

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



20127662 - NGUYỄN ĐÌNH VĂN

20127469 - PHẠM MINH ĐỨC

20127061 - LƯU MINH PHÁT

BÁO CÁO ĐỒ ÁN

HỆ ĐIỀU HÀNH

Thành phố Hồ Chí Minh – 2022

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



20127662 - NGUYỄN ĐÌNH VĂN

20127469 - PHẠM MINH ĐỨC

20127061 - LƯU MINH PHÁT

BÁO CÁO ĐỒ ÁN 1

|Đề tài|

Exceptions và các system calls đơn giản

|Giáo viên hướng dẫn|

Gv.Lê Viết Long

Gv.Phạm Tuấn Sơn

Hệ Điều Hành

Thành phố Hồ Chí Minh - 2022

Mục lục

1. THÔNG TIN THÀNH VIÊN:	2
1.1. Thông tin	2
1.2. Mức độ hoàn thành	3
2. CÀI ĐẶT CHUNG	4
2.1. Cài đặt trước	4
2.2. Các giá trị thanh ghi	4
2.3. Các bước cài đặt system call	4
3. SYSTEM CALL VÀ EXCEPTION	6
3.1. Cài đặt lại các exception:	6
3.2. Cài đặt syscall CreateFile: int CreateFile(char * name)	6
3.3. Cài đặt System Call: OpenFileID Open(char *name, int type) và int Close(OpenFileID id)...	6
3.4. Cài đặt System Call: int Read (char* buffer, int charcount, OpenFileID id) và int Write (char* buffer, int charcount, OpenFileID id)	7
3.5. Cài đặt System Call: int Seek (int pos, OpenFileID id)	8
3.6. Cài đặt system call int Delete(char *name).	9
4. DEMO CHƯƠNG TRÌNH:	10
4.1 Chương trình createfile	10
4.2 Chương trình echo	10
4.3 Chương trình cat	11
4.4 Chương trình copy	11
4.5 Chương trình delete	12
5. MÔI TRƯỜNG LẬP TRÌNH:	13
6. BẢNG PHÂN CÔNG CÔNG VIỆC CỦA THÀNH VIÊN:	13
7. TÀI LIỆU THAM KHẢO:	13

1. THÔNG TIN THÀNH VIÊN:

1.1. Thông tin

Họ và tên	MSSV	Email
Lưu Minh Phát	20127061	20127061@student.hcmus.edu.vn
Nguyễn Đình Văn	20127662	20127662@student.hcmus.edu.vn
Phạm Minh Đức	20127469	20127469@student.hcmus.edu.vn

1.2. Mức độ hoàn thành

	CHỨC NĂNG	MỨC ĐỘ HOÀN THÀNH
A. Hệ điều hành NachOS	1. Viết lại file exception.cc	100%
	2. Viết lại cấu trúc điều khiển của chương trình để nhận các Nachos system calls.	100%
	3. Tất cả các system calls (ko phải Halt) sẽ yêu cầu Nachos tăng program counter trước khi system call trả kết quả về.	100%
	4. Cài đặt system call int CreateFile(char *name)	100%
	5. Cài đặt system call OpenFileID Open(char *name, int type) và int Close(OpenFileID id)	100%
	6. Cài đặt system call int Read(char *buffer, int charcount, OpenFileID id) và int Write(char *buffer, int charcount, OpenFileID id)	100%
	7. Cài đặt system call int Seek(int pos, OpenFileID id).	100%
	8. Cài đặt system call int Delete(char *name).	100%
B. Chương trình ứng dụng chạy trên HĐH NachOS	1. Viết chương trình createfile để kiểm tra system call CreateFile	100%
	2. Viết chương trình echo	100%
	3. Viết chương trình cat	100%
	4. Viết chương trình copy	100%
	5. Viết chương trình delete để kiểm tra system call Delete.	100%

2. CÀI ĐẶT CHUNG

2.1. Cài đặt trước

- Ở `../code/threads/` trong file **system.h** và **system.cc** thực hiện khai báo, cấp phát, và xóa vùng nhớ cấp phát một biến toàn cục thuộc lớp “**SynchConsole**” để hỗ trợ việc nhập xuất với màn hình console.
- Ở `../code/userprog` trong file **exception.cc** thực hiện cài đặt hai hàm **char* User2System(int virtAddr, int limit)** để sao chép vùng nhớ từ user sang system và hàm **int System2User(int virtAddr, int len, char* buffer)** để sao chép vùng nhớ từ system về cho user.

