

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**  
**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO**

**Đồ án**

**Lập trình NachOS**  
**Exception và các syscall đơn giản**

Chuyên ngành: Khoa học máy tính

GVHD: Phạm Tuấn Sơn

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**  
**KHOA CÔNG NGHỆ THÔNG TIN**



## **BÁO CÁO**

**|Đề tài|**

# **Lập trình NachOS Exception và các syscall đơn giản**

---

**Sinh viên**

---

**Phùng Tiến Hào 1612174**

**Phạm Phong Hào 1612176**

**Trần Thị Trúc Hân 1612170**

**Trần Thị Hồng Nhung 1612476**

**Thành phố Hồ Chí Minh, ngày 11 tháng 11 năm 2018**

# Mục Lục

I.	Phân chia công việc .....	4
II.	Mức độ hoàn thành:.....	4
III.	Giới thiệu .....	4
1.	Tổng quan về Nachos: .....	4
2.	Cơ chế thực hiện, quy trình thực thi của một chương trình trên NachOS:.....	5
3.	Tổng quát cái syscall của đồ án 1: .....	5
IV.	Nội dung đồ án .....	6
1.	Cài đặt tổng quan .....	6
a)	<i>Biên dịch và cài đặt Nachos:</i> .....	6
b)	<i>Các bước cài đặt system calls:</i> .....	7
2.	Cài đặt các system calls và các exception.....	7
a)	<i>Cài đặt các Exception</i> .....	7
b)	<i>Cài đặt biến program counter</i> .....	8
c)	<i>Cài đặt system call int CreateFile(char *name):</i> .....	8
d)	<i>Cài đặt system call OpenFileID Open(char *name, int type) và int Close(OpenFileID id)</i> .....	8
e)	<i>Cài đặt system call int Read(char *buffer, int charcount, OpenFileID id) và int Write(char*buffer, int charcount, OpenFileID id).</i> .....	9
f)	<i>Cài đặt system call int Seek(int pos, OpenFileID id).</i> .....	10
g)	<i>Viết chương trình createfile để kiểm tra system call CreateFile.</i> .....	10
h)	<i>Viết chương trình echo</i> .....	10
i)	<i>Viết chương trình cat</i> .....	10
j)	<i>Viết chương trình copy</i> .....	10
V.	Chạy chương trình: .....	10
1.	CreateFile:.....	11
2.	Echo: .....	11
3.	Cat:.....	11
4.	Copy:.....	12
VI.	Tham Khảo:.....	12

## I. Phân chia công việc

STT	MSSV	Họ và tên	Email	Nội dung
1	1612174	Phùng Tiến Hào	tienhaophung@gmail.com	3, 4, 8
2	1612176	Phạm Phong Hào	phonghao21021998@gmail.com	5, 6, 7
3	1612170	Trần Thị Trúc Hân	tranhan122@gmail.com	1, 2, BC
4	1612476	Trần Thị Hồng Nhung	tthnhung98@gmail.com	9, 10, 11

## II. Mức độ hoàn thành:

STT	Tên công việc	Độ hoàn thành	Ghi chú
1	Cài đặt và biên dịch	100%	Cả nhóm
2	Hoàn thành câu 1	100%	Như phân công
3	Hoàn thành câu 2	100%	Như phân công
4	Hoàn thành câu 3	100%	Như phân công
5	Hoàn thành câu 4	100%	Như phân công
6	Hoàn thành câu 5	100%	Như phân công
7	Hoàn thành câu 6	100%	Như phân công
8	Hoàn thành câu 7	100%	Như phân công
9	Hoàn thành câu 8	100%	Như phân công
10	Hoàn thành câu 9	100%	Như phân công
11	Hoàn thành câu 10	100%	Như phân công
12	Hoàn thành câu 11	100%	Như phân công
13	Kiểm thử	100%	1612174
14	Báo cáo	100%	Như phân công
15	Nộp bài	100%	1612174
16	Họp rút kinh nghiệm	100%	Cả nhóm
17	Giải thích vô code	100%	1612174

