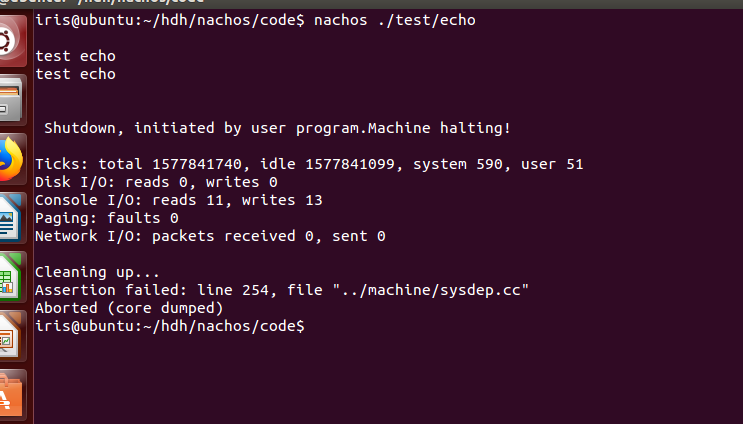
* 1. **Create**
     + - *Input:* chuỗi kí tự tên file
       - *Output:* tạo file và đưa thông báo “Tạo file thành công” hoặc “Tạo file không thành công”.



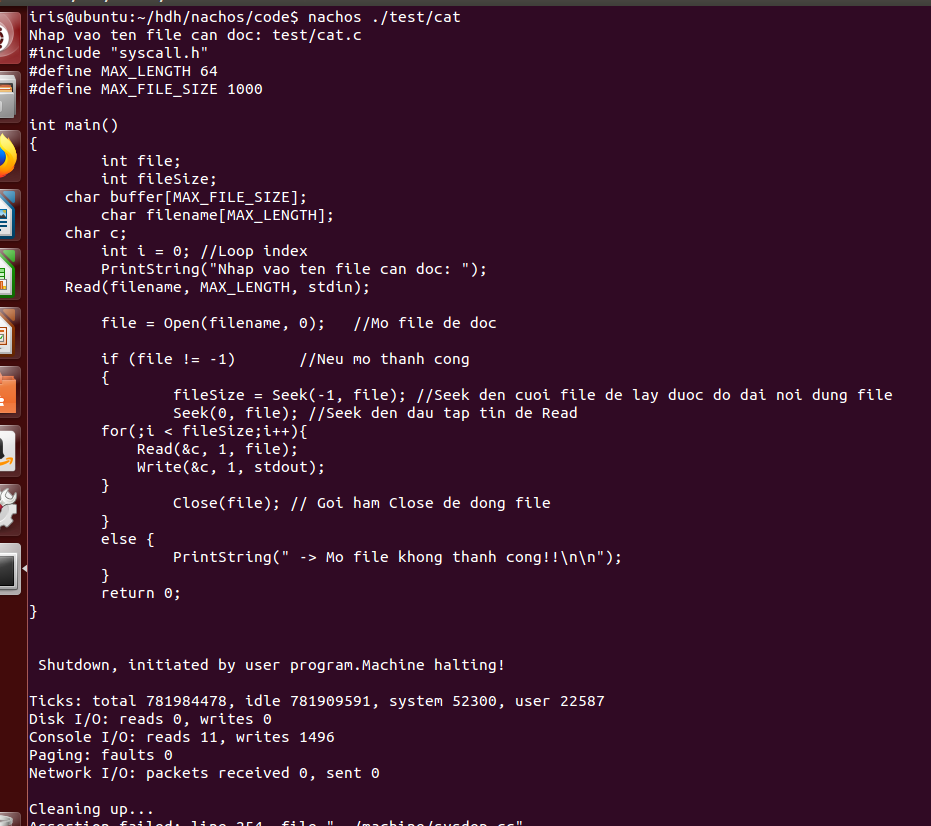
* 1. **Open, Read, Seek, Write, Close**
     + - Input: Tên file input, tên file output
       - Output: Thông bao đọc ghi file không thành công



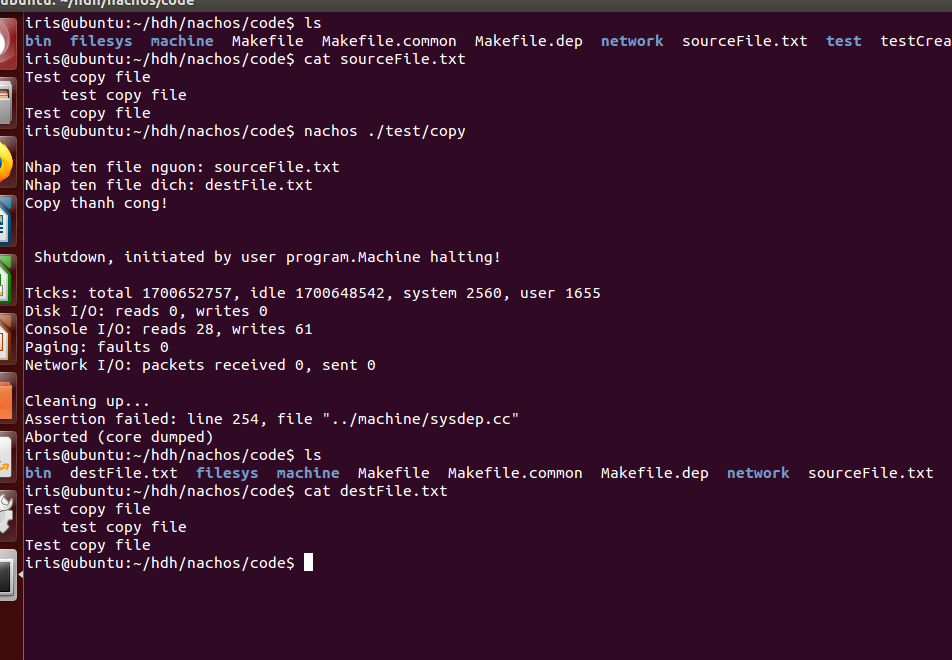
* 1. **Echo**
     + - *Input***:** Nhập 1 chuỗi từ bàn phím.
       - *Output***:** Xuất chuỗi đó ra console.



* 1. **Cat**
     + - *Input:* tên file
       - *Output:* xuất nội dung của file ra màn hình console



* 1. **Copy**
     + - *Input:* tên file nguồn, tên file đích
       - *Output:* Copy file từ nguồn tới đích và in thông báo



# TỔNG KẾT

## Đánh giá đồ án

### Mức độ hoàn thành của các thành viên

|  |  |  |
| --- | --- | --- |
| MSSV | Mức độ hoàn thành công việc | Đóng góp |
| 18120078 |  |  |
| 18120211 |  |  |
| 18120227 |  |  |

### Mức độ hoàn thành đồ án:

**Đánh giá**:

## Nguồn tham khảo

* [File System trong nachos](https://www.cas.mcmaster.ca/~qiao/courses/cs3mh3/roadmaps/4.0/filesys.html)