2.2. Các giá trị thanh ghi

- R2: Lưu mã syscall đồng thời lưu kết quả trả về của mỗi syscall nếu có.
- R4: Lưu tham số thứ nhất.
- R5: Lưu tham số thứ hai.
- R6: Lưu tham số thứ ba.
- R7: Lưu tham số thứ tư.
- R8: Lưu tham số thứ năm

2.3. Các bước cài đặt system call

- **Bước 1:** `../code/userprog/syscall.h`
 - o `#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
```

.end Create

- **Bước 3:** ./code/userprog/exception.cc sửa điều kiện if thành switch.. case (chỉ sửa một lần duy nhất, lần sau cứ theo format sẵn mà làm cho từng case system call)

- **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 <tên file>.c

<tên file>: <tên file>.o start.o

\$(LD) \$(LDFLAGS) start.o <tên file>.o -<tên file>.coff

../bin/coff2nooff <tên file>.coff <tên file>

- **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/<tên chương trình>.

3. SYSTEM CALL VÀ EXCEPTION

3.1. Cài đặt lại các exception:

B1: Nachos-3.4/code/machine → file “machine.h” lấy danh sách các exception

B2: Nachos-3.4/code/userprog/ → file “exception.cc”

B3: Viết lại các case exception này theo các ExceptionType mà máy bắt được. Mỗi exception ta thêm lệnh interrupt->Halt() để tắt hệ điều hành.

3.2. Cài đặt syscall CreateFile: `int CreateFile(char * name)`

a. Mô tả cài đặt SC_CreateFile:

- Input: Địa chỉ chứa tên file ở User Space
- Output: -1 lỗi, 0 thành công.
- Mục đích: tạo ra file rỗng với tham số là tên file.

b. Ta đọc địa chỉ của tham số name từ thanh ghi r4, sau đó thực hiện chép giá trị ở r4 từ vùng nhớ User sang System bằng hàm User2System(). Giá trị chép được thực sự chính là tên file. Ta tiếp tục kiểm tra tên file có NULL không và file có được tạo ra với tên file đó không. Nếu thành công thì trả về 0, ngược lại thì trả về -1 vào thanh ghi r2. Kết thúc.

3.3. Cài đặt System Call: `OpenFileID Open(char *name, int type)` và `int Close(OpenFileID id)`

a. Quy ước giá trị của type:

- Type =0: đọc và ghi.
- Type =1: chỉ đọc.
- Type =2: stdin.
- Type =3: stdout.

b. Mô tả cài đặt SC_Open:

- Input: Địa chỉ của tên file trên user space, chế độ mở (type).
- Output: id file nếu thành công, -1 nếu lỗi.
- Mục đích: mở một file với tham số số truyền vào gồm tên file và cách mở file.

c. Ta đọc địa chỉ của tham số name từ thanh ghi r4 và tham số type từ thanh ghi r5 sau đó kiểm tra type có hợp lệ hay không ($0 \leq \text{type} \leq 3$), kiểm tra index của file trong lớp FileSystem có nằm trong bảng mô tả file không – mỗi tiến trình Read/Write sẽ được cấp một bảng đặc tả file có kích thước là 10 - ($0 \leq \text{index} \leq 9$). Nếu kiểm tra hai điều kiện trên hợp lệ thì thực hiện

chép giá trị ở r4 từ phía User sang System bằng hàm User2System(). Giá trị chép được thực sự chính là tên file. Ta tiếp tục kiểm tra tên file, nếu là stdin và type truyền vào là 2 thì trả về cho thanh ghi r2 id của file là 0, nếu là stdout và type truyền vào là 3 thì trả về cho thanh ghi r2 id của file là 1, nếu là file bình thường thì type phải khác 2 và 3 và trả về cho thanh ghi r2 đúng id của file bằng (index- 1) của lớp FileSystem. Trường hợp mở file không tồn tại hoặc ngược lại với tất cả điều kiện trên thì trả về -1 cho thanh ghi r2. Kết thúc.

d. Mô tả cài đặt SC_Close:

- Input: ID file.
- Output: NULL.
- Mục đích: Đóng file với tham số truyền vào là ID của file.

