**VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY**

**UNIVERSITY OF SCIENCE**

**FACULTY OF INFORMATION TECHNOLOGY**

Logo

Description automatically generated

**PROJECT 1 REPORT**

**SUBJECT: OPERATING SYSTEM**

**EXCEPTIONS & NACHOS SYSTEM CALLS**

**Class: 20CLC09**

**Theory Lecturer**

Mr. Le Giang Thanh

**Lab Lecturers**

Mr. Le Ha Minh

Mr. Nguyen Thanh Quan

**Students**

Le Tan Dat 20127011

Dang Tien Dat 20127458

Pham Thi Anh Phat 20127680

(2021 - 2022)

**TABLE OF CONTENT**

[**I.** **INTRODUCTION** 3](#_Toc103380906)

[**II.** **REQUIREMENT** 3](#_Toc103380907)

[**A.** **Hiểu mã NachOS** 3](#_Toc103380908)

[**B.** **Hiểu thiết kế** 3](#_Toc103380909)

[**C.** **System calls** 3](#_Toc103380910)

[**1.** **Xử lí exceptions:** 3](#_Toc103380911)

[**2.** **Tăng giá trị PC:** 4](#_Toc103380912)

[**3.** **int ReadNum();** 4](#_Toc103380913)

[**4.** **void PrintNum(int number);** 4](#_Toc103380914)

[**5.** **char ReadChar();** 5](#_Toc103380915)

[**6.** **void PrintChar(char character);** 5](#_Toc103380916)

[**7.** **int RandomNum();** 6](#_Toc103380917)

[**8.** **void ReadString (char \*buffer, int length)** 6](#_Toc103380918)

[**9.** **void PrintString (char \*buffer);** 6](#_Toc103380919)

[**10.** **Create (char \*name);** 7](#_Toc103380920)

[**11.** **OpenFileId Open (char \*name); int Close (OpenFileId id);** 7](#_Toc103380921)

[**12.** **int Read(char \*buffer, int size, OpenFileId id);** 7](#_Toc103380922)

[**13.** **int Seek(int position, OpenFileId);** 8](#_Toc103380923)

[**14.** **int Remove(char \*name);** 8](#_Toc103380924)

[**15.** **Help** 9](#_Toc103380925)

[**16.** **Ascii** 9](#_Toc103380926)

[**17.** **Sort** 10](#_Toc103380927)

[**18.** **Createfile** 10](#_Toc103380928)

[**19.** **Cat** 10](#_Toc103380929)

[**20.** **Copy** 11](#_Toc103380930)

[**21.** **Delete** 11](#_Toc103380931)

[ **Các bước cài đặt các system call:** 12](#_Toc103380932)

[**III.** **ASSESSMENT** 12](#_Toc103380933)

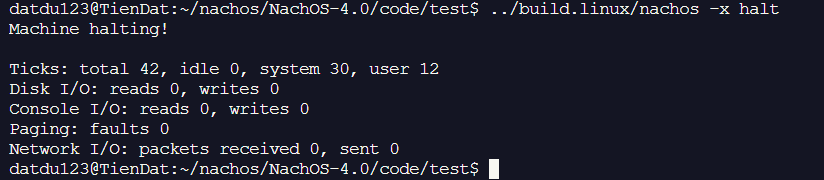
[**IV.** **REFERENCES** 13](#_Toc103380934)

1. **INTRODUCTION**

* Mức độ hoàn thành: 100%
* Ngôn ngữ lập trình: C++
* Tool: WSL (phần mềm sử dụng môi trường Linux)

1. **REQUIREMENT**
2. **Hiểu mã NachOS**

* Mã hiện nay ở mức 1 người dùng bằng chương trình C tại một thời điểm. Ta có thể dùng chương trình halt để thử nghiệm nachos, chương trình halt sẽ yêu cầu hệ điều hành tắt máy (máy ảo giả lập chạy nachos). Chạy chương trình “./build.linux/nachos -x ./test/halt”. Dò tìm khi chương trình người dùng nạp, chạy, và gọi một system call.



