

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



MÔN: HỆ ĐIỀU HÀNH
PROJECT 02
TÌM HIỂU VỀ CÁC SYSTEM CALLS - CÁC THAO TÁC VỚI FILE



THÀNH PHỐ HỒ CHÍ MINH – 13/11/2022

Mục lục

I.	THÔNG TIN THÀNH VIÊN.....	2
II.	PHÂN CÔNG CÔNG VIỆC.....	2
III.	CÀI ĐẶT CÁC SYSTEM CALLS - CÁC THAO TÁC VỚI FILE	2
1.	Phần 1.....	2
a.	Create	2
b.	Open	2
c.	Close.....	3
d.	Read.....	3
e.	Write.....	4
f.	Seek.....	4
g.	Remove	4
h.	Sao chép vào vùng nhớ kernel-user.....	5
2.	Phần 2.....	5
a.	Createfile: Tạo file	5
b.	Cat:	5
c.	Copy	5
d.	Delete	6
e.	Concatenate:	6
IV.	DEMO CHƯƠNG TRÌNH.....	6
a.	Createfile: Tạo file	6
b.	Cat:	7
c.	Copy	8
d.	Delete	9
e.	Concatenate:	10

I. THÔNG TIN THÀNH VIÊN

MSSV	HỌ VÀ TÊN
20120468	NGUYỄN VĂN HẢI
20120482	NGUYỄN TẠ HUY HOÀNG
20120532	NGUYỄN NHẬT NAM

II. PHÂN CÔNG CÔNG VIỆC

Người thực hiện	Phân công	Mức độ hoàn thành
Nguyễn Văn Hải	1.1 , 1.2 , 1.3 , 2.1	100%
Nguyễn Tạ Huy Hoàng	1.4 , 1.5 , 1.6 , 2.2 , 2.3	100%
Nguyễn Nhật Nam	1.7 , 1.8 , 2.4 , 2.5 , Báo cáo	100%

III. CÀI ĐẶT CÁC SYSTEM CALLS - CÁC THAO TÁC VỚI FILE

1. Phần 1

a. Create

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

Ý tưởng:

- Đọc địa chỉ file từ thanh ghi 4
- Chuyển con trỏ từ vùng nhớ user space tới vùng nhớ system space
- Nếu không tạo được file, ghi -1 vào thanh ghi 2
- Tạo file thành công, ghi 0 vào thanh ghi 2

b. Open

- Input: arg1 (địa chỉ của chuỗi name).
- Output: Trả về OpenFileID nếu thành công và -1 nếu gặp lỗi.
- Chức năng: Trả về ID của file.

Ý tưởng: Viết thêm hàm Open(char* fileName, int type) và cài đặt các biến liên quan vào lớp fileSystem.

- Lấy địa chỉ của tên file từ thanh ghi 4
- Lấy phương thức mở file từ thanh ghi 5
 - type=0: đọc và ghi
 - type=1: chỉ đọc

- type=2: nhập vào từ console (stdin)
- type=3: xuất ra console (stdout)
- Chuyển con trỏ từ vùng nhớ user space tới vùng nhớ system space
- Xử lý mở file và ghi kết quả vào thanh ghi 2:
 - -1: không mở được file
 - 0: xử lý stdin
 - 1: xử lý stdout
 - $n \geq 2$: địa chỉ ô nhớ còn trống

c. Close

Input: ID của file (OpenFileID).

Output: 1 nếu thành công, 0 nếu thất bại.

Ý tưởng:

- Lấy id của file từ thanh ghi 4
- Xử lý đóng file và ghi kết quả vào thanh ghi 2

d. Read

- Input: buffer(char*), số ký tự(int), id của file(OpenFileID)
- Output: -1: Lỗi, Số byte read thực sự: Thành công, -2: Thành công
- Chức năng: Đọc file với tham số là buffer, số ký tự cho phép và ID của file.

Ý tưởng: Viết thêm hàm GetString(char *buffer, int size) vào lớp synchconsole, dùng để đọc chuỗi kí tự.