e. Đọc tham số id của file từ thanh ghi r4, sau đó kiểm tra xem file cần đóng có tồn tại không bằng openf[id] đã cài đặt trong lớp FileSystem và kiểm tra id của file có nằm ngoài bảng mô tả file không. Nếu có một trong hai lỗi trên thì xuất ra thông báo lỗi và gọi syscall Halt() để tắt hệ thống, nếu thành công thì xóa đi dữ liệu openf[id] và gán lại bằng NULL. Kết thúc.

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

a. Mô tả cài đặt SC_Read:

- 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.

b. Ta 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 là hợp lệ 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 phía User sang System bằng hàm User2System(). Giá trị chép được là buffer chứa chuỗi ký tự. Xét trường hợp đọc file stdin với 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 phía System sang User bằng hàm System2User(). Xét trường hợp đọc file bình thường, 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 đọc được cho thanh ghi r2 bằng công thức: $\text{NewPos} - \text{OldPos}$ và cũng chép buffer từ phía System sang User bằng hàm System2User(). Trường hợp còn lại là đọc file rỗng thì trả về -2 cho thanh ghi r2. Kết thúc.

c. Mô tả cài đặt SC_Write:

- Input: Buffer, số ký tự cho phép, id của file.
- Output: -1 nếu lỗi, 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.

d. Ta 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 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 là hợp lệ 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 phía User sang System bằng hàm User2System(). Giá trị chép được là buffer chứa chuỗi ký tự. Xét 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: $\text{NewPos} - \text{OldPos}$. Xét 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. Kết thúc.

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

a. Mô tả cài đặt SC_Seek:

- 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 là vị trí cần dịch chuyển và id của file.

b. Ta đọc tham số pos từ thanh ghi r4 và 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 cần di chuyển con trỏ có tồn tại không và kiểm tra 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 kiểm tra nếu pos = -1 thì gán pos bằng độ dài của file bằng 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. Kết thúc.

3.6. Cài đặt system call int Delete(char *name).

a. Mô tả cài đặt SC_Delete:

- Input: Địa chỉ vùng nhớ user của file
- Output: -1 = Lỗi, 0 = Thành công
- Mục đích: Tạo ra file với tham số là tên file

- b. Đọc địa chỉ của file từ thanh ghi R4. Sau đó ta thực hiện sao chép không gian bộ nhớ User sang System, với độ dài tối đa là $(32 + 1)$ bytes. Nếu độ dài của filename = 0 thì trả -1 vào thanh ghi r2, đẩy thanh ghi về sau. Còn nếu filename không đọc được thì cũng trả về -1 cho thanh ghi r2, sau đó xóa filename nhập vào. Nếu đọc thành công thì thực hiện gọi hàm Delete xóa filename với type = 0, nếu không tạo được file -> không xóa được file thì trả -1 vào thanh ghi r2, đẩy thanh ghi về sau, xóa filename và ngược lại nếu tạo thành công thì trả về 0 cho thanh ghi r2, xóa filename, thực hiện đẩy thanh ghi về sau. Kết thúc.

4. DEMO CHƯƠNG TRÌNH:

4.1 Chương trình createfile

```
vwan@vwan-VirtualBox: ~/Desktop/HDH/nachos/nachos-3.4/code_final_
vwan@vwan-VirtualBox:~/Desktop/HDH/nachos/nachos-3.4/code_final_$ ./userprog/nachos -rs 1023 -x ./test/Createfile

#####- CREATE FILE -#####

ENTER NAME OF FILE: newfile.txt

CREATE FILE DONE!!

Shutdown, initiated by user program. Machine halting!