1. **Hiểu thiết kế**

* Hệ thống NachOS chia làm 3 phần chính như sau (không bao gồm mảng mạng): + Chương trình ứng dụng (…code/test)

+ Cỗ máy ảo MIPS (CPU , Registers , RAM , I/O , FileSys …v…v…)

+ Hệ điều hành NachOS (Nơi triển khai mã nguồn cho các system call …) Hệ điều hành NachOS cấu thành từ một vài file mã nguồn , dùng để xứ lý các system call , và được đặt trong thư mục userprog , thread ..v..vv… chủ yếu trong userprog Memory được chia vào 2 khu vực, còn được biết đến là kernel space và user space. Kernel Space: Mã thực thi có quyền truy cập không hạn chế vào bất kỳ không gian địa chỉ nào của memory và tới bất kỳ phần cứng nào. Nó được dành riêng cho các chức năng có độ tin cậy cao nhất bên trong hệ thống. User Space: Mã thực thi bị giới hạn không thể truy cập trực tiếp tới kernel space được. Khi đó lời gọi API được sử dụng tới kernel để truy vấn memory và truy cập thiết bị phần cứng Nachos cung cấp môi trường giả lập để chạy các chương trình C, chương trình biên dịch thông qua GCC/G++ tạo file object, sau đó chuyển sang định dạng đặc biệt của Nachos nhờ “coff2noff”.

1. **System calls**
2. **Xử lí exceptions:**

* Hoàn thành các system call được yêu cầu trong file exception.cc (SC\_PrintNum, SC\_PrintString,…). Viết lại file exception.cc để xử lí tất cả các exceptions được liệt kê trong *machine/machine.h*.

1. **Tăng giá trị PC:**

* Trong mã MIPS, sau một chỉ thị lệnh chúng ta phải tăng giá trị PC lên 4 bytes (void countValuePC()) nếu không chúng ta sẽ bị một vòng lặp vô hạn gọi thực hiện system call này mãi mãi.

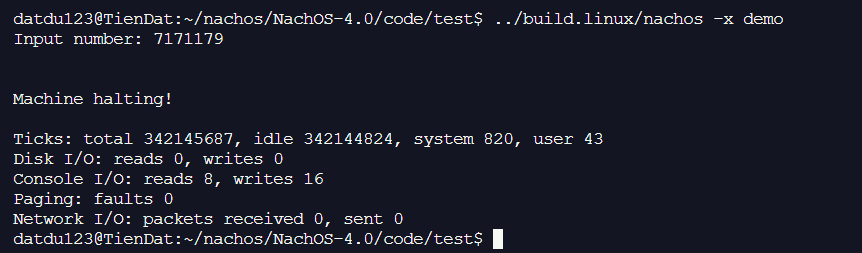
1. **int ReadNum();**

* *Comment:* System call này có chức năng cho người dùng nhập vào một số và syscall này sẽ sử dụng lớp synchConsoleIn để đọc số nguyên do người dùng nhập vào đó.
* *Cách cài đặt (ý tưởng):*

+ Sao chép các kí tự nhập vào một kiểu chuỗi và sau đó sẽ copy các kí tự số đó vào một buffer.

+ Kiểm tra các kí tự trong buffer có phải là số không, có chứa các kí tự đặt biệt hay không.

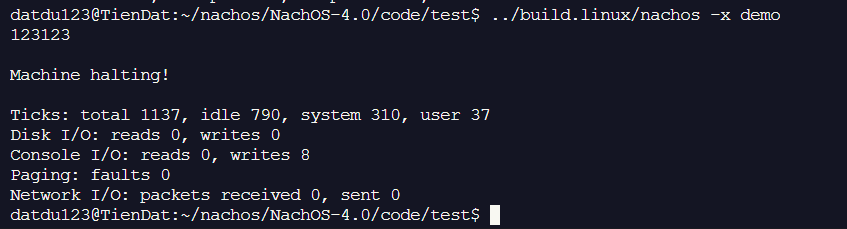
+ Nếu thỏa các kí tự đó là số, chuyển từ kiểu string sang kiểu int.

 + Trả về số chuyển đổi được.