Cài đặt:

- Lấy địa chỉ của buffer từ thanh ghi số 4
 - Lấy số lượng kí tự từ thanh ghi số 5
 - Lấy id của file từ thanh ghi số 6
 - Kiểm tra tính hợp lệ của file. Ko hợp lệ -> ghi -1 vào thanh ghi 2
 - Xử lý đọc file với các trường hợp:
 - Đọc file từ stdin: dùng hàm GetString vừa được cài đặt trong lớp synchconsole để đọc chuỗi kí tự từ console.
 - Đọc file bình thường: Dùng hàm Read trong lớp FileSystem để đọc file => Ghi số byte thực sự đọc được vào thanh ghi 2
- Đọc file rỗng: ghi -2 vào thanh ghi 2

e. Write

- Input: buffer(char*), số ký tự(int), id của file(OpenFileID)
- Output: -1: Lỗi, Số byte read thực sự: Thành công, -2: Thành công
- Chức năng: Đọc file với tham số là buffer, số ký tự cho phép và ID của file.

Ý tưởng: Viết thêm hàm PutString(char *buffer, int size) vào lớp synchconsole, dùng để đọc chuỗi kí tự.

Cài đặt:

- Lấy địa chỉ của buffer từ thanh ghi số 4
- Lấy số lượng kí tự từ thanh ghi số 5
- Lấy id của file từ thanh ghi số 6
- Kiểm tra tính hợp lệ của file. Ko hợp lệ -> ghi -1 vào thanh ghi 2
- Xử lý đọc file với các trường hợp:
 - Ghi file stdout: dùng hàm PutString vừa được cài đặt trong lớp synchconsole để ghi chuỗi kí tự lên console.
 - Đọc file bình thường: Dùng hàm Write trong lớp FileSystem để ghi file => Ghi số byte thực sự đọc được vào thanh ghi 2
- Đọc file rỗng: ghi -2 vào thanh ghi 2

f. Seek

- Input: Vị trí (int), id của file (OpenFileID)
- Output: -1 nếu gặp lỗi, Vị trí thực sự: Thành công.
- Chức năng: Di chuyển con trỏ đến vị trí thích hợp trong file với tham số đầu vào là id của file và vị trí muốn tìm kiếm.

Ý tưởng:

- Kiểm tra file ID truyền vào có nằm ngoài bảng mô tả file không.
 - Kiểm tra file có tồn tại không.
 - Kiểm tra có gọi Seek trên console không.
- ⇒ Với các trường hợp này ghi vào ô nhớ 2 giá trị -1 (gặp lỗi)
- Vị trí truyền vào bằng -1 => gán bằng Length của file của file.
 - Nếu vị trí truyền vào lớn hơn Length của file hoặc nhỏ hơn không => trả về 1
 - Nếu vị trí truyền vào hợp lệ thì trả về vị trí thực sự.

g. Remove

- Input: Tên file (char*)

- Output: -1 nếu lỗi, 0 nếu thành công
- Chức năng: Xóa file

Ý tưởng:

- Đọc địa chỉ của file từ thành ghi thứ 4.
- Sao chép không gian bộ nhớ User sang System, với độ dài tối đa là 32 + 1 byte
- Kiểm tra các trường hợp:
 - Nếu độ dài file name truyền vào bằng 0 => trả về -1 vào thanh ghi R2.
 - Nếu file name == NULL => trả về -1 vào thanh ghi R2.
 - Trường hợp còn lại thì xóa file bằng hàm Remove trong fileSystem.cc => trả về kết quả.

h. Sao chép vào vùng nhớ kernel-user

```
void StringSys2User(char *str, int addr, int convert_length = -1)
```

- Đọc dữ liệu từ bộ nhớ hệ thống ghi sang địa chỉ ô nhớ của người dùng

```
char *stringUser2System(int addr, int convert_length = -1)
```

- Đọc dữ liệu từ bộ nhớ người dùng ghi sang địa chỉ ô nhớ của hệ thống

2. Phần 2

a. Createfile: Tạo file

- Gọi hàm ReadString: đọc tên file
- Gọi hàm Create: tạo file mới, dùng tên file vừa đọc được

b. Cat:

- Gọi hàm ReadString: đọc tên file
- Gọi hàm Open: mở file với chế độ chỉ đọc
- Gọi hàm Seek: di chuyển đến cuối file đọc kích thước file, sau đó di chuyển đến đầu file để chuẩn bị đọc
- Tiến hành đọc file bằng cách gọi hàm Read đọc từng kí tự và gọi hàm PrintChar in từng kí tự ra màn hình
- Gọi hàm Close: đóng file

c. Copy

- Gọi hàm ReadString: đọc tên file nguồn và file đích
- Gọi hàm Open: mở file nguồn với chế độ chỉ đọc, file đích với chế độ đọc và viết
- Gọi hàm Seek: di chuyển đến cuối file đọc kích thước file nguồn, sau đó di chuyển đến đầu file chuẩn bị đọc

- Tiến hành đọc file nguồn bằng cách gọi hàm Read đọc từng kí tự và gọi hàm Write ghi từng kí tự vào file đích
- Gọi hàm Close đóng file nguồn và file đích

d. Delete

- Gọi hàm ReadString: đọc tên file cần xóa
- Gọi hàm Delete: xóa file

e. Concatenate:

- Gọi hàm ReadString: đọc tên 2 file nguồn và file đích
- Gọi hàm Open: mở 2 file nguồn với chế độ chỉ đọc, file đích với chế độ đọc và viết
- Gọi hàm Seek: di chuyển đến cuối file đọc kích thước file nguồn, sau đó di chuyển đến đầu file chuẩn bị đọc
- Tiến hành đọc lần lượt 2 file nguồn bằng cách gọi hàm Read đọc từng kí tự và gọi hàm Write ghi từng kí tự vào file đích

IV. DEMO CHƯƠNG TRÌNH

a. Createfile: Tạo file

```

● hoang@hoang:~/Desktop/Nachos_2_finished_editing/nachos/NachOS-4.0/code/test$ ../build.linux/nachos -x createfile
Enter file's name's length: 10
Enter file's name: nachos.txt
File nachos.txt created successfully!

Machine halting!

Ticks: total 492681976, idle 492678626, system 3270, user 80
Disk I/O: reads 0, writes 0
Console I/O: reads 14, writes 85
Paging: faults 0
Network I/O: packets received 0, sent 0

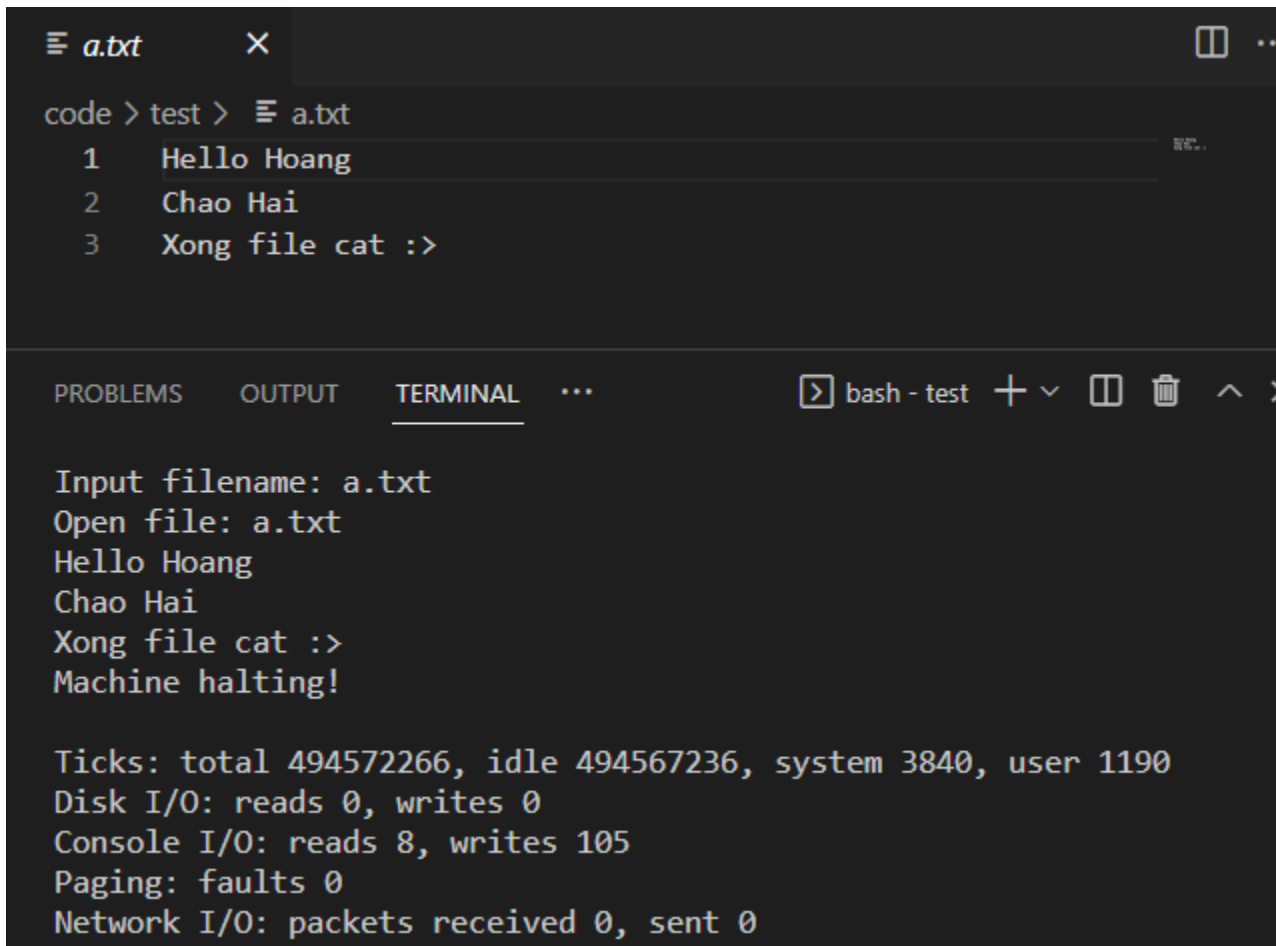
```

```

≡ matmult
C matmult.c
≡ matmult.coff
≡ matmult.o
≡ nachos.txt
≡ num_io
C num_io.c
≡ num_io.coff
≡ num_io.o
{} profile.code-profile

