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**

Ngô Phù Hữu Đại Sơn	18120078
Võ Thế Minh	18120211
Phạm Văn Minh Phương	18120227
Lớp:	HĐH 18_4
Khóa:	2018

31/10/2020 – THÀNH PHỐ HỒ CHÍ MINH

MỤC LỤC

A. 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	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
18120211	Cài đặt + kiểm thử system call Read
18120211	Cài đặt + kiểm thử system call Seek
18120211	Cài đặt + kiểm thử system call Write
18120078	Cài đặt + kiểm thử system call Close
18120227	Cài đặt + kiểm thử chương trình Echo
18120227	Cài đặt + kiểm thử chương trình Cat
18120227	Cài đặt + kiểm thử chương trình Copy

B. NỘI DUNG

I. 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

II. 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.

III. Ý tưởng thiết kế

1. Chuyển đổi dữ liệu liệu giữa System space và User space

a. 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

b. 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

2. Chèn lớp SynchConsole vào 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.

3. 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 dung để quản lý thiết bị xuất nhập chuẩn.

4. 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.

5. 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.

6. 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.

7. 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. Ngược lại nếu không có sẽ trả về -1. Lưu ý không thể đóng 2 file đầu tiên của danh sách (stdin và stdout).

8. Xây dựng system call Read

- Dùng function openFileId đã được định nghĩa trong filesys để mở và kiểm tra tính đúng đắn của file. Trường hợp file không tồn tại, id file không hợp lệ và file có kiểu là stdout thì đọc file thất bại trả về -1.
- Trường hợp đọc file stdin thì dùng phương thức System2User đã được định nghĩa trong lớp machine để chuyển dữ liệu từ System->User, đọc file thành công trả về số byte đọc được.
- Trường hợp đọc file rỗng trả về -2.

9. Xây dựng system call Write

- Tương tự như syscall Read dùng openFileId để mở file và kiểm tra tính đúng đắn của file. Trường hợp file không tồn tại, id file không hợp lệ và file có kiểu là stdin thì đọc thất bại trả về -1
- Trường hợp đọc file stdout thì dùng phương thức User2System để chuyển dữ liệu User->System, ghi file thành công trả về số byte thực sự ghi được
- Trường hợp đọc file read&write thì trả về số byte thực sự của file đó
- Trường hợp ghi file rỗng trả về -2

10. 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.

11. Xây dựng chương trình echo

Dựa vào các system call đã được xây dựng trước đó (Read, Write) để có thể lấy chuỗi người dùng nhập từ stdin và xuất chuỗi ra stdout.

12. Xây dựng chương trình cat

Dựa vào các system call đã được xây dựng trước đó (Open, Close, Read, Write, Seek) để đọc tên file từ stdin, mở và đóng file, tính kích thước nội dung file và xuất nội dung file ra stdout.

13. Xây dựng chương trình copy

Dựa vào các system call đã được xây dựng trước đó (Open, Close, Read, Write, Seek) để đọc tên file từ stdin, mở và đóng file, tính kích thước nội dung file nguồn và chép từng kí tự nội dung từ file nguồn qua file đích.

IV. 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.

2. 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.

3. 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.

4. Tái cấu trúc OpenFile

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

- chỉ đọc
- đọc và ghi
- là stdin
- là stdout

5. 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.

6. 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

7. 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.

8. Xây dựng system call Read

int Read(char *buffer, int charcount, OpenFileID id)

If (file rỗng || id file unValid || file stdout):

    Return -1

If (Doc file stdin):

    Length := chiều dài byte đọc được

    Chuyển dữ liệu đọc được cho User

    Return length

if (Đọc file thành công):

    Chuyển dữ liệu đọc được cho user

    Trả về số byte của file

Else:

    Return -2

Xóa buffer

9. Xây dựng system call Write

If (Đọc file rỗng || file stdin ||file read || id file unValid):

Return -1

if(Đọc file stdout): //Ghi lên console

while(file chưa kết thúc):

Đọc từng ký tự

Dùng syncConsole ghi Buffer

Lưu số byte ghi được

if (Đọc file thành công): //Ghi lên file

NextAddress := Vị trí hiện tại của file

Lưu số byte ghi được

Else: (Ghi file rỗng):

Return -2

Xóa Buffer

10. 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.