1. **void PrintNum(int number);**

* *Comment:* System call này có chức năng là in ra một số được truyền vào. System call này sử dung synchConsoleOut để xuất số nguyên ra màn hình.
* *Cách cài đặt (ý tưởng):*

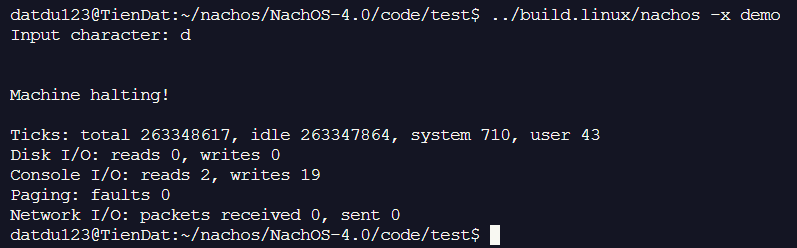
+ Sử dụng lớp synchConsoleOut->PutChar() để xuất kí tự ra màn hình.

 + Kiểm tra đối số truyền vào (number): nếu là số 0 thì put kí tự ‘0’ ra màn hình, nếu nó là số âm thì chúng ta put thêm kí tự ‘-‘ phía trước số đó và chuyển về kiểu kí tự để xuất.

1. **char ReadChar();**

* *Comment:* System call này có chức năng cho người dùng nhập vào một kí tự và syscall này sẽ sử dụng lớp synchConsoleIn để đọc kí tự do người dùng nhập vào đó.
* *Cách cài đặt (ý tưởng):*

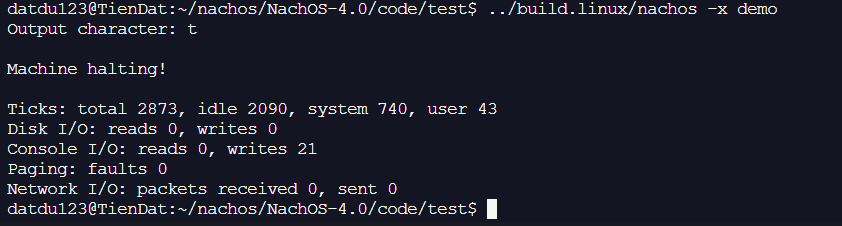
+ Sử dụng lớp kernel->synchConsoleIn->GetChar(); để yêu cầu người dùng nhập vào kí tự.



1. **void PrintChar(char character);**

* *Comment:* system call này có chức năng in một kí tự ra màn hình console và system call này sử dụng lớp synchConsoleIn để in kí tự đó ra màn hình.
* *Cách cài đặt (ý tưởng):*

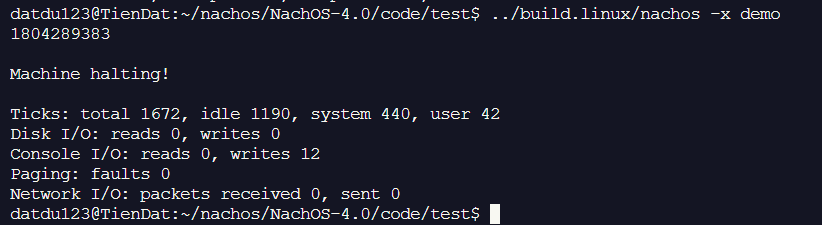
+ Sử dụng lớp kernel->synchConsoleOut->PutChar(); để xuất kí tự đó ra màn hình.



1. **int RandomNum();**

* *Comment:* system call này có chức năng ramdom một số.
* *Cách cài đặt (ý tưởng):*

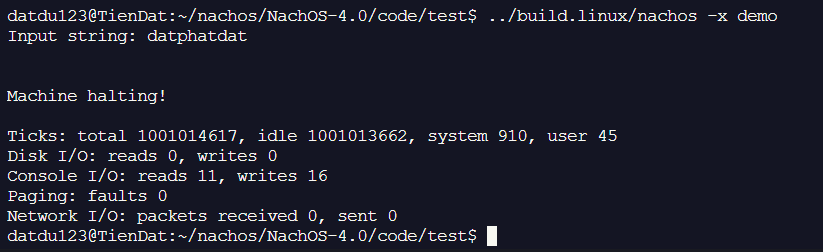
+ Sử dụng thư viện ramdom() (return ramdom()) để ramdon một số.