```

b. Cat:



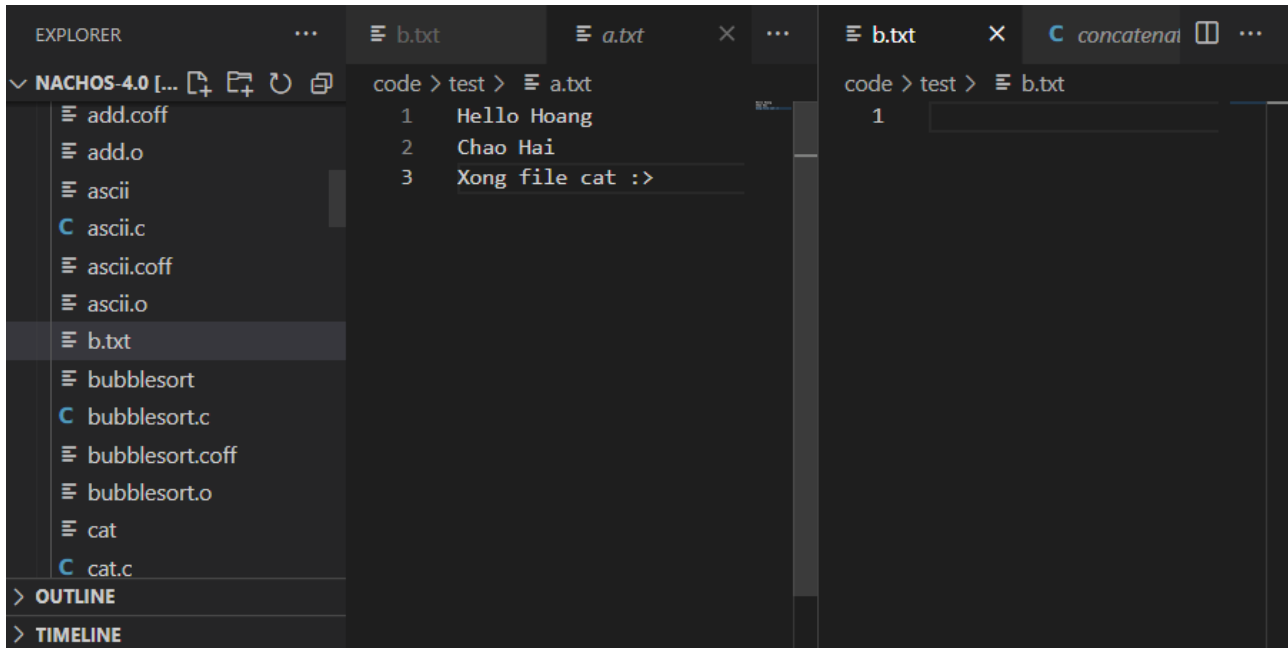
The image shows a code editor window with a file named `a.txt` open. The editor displays three lines of text: `1 Hello Hoang`, `2 Chao Hai`, and `3 Xong file cat :>`. Below the editor is a terminal window titled `bash - test`. The terminal output shows the execution of the `cat` command on `a.txt`, displaying the contents of the file and then the prompt `Xong file cat :>`. Below the output, the terminal shows system statistics: `Ticks: total 494572266, idle 494567236, system 3840, user 1190`, `Disk I/O: reads 0, writes 0`, `Console I/O: reads 8, writes 105`, `Paging: faults 0`, and `Network I/O: packets received 0, sent 0`.

```
code > test > cat a.txt
1 Hello Hoang
2 Chao Hai
3 Xong file cat :>

PROBLEMS OUTPUT TERMINAL ... bash - test
Input filename: a.txt
Open file: a.txt
Hello Hoang
Chao Hai
Xong file cat :>
Machine halting!

Ticks: total 494572266, idle 494567236, system 3840, user 1190
Disk I/O: reads 0, writes 0
Console I/O: reads 8, writes 105
Paging: faults 0
Network I/O: packets received 0, sent 0
```