## III. Giới thiệu

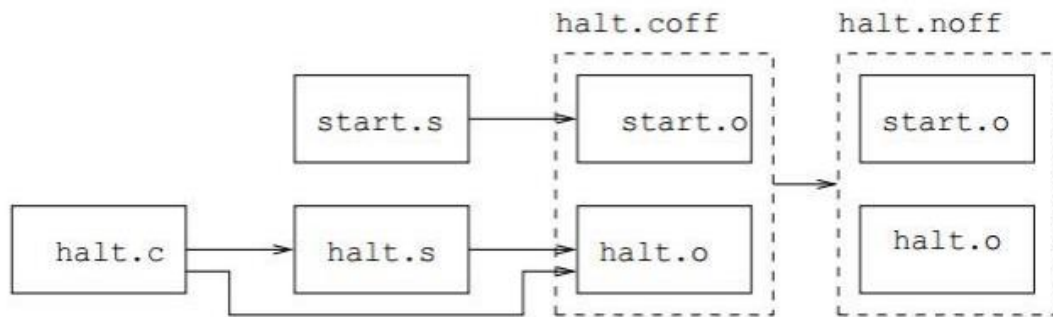
### 1. Tổng quan về Nachos:

NachOS là viết tắt của Not Another Completely Heuristic Operating System, một phần mềm giả lập hệ điều hành với kiến trúc MIPS (Million Instructions Per Second) chạy trên nền tảng linux

## 2. Cơ chế thực hiện, quy trình thực thi của một chương trình trên NachOS:

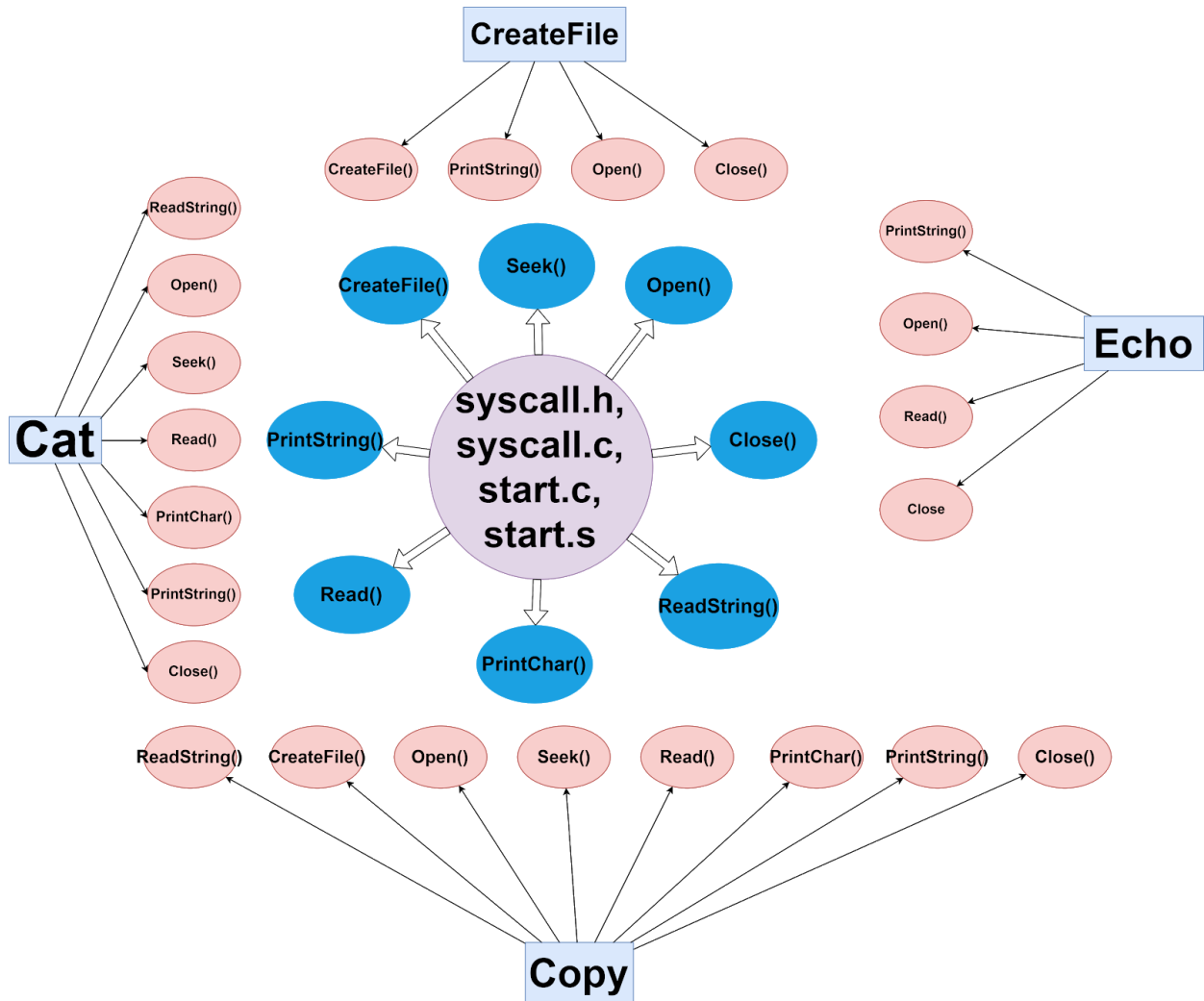
Để một chương trình (VD: halt.c) có thể thực thi nó cần phải được biên dịch. Quá trình biên dịch trên NachOS gồm có ba bước:

- **Bước 1.** Chương trình halt.c được cross-compiler biên dịch thành tập tin halt.s là mã hợp ngữ chạy trên kiến trúc MIPS
- **Bước 2.** Tập tin halt.s được liên kết với starts.s (được coi là thư viện) để tạo thành tập tin halt.coff (bao gồm halt.o và start.o) dạng file thực thi trên Linux với kiến trúc MIPS
- **Bước 3.** Tập tin halt.coff được phần mềm coff2noff chuyển thành tập tin halt.noff đây chính là dạng file thực thi trên NachOS kiến trúc MIPS Quá trình này được mô tả bằng hình dưới đây:



## 3. Tổng quát cái syscall của đồ án 1:

Ta xét sơ đồ sau:



## IV. Nội dung đồ án

### 1. Cài đặt tổng quan

#### a) Biên dịch và cài đặt Nachos:

##### ➤ Bước 1:

Giải nén 2 file nachos.rar và synchcons.rar ta được 2 thư mục nachos và synchcons, trong đó:

- nachos là thư mục chứa mã nguồn hệ điều hành Nachos chạy trên máy ảo được giả lập theo kiến trúc MIPS kèm với cross-compiler (đã biên dịch sẵn) để biên dịch các chương trình C thành các file thực thi có thể chạy được trên hệ điều hành này.
- synchcons là thư mục chứa mã nguồn lớp SynchConsole dùng để điều khiển nhập xuất ra màn hình console theo cơ chế đồng bộ.

##### ➤ Bước 2:

Thiết lập đường dẫn tới cross-compiler cho các mã nguồn do người dùng viết để có thể chạy được trên hệ điều hành Nachos bằng cách chỉnh sửa file Makefile ở thư mục nachos/nachos-3.4/code:

# GCCDIR = [link to cross-compiler directory]

- # CC = \$(GCCDIR)gcc -B[link to cross-compiler directory]
- **Bước 3:**  
 Biên dịch và chạy thử hệ điều hành Nachos bằng cách mở terminal với thư mục hiện hành là nachos/nachos-3.4/code:  
 Cấp quyền thực thi cho toàn bộ các file và thư mục trong hệ điều hành Nachos và cross-compiler: `chmod -R +x ../..`  
 Biên dịch: `gmake all`  
 Chạy thử: `./userprog/nachos -x test/halt`
- b) **Các bước cài đặt system calls:**
  - **Bước 1:**  
`./code/userprog/syscall.h`  
`#define SC_Create 4 // define syscall để dùng trong switch-case`  
`int Create(char* name); // Khai báo prototype của hàm`
  - **Bước 2:**  
`./code/test/start.c` và `./code/test/start.s` thêm dòng  
`.globl Create`  
`.ent`  
`Create Create: addiu $2, $0, SC_Create`  
`syscall`  
`j $31`
  - **Bước 3:**  
`./code/userprog/exception.cc` sửa điều kiện if thành switch.. case
  - **Bước 4:**  
 Viết chương trình ở mức người dùng để kiểm tra file `.c` `./code/test` Sử dụng hàm như đã khai báo prototype ở `./code/userprog/syscall.h`
  - **Bước 5:**  
`./code/test/Makefile`  
 Thêm tên chương trình (tên file) vào dòng all  
`all: halt shell matmult sort (tên file chương trình ở ./test)`  
 Thêm đoạn sau phía sau matmult :  
`<tên file>.o: <tên file>.c`  
`$(CC) $(CFLAGS) -c .c : .o start.o`  
`$(LD) $(LDFLAGS) start.o .o -.coff`  
`../bin/coff2noff .coff`
  - **Bước 6:**  
 Biên dịch lại nachos, cd tới `./nachos/code` chạy lên “gmake all”.
  - **Bước 7:**  
 Chạy thử chương trình: `./userprog/nachos -rs 1023 -x ./test/`