1. **void ReadString (char \*buffer, int length)**

* *Comment:* system call này dùng để đọc một chuỗi string được người dùng nhập từ bàn phím.
* *Cách cài đặt (ý tưởng):*

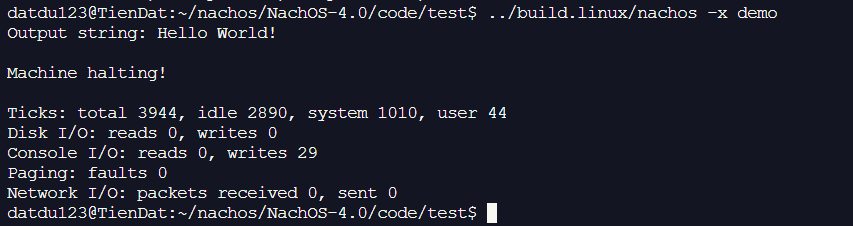
+ Vì khi người dùng nhập chuỗi thì nội dung được lưu trữ ở kernel space thì chúng ta sử dụng hàm để chuyển đổi dữ liệu từ kernelspace sang userspace.



1. **void PrintString (char \*buffer);**

* *Comment:* system call này dùng để in một chuỗi ra màn hình.
* *Cách cài đặt (ý tưởng):*

+ Duyệt hết chuỗi và sử dụng tương tự như PrintChar để xuất từng kí tự ra màn hình.



1. **Create (char \*name);**

* *Comment:* System call sẽ sử dụng Nachos FileSystem Object để tạo một file rỗng.
* *Cách cài đặt(ý tưởng):*

+ Kiểm tra điều kiện tên file cũng như độ dài tên file.

+ Sử dụng lớp kernel->fileSystem->Create(fileName) để tạo file, nếu thành công trả về 0 và ngược lại trả về -1.

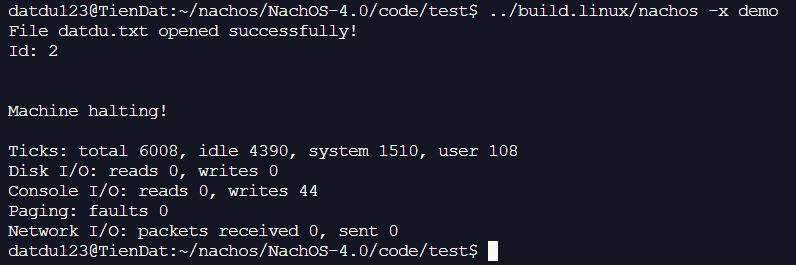
+ Chú ý, ở đây ta cần chuyển filename từ userspace sang systemspace bằng hàm user2System và system2User.

1. **OpenFileId Open (char \*name); int Close (OpenFileId id);**

* *Comment:* các system call này dùng để mở file và đóng file.
* *Cách cài đặt (ý tưởng):*

+ Sử dụng lớp kernel->fileSystem->Open(fileName, type) để mở file lên. Đối số truyền vào fileName là tên file cần mở và type là đối số chế độ ‘open to read’ hay ‘open to write’ hay ‘open to read and write’.

+ Sử dụng kernel->fileSystem->Close(id) để đóng file, trả về 0 nếu đóng thành công và ngược lại trả về -1.



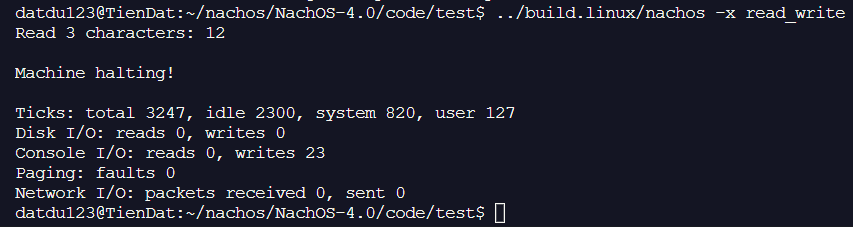
1. **int Read(char \*buffer, int size, OpenFileId id);**

* *Comment:* system call này dùng để đọc file với id cho trước.
* *Cách cài đặt:*

+ Kiểm tra xem fileId truyền vào = 0 thì chúng ta GetString của buffer, ngược lại thì chương trình gọi đến

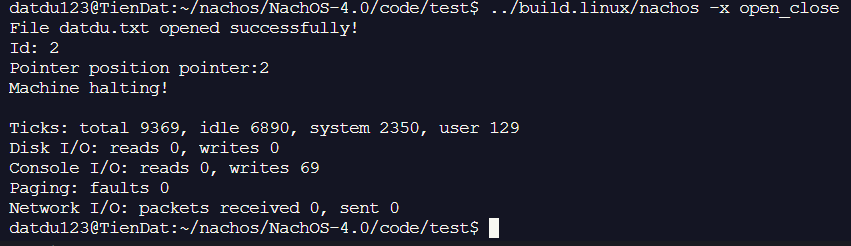
kernel->fileSystem->Read(buffer, charCount, fileId);

để thực hiện đọc file từ fileId và lưu nội dung vào buffer và 'size' của buffer vào charCount.