c. Copy

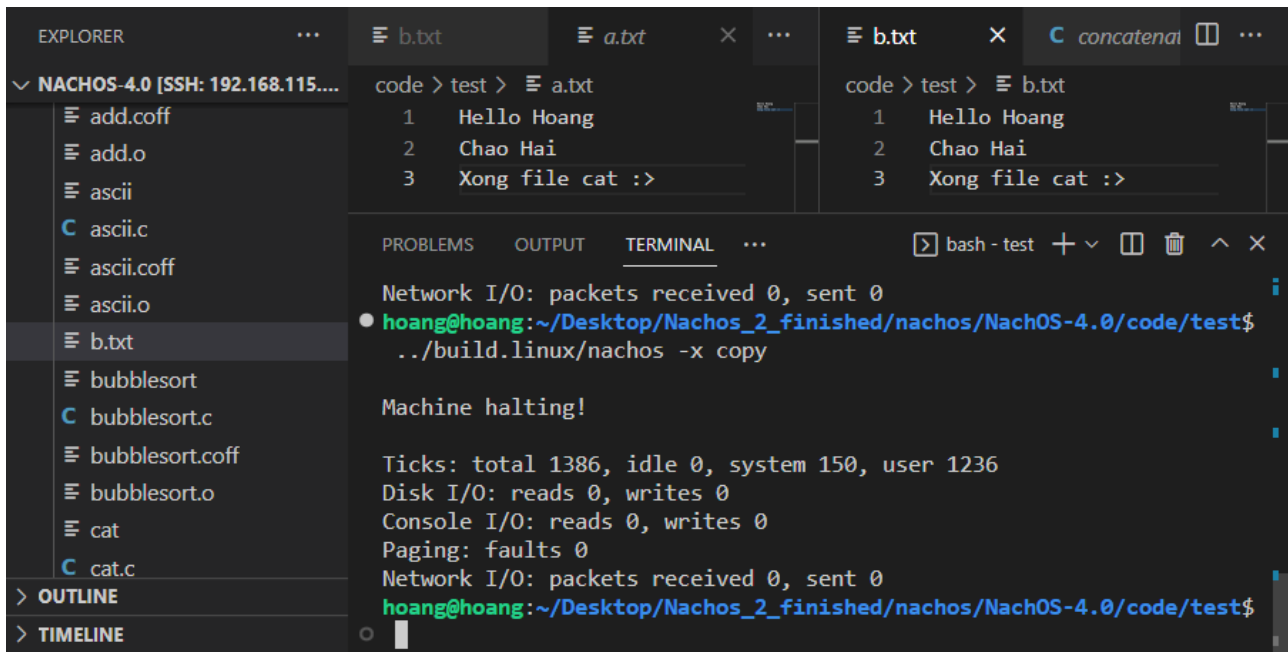


This screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the file structure of the 'NACHOS-4.0' project, with 'b.txt' selected. The main editor area is split into two panes. The left pane shows 'code > test > a.txt' with the following content:

```
1 Hello Hoang
2 Chao Hai
3 Xong file cat :>
```

The right pane shows 'code > test > b.txt' with the following content:

```
1
```



This screenshot shows the same VS Code interface as above, but with the Terminal panel open at the bottom. The terminal shows the execution of the 'copy' command in the 'code/test' directory. The output is as follows:

```
Network I/O: packets received 0, sent 0
hoang@hoang:~/Desktop/Nachos_2_finished/nachos/NachOS-4.0/code/test$
../build.linux/nachos -x copy

Machine halting!

Ticks: total 1386, idle 0, system 150, user 1236
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
hoang@hoang:~/Desktop/Nachos_2_finished/nachos/NachOS-4.0/code/test$
```

d. Delete

The screenshot shows the VS Code interface with the Explorer panel on the left displaying the file structure of the NachOS-4.0 project. The file `c.txt` is selected. The main editor shows the content of `c.txt`, which contains five lines of text. The terminal panel at the bottom shows the execution of the `delete` command, which prompts for the file's name and length, and then successfully removes the file.

```
code > test > c.txt
1 Hello Hoang
2 Chao Hai
3 Xong file cat :>Hello Hoang
4 Chao Hai
5 Xong file cat :>
```

```
hoang@hoang:~/Desktop/Nachos_2_finished/nachos/NachOS-4.0/code/test$
../build.linux/nachos -x delete
Enter file's name's length: 5
Enter file's name: c.txt
```