Ticks: total 1278613054, idle 1278611893, system 1100, user 61
Disk I/O: reads 0, writes 0
Console I/O: reads 12, writes 76
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
vwan@vwan-VirtualBox:~/Desktop/HDH/nachos/nachos-3.4/code_final_$
```

4.2 Chương trình echo

```
vwan@vwan-VirtualBox: ~/Desktop/HDH/nachos/nachos-3.4/code_final_
vwan@vwan-VirtualBox:~/Desktop/HDH/nachos/nachos-3.4/code_final_$ clear
vwan@vwan-VirtualBox:~/Desktop/HDH/nachos/nachos-3.4/code_final_$ ./userprog/nachos -rs 1023 -x ./test/Echo
#####- ECHO -#####

Enter: DO AN NACHOS HCMUS
Echo: DO AN NACHOS HCMUS

Shutdown, initiated by user program. Machine halting!

Ticks: total 610184369, idle 610183262, system 1040, user 67
Disk I/O: reads 0, writes 0
Console I/O: reads 19, writes 62
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
vwan@vwan-VirtualBox:~/Desktop/HDH/nachos/nachos-3.4/code_final_$
```


4.3 Chương trình cat

```
vwan@vwan-VirtualBox: ~/Desktop/HDH/nachos/nachos-3.4/code_final_
vwan@vwan-VirtualBox:~/Desktop/HDH/nachos/nachos-3.4/code_final_$ ./userprog/nachos -rs
1023 -x ./test/Cat

#####- CAT FILE -#####

Enter name of file: ngamtrang.txt
-> Contents file:

Trong tù không rượu cũng không hoa,
Cảnh đẹp đêm nay khó hững hờ.
Người ngắm trăng soi ngoài cửa sổ,
Trăng nhòm khe cửa ngắm nhà thơ.

Shutdown, initiated by user program. Machine halting!

Ticks: total 825836678, idle 825824082, system 7450, user 5146
Disk I/O: reads 0, writes 0
Console I/O: reads 14, writes 247
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
vwan@vwan-VirtualBox:~/Desktop/HDH/nachos/nachos-3.4/code_final_$
```

4.4 Chương trình copy

```
vwan@vwan-VirtualBox: ~/Desktop/HDH/nachos/nachos-3.4/code_final_
vwan@vwan-VirtualBox:~/Desktop/HDH/nachos/nachos-3.4/code_final_$ ./userprog/nachos -rs 1023 -x ./test/Copy

#####- COPY FILE -#####

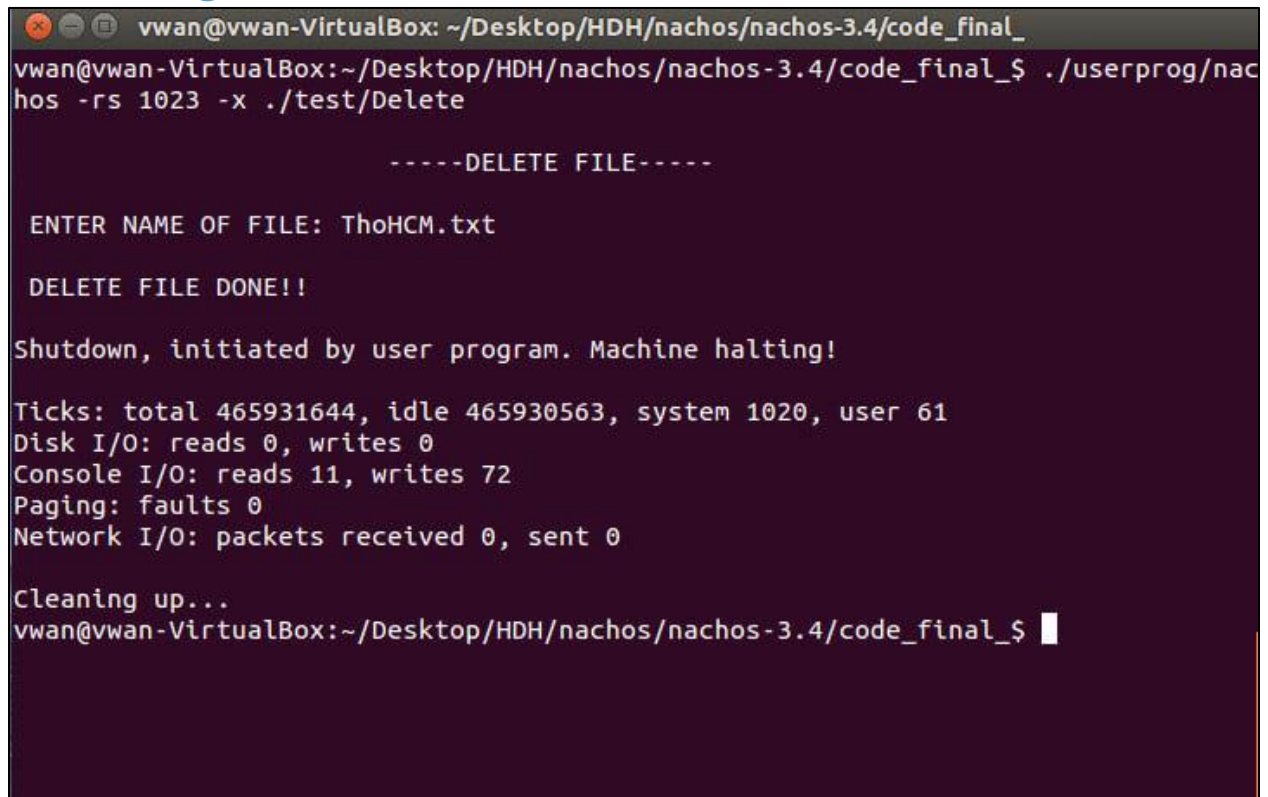
Enter source file: ngamtrang.txt
Enter destination file: ThoHCM.txt
-> Copy successful.