11. Xây dựng chương trình echo

Chương trình echo sẽ xuất lại chuỗi kí tự mà người dùng nhập từ console.
Gọi system call Read với 3 tham số là địa chỉ buffer, số kí tự tối đa và OpenFileID (Trong trường hợp này là stdin, được define là 0) để đọc chuỗi người dùng nhập từ console. Nếu hàm Read đọc được thành công thì nó sẽ trả về độ dài chuỗi thực sự đọc được vào biến length. Nếu length khác -1 và khác -2 (Trường hợp file lỗi hoặc EOF), tức là đọc được chuỗi từ stdin thành công, system call Write sẽ được gọi với 3

tham số là địa chỉ buffer, số kí tự của chuỗi đọc được và OpenFileID (Trong trường hợp này là stdout, được define là 1) để xuất chuỗi đọc được ra console.

12. Xây dựng chương trình cat

Chương trình cat sẽ hiển thị nội dung file với filename được nhập từ console.
Gọi system call Read để đọc tên file người dùng nhập từ console. System call Open được gọi để mở file vừa nhập, nếu mở file thành công thì sẽ dùng system call Seek để dịch con trỏ file về cuối file để lấy kích thước nội dung file và chuyển con trỏ file về đầu file để tiến hành đọc và xuất ra stdout từng kí tự thông qua system call Read và Write (Từ đầu đến hết kích thước file). Sau khi xuất hết nội dung file thì đóng file bằng system call Close. Nếu mở file không thành công sẽ thông báo cho người dùng.

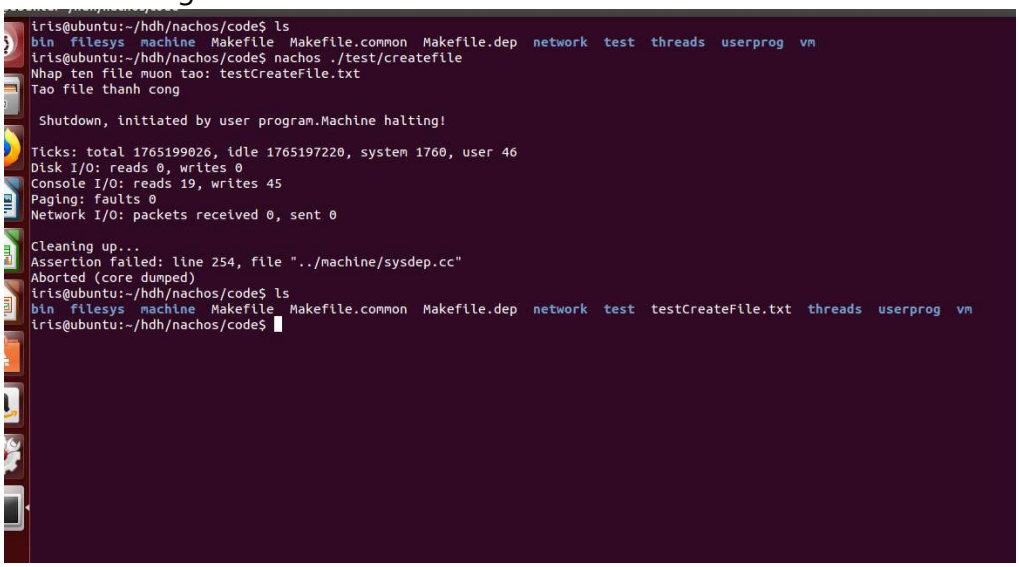
13. Xây dựng chương trình copy

Chương trình copy nhận tên file nguồn và tên file đích để thực hiện copy nội dung từ file nguồn sang file đích.
Gọi system call Read 2 lần để đọc 2 tên file nguồn và đích. Tiếp theo gọi system call Open với type = 0 để mở file nguồn, nếu mở file nguồn thành công thì mở file đích (type = 1). OpenFileID của file nguồn và file đích phải khác nhau. Nếu mở cả 2 file thành công thì tiến hành dịch con trỏ đến cuối file nguồn bằng system call Seek để lấy kích thước nội dung file nguồn và sau đó dịch cả con trỏ file nguồn và file đích về đầu file rồi dùng system call Read để đọc từng kí tự file nguồn, Write để chép từng kí tự đọc được sang file đích cho đến khi hết nội dung file nguồn. Sau khi copy xong nội dung thì đóng cả 2 file bằng system call Close. Nếu ở bước tạo file đích xảy ra lỗi thì sẽ thông báo và đóng file nguồn.