1. **int Seek(int position, OpenFileId);**

* *Comment:* system call này dùng để chuyển vị trí con trỏ tới vị trí position trong đối số truyền vào thích hợp trong file id, nếu position = -1 thì chuyển đến cuối file và trả về vị trí thực nếu thành công và trả về -1 nếu bị lỗi.
* *Cách cài đặt:*

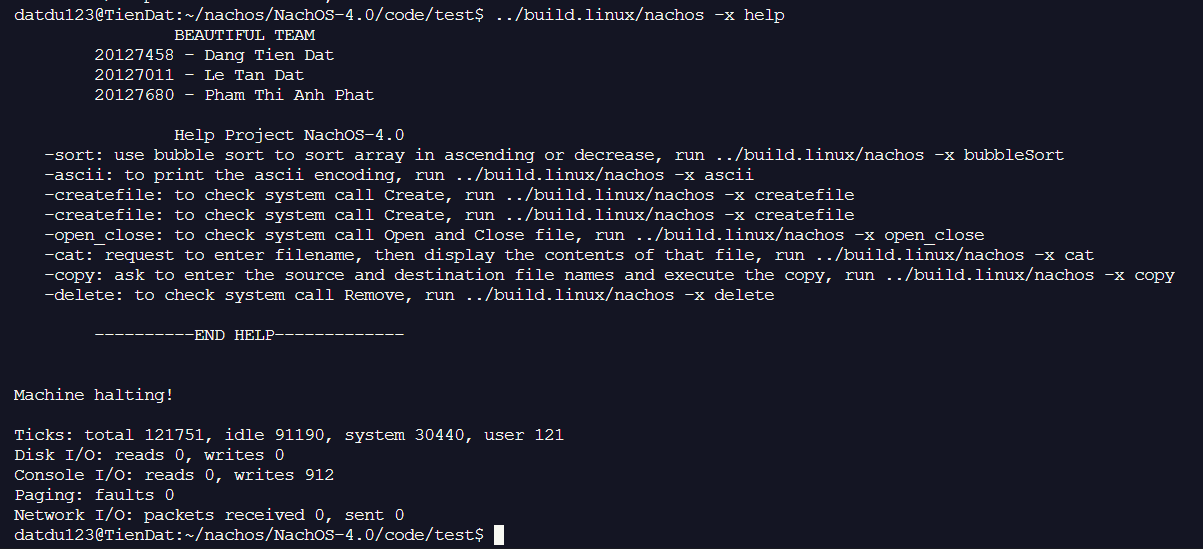


1. **int Remove(char \*name);**

* *Comment:* system call này dung để xóa file thư mục có tên name trong đối số truyền vào và sử dụng FileSystem object để xóa file.
* *Cách cài đặt:*

+ Sử dụng hàm kernel->fileSystem->Remove(fileName); để thực hiện công việc xóa file có tên fileName.