Shutdown, initiated by user program. Machine halting!

Ticks: total 1024590888, idle 1024583154, system 2180, user 5554
Disk I/O: reads 0, writes 0
Console I/O: reads 25, writes 100
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
vwan@vwan-VirtualBox:~/Desktop/HDH/nachos/nachos-3.4/code_final_$
```

4.5 Chương trình delete



```
vwan@vwan-VirtualBox: ~/Desktop/HDH/nachos/nachos-3.4/code_final_  
vwan@vwan-VirtualBox:~/Desktop/HDH/nachos/nachos-3.4/code_final_$ ./userprog/nac  
hos -rs 1023 -x ./test/Delete  
  
-----DELETE FILE-----  
  
ENTER NAME OF FILE: ThoHCM.txt  
  
DELETE FILE DONE!!  
  
Shutdown, initiated by user program. Machine halting!  
  
Ticks: total 465931644, idle 465930563, system 1020, user 61  
Disk I/O: reads 0, writes 0  
Console I/O: reads 11, writes 72  
Paging: faults 0  
Network I/O: packets received 0, sent 0  
  
Cleaning up...  
vwan@vwan-VirtualBox:~/Desktop/HDH/nachos/nachos-3.4/code_final_$
```

5.MÔI TRƯỜNG LẬP TRÌNH:

- Ngôn ngữ lập trình: c, c++
- Môi trường giả lập: Oracle VM VirtualBox
- Hệ điều hành: Ubuntu (32-bit) Nachos-3.4

6. BẢNG PHÂN CÔNG CÔNG VIỆC CỦA THÀNH VIÊN:

Chức năng	Thành Viên
A.1,2,3,6,7	Lưu Minh Phát
A. 4,5 B. 1,2,3	Nguyễn Đình Văn
A. 8 B. 4,5 Viết báo cáo	Phạm Minh Đức

7. TÀI LIỆU THAM KHẢO:

- Tạo HĐH ảo và thiết lập nachos cơ bản:
https://www.youtube.com/watch?v=t0jtY1C129s&list=PLRgTVtca98hUgCN2_2vzsAAXPiTFbvHpO&index=3
<https://www.youtube.com/watch?v=P5dqm6Pnq00>
- Tham khảo code và trình bày:
<https://github.com/nguyenthanhchungfit/Nachos-Programing-HCMUS/tree/master/project/nachos-3.4/code>
- Tài liệu tham khảo thông hiểu nachos cơ bản :
Moddle HCMUS – Hệ Điều Hành