V. Kiểm thử

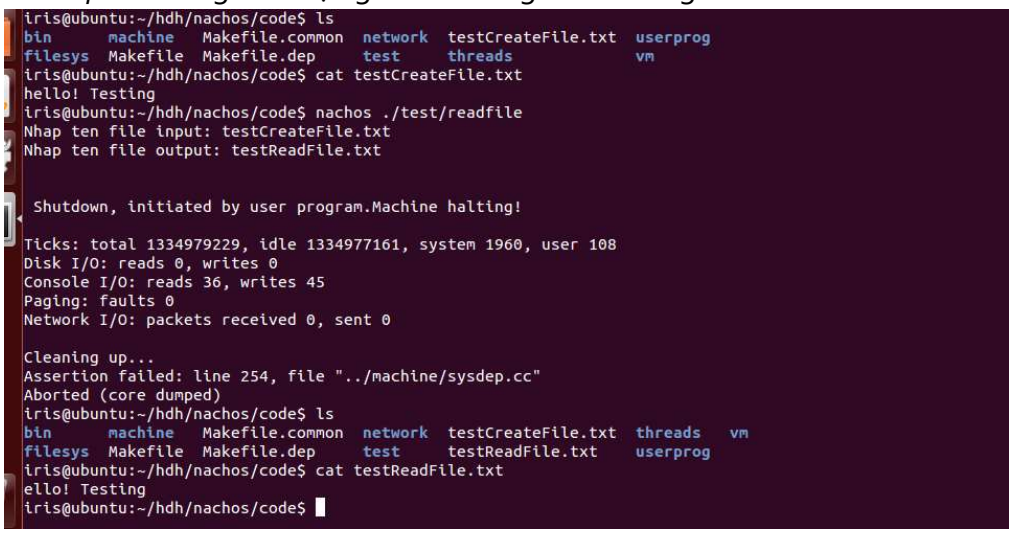
1. Create

1. *Input:* chuỗi kí tự tên file
2. *Output:* tạo file và đưa thông báo "Tạo file thành công" hoặc "Tạo file không thành công".



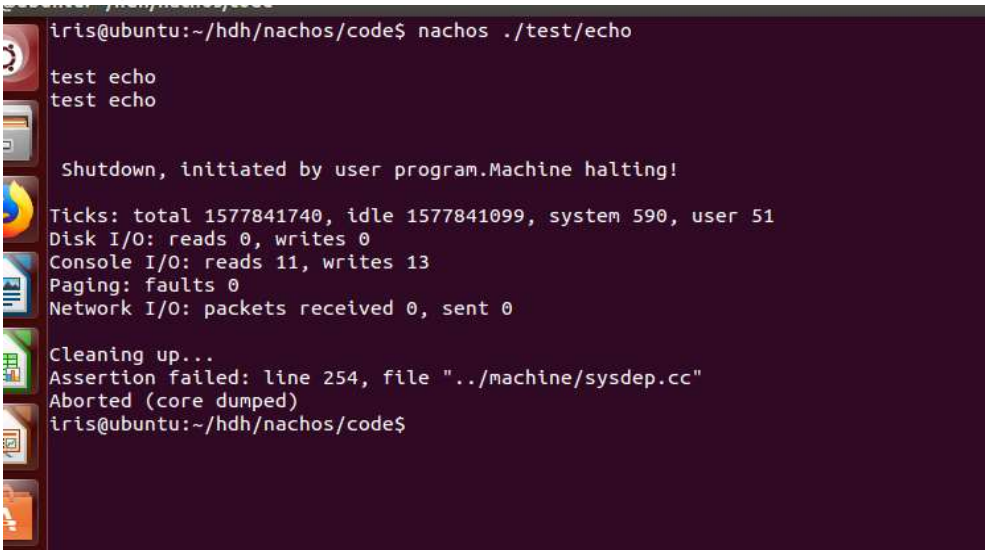
2. Open, Read, Seek, Write, Close

- *Input*: Tên file input, tên file output
- *Output*: Thông bao đọc ghi file không thành công



