**BÁO CÁO ĐỒ ÁN 1  
NACHOS  
Môn: Hệ điều hành**

**Lớp TH2014/3**

**GVLT: Vũ Minh Trí.**

**GVTH: Trương Phước Hưng**

**Sinh viên thực hiện:**

1. **Trần Thị Nhã - 1412363**
2. **Đoàn Hiếu Tâm - 1412477**
3. **Nguyễn Đình Sơn - 1412465**

**MỤC LỤC**

[**I.** **PHÂN CÔNG CÔNG VIỆC VÀ ĐÁNH GIÁ** 3](#_Toc448780269)

[**II.** **BÁO CÁO KẾT QUẢ ĐỒ ÁN** 3](#_Toc448780270)

[**1.** **Sơ lược về NachOS.** 3](#_Toc448780271)

[**1.1** **Mô hình giao tiếp chung giữa hệ điều hành và người dùng.** 3](#_Toc448780272)

[**1.2** **Giao tiếp giữa hệ điều hành nachos và chương trình người dùng.** 4](#_Toc448780273)

[**2.** **Cài đặt các xử lý cho Exceptions và system calls.** 4](#_Toc448780278)

[**a)** **Xử lý exceptions:** 4](#_Toc448780279)

[**b)** **Xử lý System calls:**. 5](#_Toc448780280)

[**c)** **Xử lý tăng program counter:** 5](#_Toc448780281)

[**3.** **Cài đặt system call.** 6](#_Toc448780285)

[**a)** **Mô tả cài đặt ReadInt:** 7](#_Toc448780286)

[**b)** **Mô tả cài đặt PrintInt:** 7](#_Toc448780287)

[**c)** **Mô tả cài đặt ReadChar:** 7](#_Toc448780288)

[**d)** **Mô tả cài đặt void PrintChar(char character):** 8](#_Toc448780289)

[**e)** **Mô tả cài đặt void ReadString(char[]buffer, int lenght):** 8](#_Toc448780290)

[**f)** **Mô tả cài đặt void PrintString (char[] buffer):** 8](#_Toc448780291)

[**4.** **Viết chương trình:** 9](#_Toc448780292)

[**a)** **Chương trình help:** 9](#_Toc448780293)

[**b)** **Chương trình ascii:** 9](#_Toc448780294)

[**c)** **Chương trình sort:** 9](#_Toc448780295)

[**5.** **Các hình ảnh minh họa kết quả:** 10](#_Toc448780296)

[**III.** **Tài liệu tham khảo:** 17](#_Toc448780297)

1. **PHÂN CÔNG CÔNG VIỆC VÀ ĐÁNH GIÁ**

|  |  |  |  |
| --- | --- | --- | --- |
| **Công việc thực hiện** | **Người thực hiện** | **Mức độ hoàn thành** | **Đánh giá của nhóm** |
| Tìm hiểu và cà đặt các xử lý cho Exceptions và System calls | Trần Thị Nhã | 100% | 100% |
| Cài đặt system call | Đoàn Hiếu Tâm | 100% | 100% |
| Viết chương trình | Nguyễn Đình Sơn | 100% | 100% |

1. **BÁO CÁO KẾT QUẢ ĐỒ ÁN**
2. **Sơ lược về NachOS.**
   1. **Mô hình giao tiếp chung giữa hệ điều hành và người dùng.**

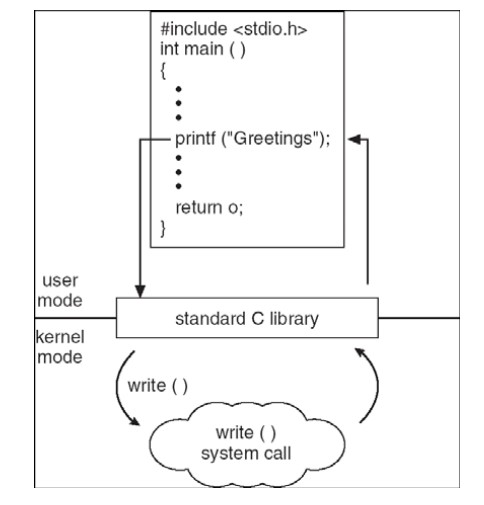


Figure 1: Mô hình giao tiếp giữa HĐH nachos và người dùng.