The screenshot shows the VS Code interface with the Explorer panel on the left displaying the file structure of the NachOS-4.0 project. The file `b.txt` is selected. The main editor shows the content of `b.txt`, which contains five lines of text. The terminal panel at the bottom shows the execution of the `delete` command, which prompts for the file's name and length, and then successfully removes the file. The terminal also displays system statistics and the prompt for the next command.

```
Enter file's name's length: 5
Enter file's name: c.txt
File c.txt removed successfully!

Machine halting!

Ticks: total 467125946, idle 467122966, system 2900, user 80
Disk I/O: reads 0, writes 0
Console I/O: reads 8, writes 80
Paging: faults 0
Network I/O: packets received 0, sent 0
hoang@hoang:~/Desktop/Nachos_2_finished/nachos/NachOS-4.0/code/test$
```

e. Concatenate:

EXPLORER NACHOS-4.0 [SSH: 192.168.115....]

- add.coff
- add.o
- ascii
- ascii.c
- ascii.coff
- ascii.o
- b.txt
- bubblesort
- bubblesort.c
- bubblesort.coff
- bubblesort.o
- c.tx
- cat

code > test > a.txt

```
1 Hello Hoang
2 Chao Hai
3 Xong file cat :>
```

code > test > b.txt

```
1
2 Nachos
3 Đồ án 2
```

EXPLORER NACHOS-4.0 [SSH: 192.168.115....]

- add.coff
- add.o
- ascii
- ascii.c
- ascii.coff
- ascii.o
- b.txt
- bubblesort
- bubblesort.c
- bubblesort.coff
- bubblesort.o
- c.txt
- cat

code > test > c.txt

```
1 Hello Hoang
2 Chao Hai
3 Xong file cat :>
4 Nachos
5 Đồ án 2
```

PROBLEMS OUTPUT TERMINAL

bash - test

```
hoang@hoang:~/Desktop/Nachos_2_finished/nachos/NachOS-4.0/code/test$
● ../build.linux/nachos -x concatenate

Machine halting!

Ticks: total 2132, idle 0, system 230, user 1902
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0
hoang@hoang:~/Desktop/Nachos_2_finished/nachos/NachOS-4.0/code/test$
```