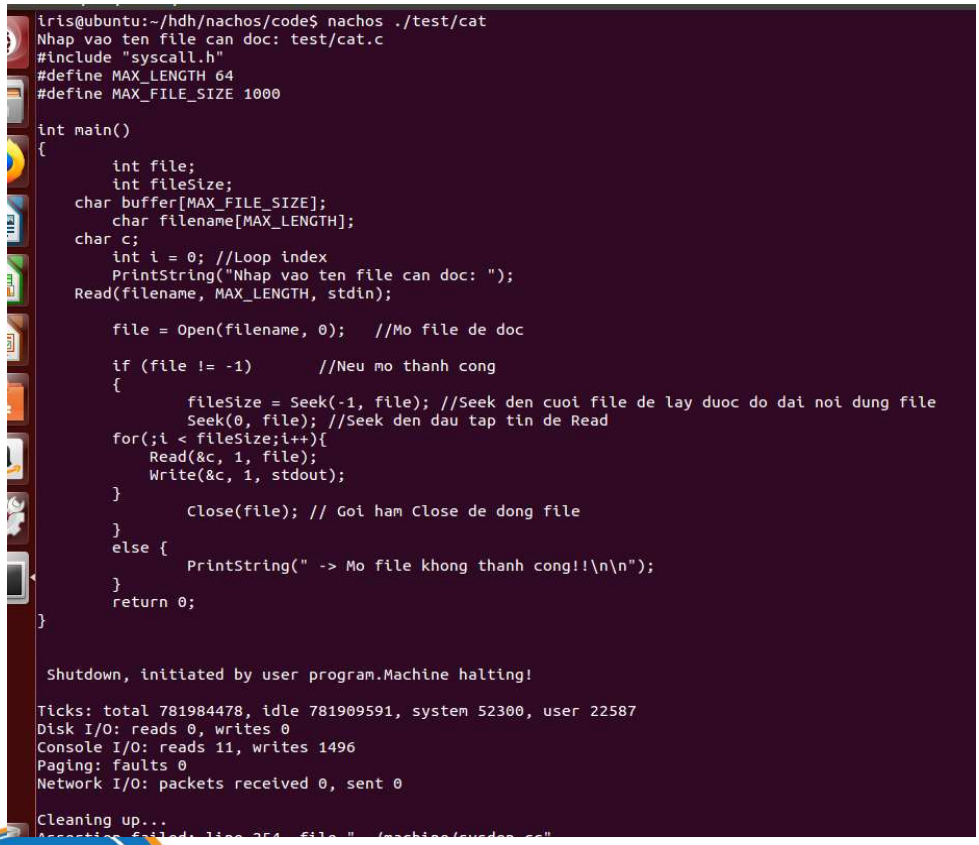
3. Echo

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



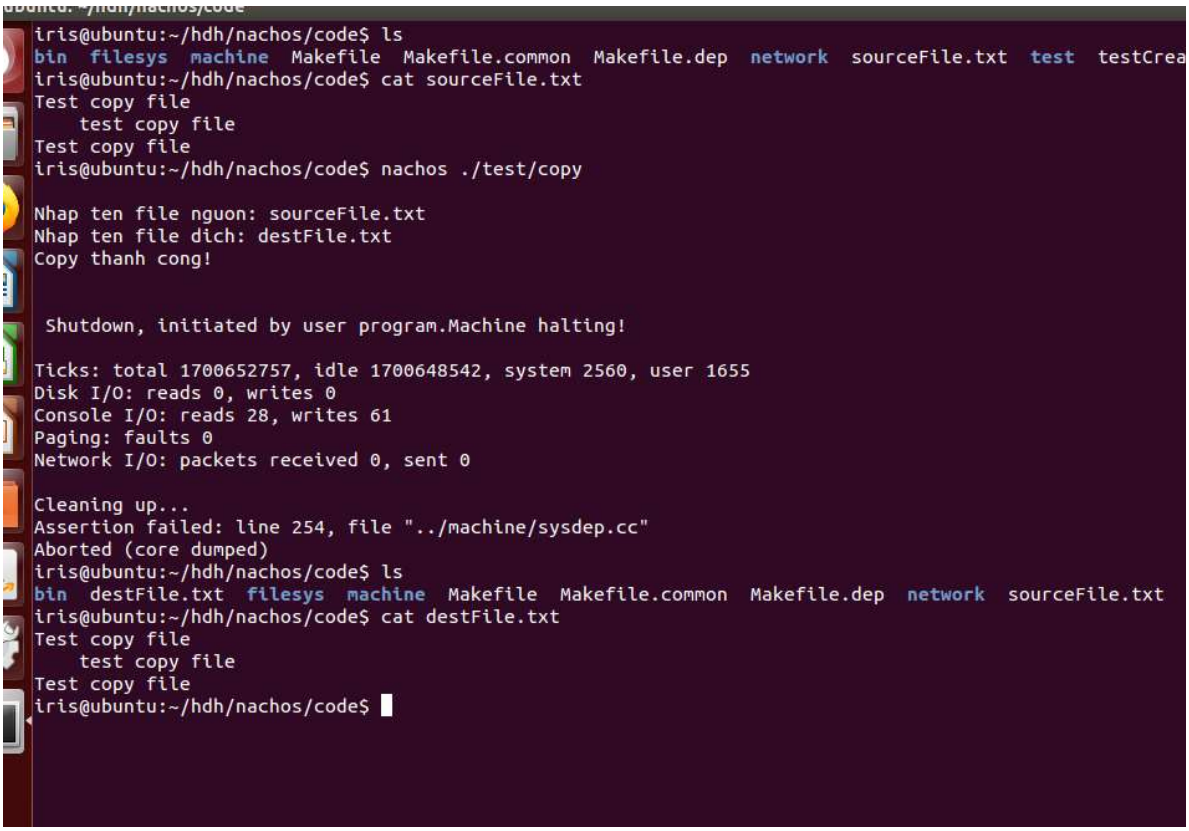
4. Cat

- *Input:* tên file
- *Output:* xuất nội dung của file ra màn hình console



5. Copy

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



C. TỔNG KẾT

I. Đánh giá đồ án

1. Mức độ hoàn thành của các thành viên

MSSV	Mức độ hoàn thành công việc	Đóng góp
18120078	100/100	40%
18120211	100/100	30%
18120227	100/100	30%

2. Mức độ hoàn thành đồ án:

ID TASK	NỘI DUNG	HOÀN THÀNH
Task 1	Viết lại file exception.cc để xử lý tất cả các exceptions được liệt kê trong machine/machine.h	Tốt
Task 2	Viết lại cấu trúc điều khiển của chương trình để nhận các Nachos system calls	Tốt
Task 3	Tất cả các system calls (ko phải Halt) sẽ yêu cầu Nachos tăng program counter trước khi system call trả kết quả về	Tốt
Task 4	Cài đặt system call int CreateFile(char *name)	Tốt
Task 5	Cài đặt system call OpenFileID Open(char *name, int type) và int Close(OpenFileID id)	Tốt
Task 6	Cài đặt system call int Read(char *buffer, int charcount, OpenFileID id) và int Write(char *buffer, int charcount, OpenFileID id)	Tốt
Task 7	Cài đặt system call int Seek(int pos, OpenFileID id)	Tốt
Task 8	Viết chương trình createfile để kiểm tra system call CreateFile	Tốt
Task 9	Viết chương trình echo	Tốt
Task 10	Viết chương trình cat	Tốt
Task 11	Viết chương trình copy	Tốt

Đánh giá: 100%

II. Nguồn tham khảo

- File System trong nachos