## 2. Cài đặt các system calls và các exception

### a) Cài đặt các Exception

- Viết lại file `exception.cc` để xử lý tất cả các exceptions được liệt kê trong `machine/machine.h`.

- Vào thư mục ./code/machine vào file “machine.h” ta có danh sách các exception được liệt kê, ta qua file exception.cc viết lại các case này theo các ExceptionType mà tắt bắt được. Mỗi exception ta thêm lệnh interrupt->Halt() để tắt hệ điều hành.
- b) Cài đặt biến program counter**
  - Để tăng Program counter cho việc nạp lệnh tiếp theo. Ta cần lưu giá trị của thanh ghi PC hiện tại (current PC) vào PC trước (previous PC), nạp giá trị của PC kế tiếp (next PC) vào current PC và cuối cùng là nạp giá trị kế tiếp nữa cho next PC.
- c) Cài đặt system call int CreateFile(char \*name):**  
CreateFile(char \*name)
  - Input: Địa chỉ chứa tên file trong User space
  - Output: Trả về 0 nếu thành công và ngược lại: -1
  - Purpose: Tạo ra 1 file rỗng với tham số tên file là name
  - Sử dụng Nachos fileSystem để tạo file rỗng. Vùng nhớ của biến name ban đầu nằm ở User space. Do đó, ta cần chuyển đổi vùng nhớ ấy sang System space.
  - Trước tiên, ta đọc địa chỉ của biến name trong thanh ghi r4 và chuyển đổi giá trị từ User space sang System space bằng hàm User2System(). Ta sẽ thu được chuỗi filename trả về, đây chính là tên file. Sau đó, ta sẽ kiểm tra xem chuỗi filename có rỗng hay phương thức Create của fileSystem có thành công hay không. Nếu thành công thì trả về kết quả vào thanh ghi r2, 0 nếu thành công và 1 ngược lại.
- d) Cài đặt system call OpenFileID Open(char \*name, int type) và int Close(OpenFileID id)**
  - File có hai loại:
    - File chỉ có thể đọc.
    - File có thể đọc và ghi
  - Trong đó, ta quy định như sau: Mỗi tiến trình sẽ được cấp một bảng mô tả file với kích thước cố định. Kích thước của bảng mô tả file 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 (theo đề bài quy định).
  - Ta quy ước các trường hợp như sau về việc mở file, ta dựa vào biến type
    - type =0: đọc và ghi.
    - type =1: chỉ đọc.
    - type =2: stdin.
    - type =3: stdout.
  - Để viết được 2 syscall này ta cần hai syscall hỗ trợ là syscall SC\_Open và SC\_Close
  - SC\_Open:
    - Input: Địa chỉ của tên file trên user space, chế độ mở (type).
    - Output: id file nếu thành công, -1 nếu lỗi.
  - SC\_Close:
    - Input: ID file.
    - Output: NULL.



**e) 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, cần phân biệt giữa Console IO (OpenFileID 0, 1) và File.
- Cách làm việc: Phần console read và write sẽ sử dụng lớp SynchConsole, khởi tạo qua biến toàn cục gSynchConsole. Sử dụng các hàm mặc định của SynchConsole để đọc và ghi. 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. 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 về -1 nếu bị lỗi và -2 nếu cuối file.