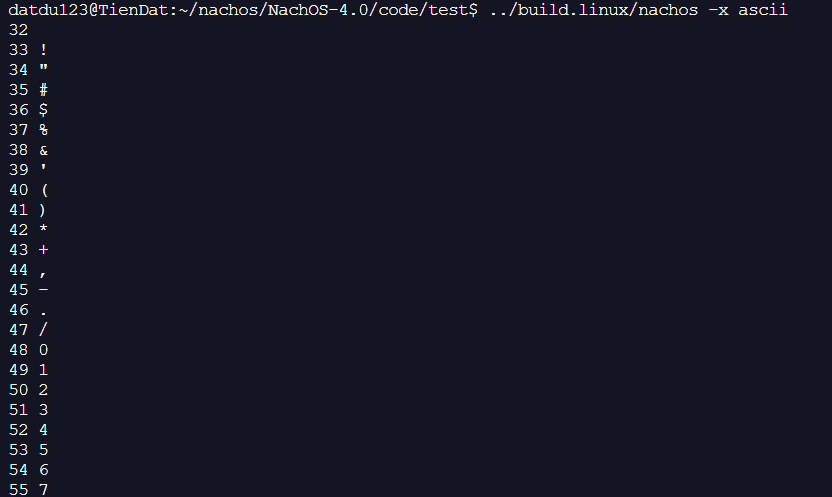
1. **Help**

* *Comment*: In ra các dòng giới thiệu về các thành viên trong nhóm và giới thiệu sơ lược về cách sử dụng của các hàm bằng việc gọi system call PrintString();

1. **Ascii**

* *Comment*: in ra bảng mã ASCII.
* *Cách cài đặt:*

+ In các kí tự từ 32 đến 126 ra kiểu char bằng cách gọi system call PrintChar((char)i);



1. **Sort**

* *Comment:* chương trình cho phép người dùng nhập vào n số nguyên với n <= 100 và sử dụng thuật toán bubble sort để tăng hay giảm dãy số.
* *Cách cài đặt(ý tưởng):*

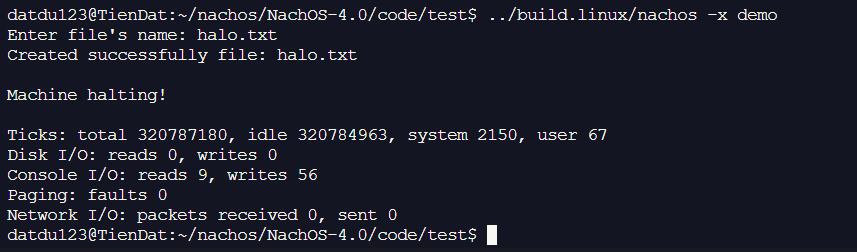
+ Kiểm tra điều kiện n nhập vào bằng system call ReadNum(n) và cho người dùng lựa chọn sắp xếp tăng hay giảm. Sau đó in ra dãy số trước và sau khi sắp xếp bằng system call PrintChar();

1. **Createfile**

* *Comment:* chương trình dùng để kiểm tra system call Create.
* *Cách thực hiện:*

+ Yêu cầu người dùng nhập vào tên file cần tạo.

+ Kiểm tra system call create có tạo được không, nếu được thì in "Create file successfully", nếu không in "Error create file".

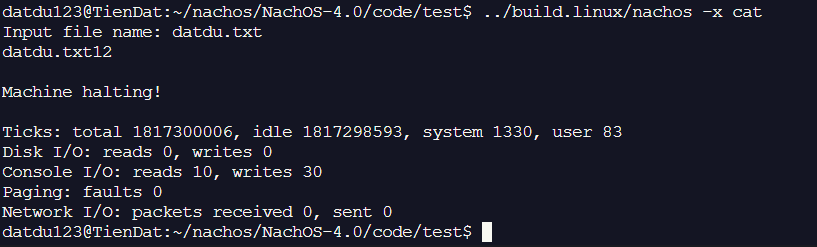


1. **Cat**

* Comment: chương trình cho người dùng nhập vào filename và hiển thị nội dung lên màn hình.
* Cách thực hiện:

+ Cho người dùng nhập vào tên file bằng cách gọi system call ReadString.

+ Mở file bằng cách gọi system call Open(filename, MODE\_READ); và đọc file lên bằng cách gọi hàm Read(buffer, 100, fileID) và in nội dung file ra màn hình bằng cách gọi system call PrintString().



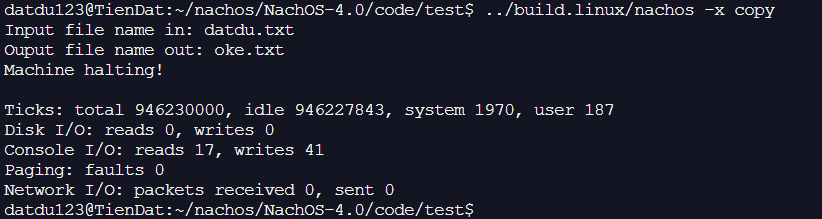
1. **Copy**

* *Comment:* thực hiện sao chép nội dung từ file này sang file khác.
* *Cách thực hiện:*

+ Yêu cầu người dùng nhập vào file nguồn cần copy, và nhập vòa file đích muốn copy tới.