* 1. **Giao tiếp giữa hệ điều hành nachos và chương trình người dùng.**

Xét chương trình **x.c** được viết để chạy trên hệ điều hành nachos. Để chương trình này chạy được ta phải biên dịch nó. Quấ trình biên dịch gồm 3 giai đoạn:

1. **X.c** được biên dịch bởi cross-compiler gcc thành tập tin **x.s** là mã hợp ngữ chạy trên kiến trúc MIPS.
2. Tập tin **x.s** này sẽ được liên kết với tập tin **start.s** để tạo thành tập tin **x.coff**, là định dạng thực thi trên hệ điều hành Linux cho kiến trúc máy MIPS.
3. Tập tin **x.coff** được chuyển thành tập tin **halt.noff** (hình dưới), là dạng thực thi trên hệ điều hành nachos cho kiến trúc máy MIPS, sử dụng tiện ích “**coff2noff**” được cung cấp sẵn trong thư mục code/bin.

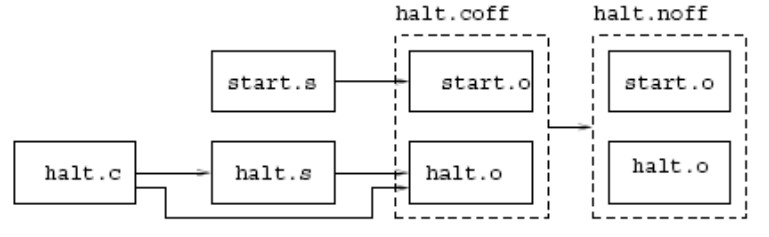
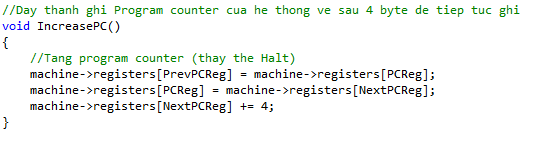


Figure 2: Mô hình giao tiếp giữa hệ điều hành nachos

và người dùng, lấy halt làm ví dụ

1. **Cài đặt các xử lý cho Exceptions và system calls.**
2. **Xử lý exceptions:**- Viết lại file exception.cc để xử lý tất cả các exceptions được liên kết trong machine/machine.h. Hầu hết các exception trong này là run-time errors, khi các exception này xảy ra thì user program không thể phục hồi. Trường hợp đặc biệt duy nhất là **no exception** sẽ trả quyền điều khiển về cho hệ điều hành, còn syscalls exception sẽ được xử lý bởi các hàm chúng ta biết cho user system calls. Với tất cả các exception khác, hệ điều hành hiển thị ra một thông báo lỗi và halt hệ thống.  
   - Với hàm ExceptionHandle(ExceptionType which), dùng Switch(which) để xử lý các exception (which), mỗi case tương ứng với mỗi exception (dùng break sau mỗi case để tránh trôi sang case khác).
3. **Xử lý System calls:**- Cài đặt cross-compiler, thêm thư mục “gnu-decstation-ultrix” vào thư mục hdh, đổi tên “gnu-decstation-ultrix” thành “cross-compiler” sau đó vào trong thư mục cross-compiler/decstation-ultrix/ đổi tên thư mục “2.95.3” thành thư mục bin.  
   - Thêm User2System và System2User . Công dụng: để chuyển đổi vùng nhớ từ User mode sang Kernel Mode và ngược lại. Hai hàm này sẽ trở thành phương thức public trong lớp Machine. Khai báo hàm nằm ở **machine/machine.h** và phần định nghĩa sẽ nằm ở **machine/machine.cc.  
   -** Viết lại cấu trúc điều khiển của chương trình để nhận các nachos system calls. Kiểm tra cấu trúc mới bằng cách dùng system call **Halt** để kiểm tra tính đúng đắn của cấu trúc mới  
    Các system call sẽ được xử lý trong case SyscallException bằng switch(type). Mỗi case của Switch(type) tương ứng với một system call. Thêm interrupt-> Halt() ở cuối cùng case SyscallException (trước break) để kiểm tra tính đúng đắn.
4. **Xử lý tăng program counter:**