❖ **int Read (char\* buffer, int charcount, OpenFile id).**

Mô tả:

- Input: Buffer, số ký tự cho phép, id của file.
- Output: -1 nếu lỗi, -2 nếu thành công với số byte được đọc.
- Mục đích: Đọc file với tham số là buffer, số ký tự cho phép (charcount) và id của file. Đọc địa chỉ của tham số buffer từ thanh ghi r4, tham số charcount từ thanh ghi r5 và id của file từ thanh ghi r6, sau đó ta tiến hành kiểm tra id của file truyền vào có nằm ngoài bảng mô tả file không, file cần đọc có tồn tại không và file cần đọc có phải là stdout với type = 3 không. Nếu vi phạm các điều kiện trên thì trả về -1 cho thanh ghi r2 ngược lại thì lấy vị trí con trỏ ban đầu trong file (dùng phương thức GetCurrentPos() của lớp FileSystem gọi là OldPos) và thực hiện chép giá trị ở r4 từ phía User sang System bằng hàm User2System().

Trường hợp type = 2, ta gọi phương thức Read của lớp SynchConsole đọc buffer với độ dài charcount, trả về số byte thực sự đọc được cho thanh ghi r2 và chép buffer từ System sang User bằng hàm System2User().

Trường hợp đọc file bình thường, lấy vị trí con trỏ hiện tại trong file bằng phương thức GetCurrentPos() của lớp FileSystem, trả về số byte thực sự đọc được cho thanh ghi r2 bằng công thức: NewPos – OldPos và chép buffer từ System sang User bằng hàm System2User().

Trường hợp còn lại là đọc file rỗng thì trả về -2 cho thanh ghi r2.

❖ **int Write (char\* buffer, int charcount, OpenFileID id).**

Mô tả:

- Input: Buffer, số ký tự cho phép, id của file.
- Output: -1 nếu lỗi, hoặc số byte thực sự ghi được nếu thành công.
- Mục đích: Ghi file với tham số là buffer, số ký tự cho phép (charcount) và id của file. Ta đọc địa chỉ của buffer từ thanh ghi r4, tham số charcount từ thanh ghi r5 và id của file từ thanh ghi r6, sau đó ta tiến hành kiểm tra id của file truyền vào có nằm ngoài bảng mô tả file không, file cần ghi có tồn tại không và file cần ghi có phải là stdin với type = 2 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 thì lấy vị trí con trỏ ban đầu trong file bằng phương thức GetCurrentPos() của lớp FileSystem gọi là OldPos và thực hiện chép giá trị ở r4 từ User sang System bằng hàm User2System().

Trường hợp ghi file đọc và ghi với type = 0, thì ta lấy vị trí con trỏ hiện tại trong file bằng phương thức GetCurrentPos() của lớp FileSystem gọi là NewPos, trả về số byte thực sự ghi được cho thanh ghi r2 bằng công thức: NewPos – OldPos.

Trường hợp ghi file stdout với type = 3, ta gọi phương thức Write của lớp SynchConsole để ghi từng ký tự trong buffer và kết thúc là ký tự xuống dòng '\n', trả về số byte thực sự ghi được cho thanh ghi r2.

**f) Cài đặt system call int Seek(int pos, OpenFileID id).**

Pos mang ý nghĩa vị trí cần chuyển tới, nếu pos = -1 thì di chuyển đến cuối file. Hàm trả về vị trí thực sự trong file nếu thành công hoặc -1 nếu có lỗi.

Mô tả:

- Input: Vị trí cần chuyển tới, id của file.
- Output: -1: Lỗi

Vị trí thực sự trong file: Thành công.

▪ Mục đích: Di chuyển con trỏ đến vị trí thích hợp trong file với tham số đầu vào là vị trí cần dịch chuyển và id của file. Đọc vị trí cần chuyển tới từ thanh ghi r4, id của file từ thanh ghi r5, sau đó ta tiến hành kiểm tra id của file truyền vào có nằm ngoài bảng mô tả file không, file có tồn tại không và 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 nếu pos = -1 thì gán pos bằng độ dài của file (dùng 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.

**g) Viết chương trình createfile để kiểm tra system call CreateFile.**

Dùng tên file do người dùng nhập vào từ console.

Gọi system call Open để mở file stdin với type = 2. Nếu mở file thành công thì gọi system call Read đọc tên file vừa nhận và gọi system call CreateFile để tạo file với tham số truyền vào là tên file. Kết thúc là đóng file stdin với system call Close.