+ Chúng ta mở fie nguồn được nhập vào để đọc các nội dung bên trong file đó lưu vào một buffer.

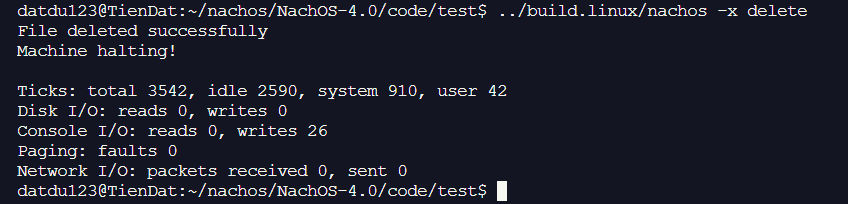
+ Sau đó chúng ta mở file đích được nhập vào, nếu file không tồn tại thì bắt đầu tạo file. Và sẽ lấy nội dung trong buffer sử dụng system call write để viết nội dung của buffer vào file đích.



1. **Delete**

* *Comment:* chương trình để kiểm tra system call Remove.
* *Cách thực hiện:*

+ Gọi system call Remove(filename), nếu thành công trả về 1 và ngược lại trả về 0.



* **Các bước cài đặt các system call:**

Bước 1: Tạo file demo.c trong thư mục test để viết chương trình main.

Bước 2: Trong file Makefile, thêm tên chương trình cần thực thi vào dòng PROGRAMS và thêm các dòng lệnh sau để tạo ra demo.c và demo.o và demo

Graphical user interface, text

Description automatically generated

Bước 3: Trong file code/test/start.S thêm đoạn mã MIPS code cho hàm cần thực thi để máy tính có thể hiểu được chương trình khai báo system call này để sử dụng.

Bước 4: Trong file code/userprog/syscall.h thêm dòng để truyền giá trị cho system call và prototype cho hàm cần viết.

Bước 5: Trong file code/userprog/ksyscall.h để định nghĩa hàm cần thực thi.

Bước 6: Trong file code/userprog/exception.cc xử lí các system call trong hàm exceptionHandler, định nghĩa các hàm handle\_SC có nhiệm vụ là xử lí đọc cái địa chỉ của thanh ghi và ghi lại kết quả vào địa chỉ.

Bước 7: Viết chương trình người dùng trong file demo.c và biên dịch chương trình với lệnh sau:

~nachos/NachOS-4.0/code/build.linux$ make && make depend

~nachos/NachOS-4.0/code/test: make && make depend

Sau khi biên dịch chương trình ta có các file demo.coff demo.o và demo

~nachos/NachOS-4.0/code/test: ../build.linux/nachos -x demo

* **Note:**
* Bởi vì hệ thống không thể dùng trực tiếp giá trị truyền từ người dùng nên cần có các hàm system2User và user2System để chuyển đổi qua lại.

1. **ASSESSMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Task** | **Description** | **Done** |
| 1 |  | Hiểu mã NachOS | x |
| 2 |  | Hiểu thiết kế | x |
| 3 | 1 | Exceptions handler | x |
|  | 2 | Program Counter | x |
|  | 3 | Read Num | x |
|  | 4 | Print Num | x |
|  | 5 | Read Char | x |
|  | 6 | Print Char | x |
|  | 7 | Random Num | x |
|  | 8 | Read String | x |
|  | 9 | Print String | x |
|  | 10, 14 | Create, Remove file | x |
|  | 11 | Open, Close file | x |
|  | 12 | Read file | x |
|  | 13 | Seek in file | x |
|  | 15, 16, 17 | Help, ascii, sort | x |
|  | 18, 19, 20, 21 | Create file, cat, copy, delete | x |
|  |  | Không để làm sập hệ điều hành | x |
| 4 |  | Báo cáo | x |

1. **REFERENCES**

[1]. <https://courses.ctda.hcmus.edu.vn/mod/folder/view.php?id=48702>

[2]. [https://github.com/leduythuccs/nachos-project/](https://github.com/leduythuccs/nachos-project/blob/master/code/network/post.cc)

[3]. <https://www.youtube.com/playlist?list=PLRgTVtca98hUgCN2_2vzsAAXPiTFbvHpO>