**h) Viết chương trình echo**

Chương trình echo xuất lại chuỗi người dùng nhập từ console

Sử dụng syscall ReadString cho người dùng nhập vào chuỗi từ bàn phím, đẩy xuống lưu trong vùng nhớ. Sau đó sử dụng syscall PrintString để lấy dữ liệu ra khỏi vùng nhớ và xuất lên màn hình console:

**i) Viết chương trình cat**

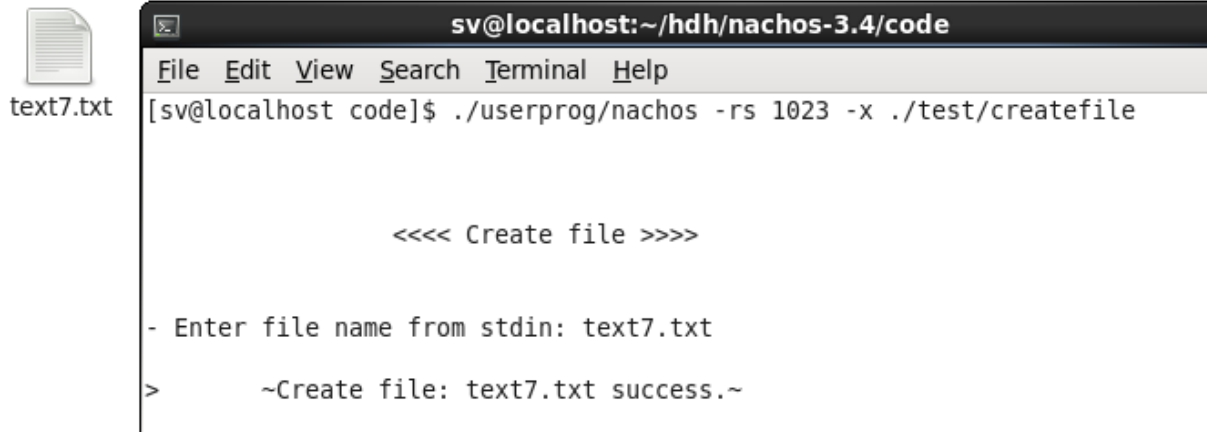
Dùng syscall ReadString cho người dùng nhập vào tên file cần hiển thị, đẩy tên file xuống lưu vào trong vùng nhớ. Lấy tên file đưa vào syscall Open với type = 1 (chỉ đọc). Trong khi còn đọc không bị lỗi và chưa đến cuối file thì đọc file với độ dài maxlength và xuất ra console.

**j) Viết chương trình copy**

Dùng syscall ReadString cho người dùng nhập vào tên file nguồn và đích, đẩy tên file xuống lưu vào trong vùng nhớ. Lấy tên file đưa vào syscall Open với type của file nguồn là 1 (tránh bị thay đổi nội dung file nguồn) và file đích với type = 0. Trong khi còn đọc không bị lỗi và chưa đến cuối file thì đọc file nguồn với độ dài maxlength và ghi vào file đích:

## V. Chạy chương trình:

### 1. CreateFile:



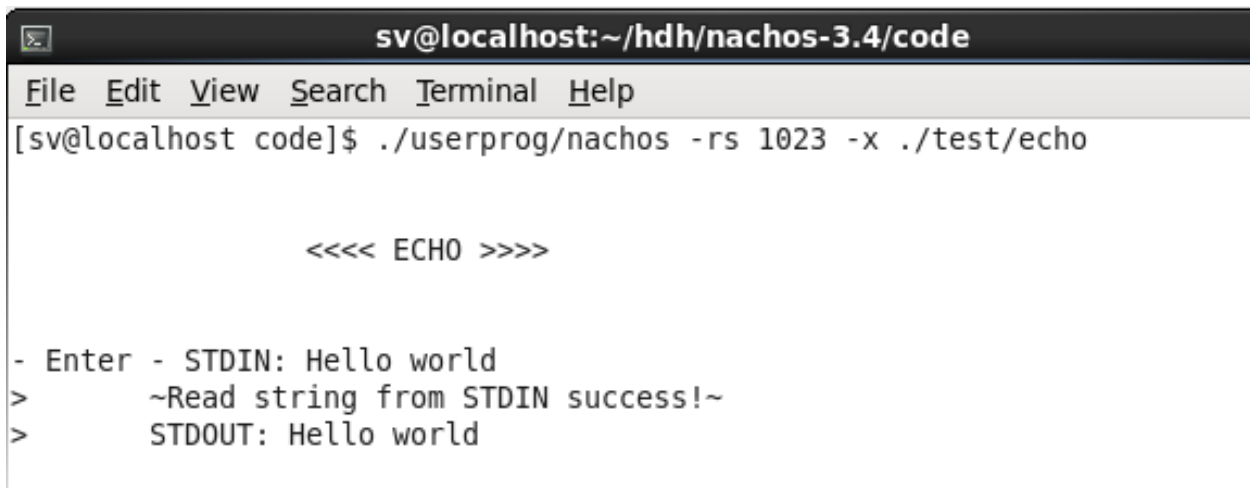
```

sv@localhost:~/hdh/nachos-3.4/code
File Edit View Search Terminal Help
[sv@localhost code]$ ./userprog/nachos -rs 1023 -x ./test/createfile

<<<< Create file >>>>

- Enter file name from stdin: text7.txt
> ~Create file: text7.txt success.~
    
```

### 2. Echo:



```

sv@localhost:~/hdh/nachos-3.4/code
File Edit View Search Terminal Help
[sv@localhost code]$ ./userprog/nachos -rs 1023 -x ./test/echo

<<<< ECHO >>>>

- Enter - STDIN: Hello world
> ~Read string from STDIN success!~
> STDOUT: Hello world
    
```

### 3. Cat:

The screenshot shows a text editor window with a file named `text1.txt` containing a Vietnamese poem. Below it, a terminal window titled `sv@localhost:~/hdh/nachos-3.4/code` displays the output of the command `./userprog/nachos -rs 1023 -x ./test/cat`. The terminal output shows the contents of `text1.txt` being printed, followed by a prompt to continue.

```

File Edit View Search Terminal Help
Ticks: total 513120321, idle 513118817, system 1400, user 104
Disk I/O: reads 0, writes 0
Console I/O: reads 10, writes 104
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
[sv@localhost code]$ ./userprog/nachos -rs 1023 -x ./test/cat
- Input filename: text1.txt
> ~Content of file:
Anh chôi voi, noi gac pho quen minh tung,
Tung la tat ca.
Hom ay may xanh ngat troi.
Co giu lay chut am ap tu tay em
Nhưng biet sao duoc
Hom nay minh chia tay.
Nhưng tam hinh ky niem cua chung minh
Van day ma
Vay co sao em voi xa?
Nhưng mon qua cung loi chuc, yeu anh thiet tha.
Chỉ moi anh la nho em cua ngay hom qua.

- Do you wanna continue? (Y/N)
  
```

#### 4. Copy:

The screenshot shows two text editor windows side-by-side. The left window, `text1.txt`, contains the same Vietnamese poem as before. The right window, `text7.txt`, contains the same text. Below them, a terminal window titled `sv@localhost:~/hdh/nachos-3.4/code` displays the output of the command `./userprog/nachos -rs 1023 -x ./test/copy`. The terminal output shows the source file `text1.txt` and destination file `text7.txt` being specified, followed by a success message.

```

File Edit View Search Tools Documents Help
File Edit View Search Tools Documents Help

Cleaning up...
[sv@localhost code]$ ./userprog/nachos -rs 1023 -x ./test/copy
- Enter source file's name: text1.txt
- Enter destination file's name to copy into it: text7.txt
> ~Copied successfully!~
  
```

## VI. Tham Khảo:

- 4 file pdf của thầy:  
[https://drive.google.com/open?id=1gKJWq99PWYiAA8Us5Lk\\_UCbRFXd6Nj0C](https://drive.google.com/open?id=1gKJWq99PWYiAA8Us5Lk_UCbRFXd6Nj0C)
- Tài liệu hướng dẫn Nachos căn bản:  
[http://read.pudn.com/downloads161/ebook/733633/Nachos\\_CanBan/Nachos\\_CanBan.pdf](http://read.pudn.com/downloads161/ebook/733633/Nachos_CanBan/Nachos_CanBan.pdf)