* Tất cả các system call (không phải halt) sẽ yêu cầu nachos tăng program counter trước khi system call trả kết quả về. Nếu không lập trình đúng phần này thì sẽ bị vòng lặp gọi thực hiện system call này mãi mãi. Cũng như các hệ thống khác, MIPS xử lý dựa trên giá trị của program counter, vì vật bạn phải viết mã để tăng giá trị biến program counter, tìm đoạn mã này trong thư mục **machine** (file/code/machine/mipssim.c, dòng thứ 118 và dóng thứ 560 đến 564). Bạn phải copy mã này vào vị trí thích hợp trong phần xử lý các system call của bạn. Hiện tại bạn phải dùng **Halt system call** tại cuối mỗi user program.
* Được viết riêng trong hàm **IncreasePC ()**.  
    
  
* Ta sử dụng machine->registers để gán PrevPCReg = PCReg, PCReg = NextPCReg, NextPCReg = NextPCReg + 4 nhằm đẩy giá trị các thanh ghi trên tăng dần.   
  Đặt chuỗi lệnh này ở cuối cùng case SyscallException trước break (khi kết thúc một *system call* sẽ văng khỏi swich(type) và nhảy tới chuỗi lệnh này) thay thế cho hàm Halt() cũ.

1. **Cài đặt system call.***Các bước thực hiện khi cài đặt một system call mới:****Bước 1:***Vào userprog/syscall.h để define một system call mới kèm theo là mã system call.  
   ***Bước 2:***Vào code/test/start.c và code/test/start.s để khai báo tên system call sẽ viết trong exception.cc. ***Bước 3:***Vào userprog/exception.cc cài đặt system call nằm trong cấu trúc rẽ nhánh của SyscallException.  
   ***Bước 4:***Tạo một tập tin .c và viết chương trình ở mức người dùng trong code/test.  
   ***Bước 5:***Vào code/test/Makefile thêm tập tin vừa tạo ở hàng all:... .  
   ***Bước 6:***Biên dịch lại nachos.
2. **Mô tả cài đặt ReadInt:**

* Input: Không có.
* Output: Số nguyên.
* Công dụng: Đọc một số nguyên do người dùng nhập vào.
* Cấp phát vùng nhớ kiểu char để chứa chuỗi số đọc vào.
* Đọc thông tin vào vùng nhớ vừa cấp phát bằng phương thức Read trong SynchConsole. Trả về số byte đọc được vào biến lenght.
* Nếu length < 1 thì gán thanh ghi kết quả = -1, tăng PC lên.
* Kiểm tra tính đúng đắn của chuỗi ký số nhập vào (chỉ gồm các ký tự từ ‘0’ đến ‘9’). Nếu sai gán thanh ghi kết quả = 0, tăng PC. Chuyển chuỗi số thành số nguyên KQ = ktu – ‘0’ + KQ \* 10 – Viết kết quả vào thanh kết quả.

1. **Mô tả cài đặt PrintInt:**

* Input: Số nguyên.
* Output: Chuỗi số nguyên.
* Công dụng: In một số nguyên là tham số truyền vào ra màn hình.
* Đọc giá trị trong thanh ghi số 4 (tham số duy nhất).
* Chuyển đối số thành chuỗi theo công thức: chuoi = so % 10 + ‘0’.
* Dùng hàm Write trong SynchConsole để in chuỗi ra màn hình.

1. **Mô tả cài đặt ReadChar:**

* Input: Không có.
* Output: Kí tự.
* Công dụng: Đọc một kí tự do người dùng nhập vào.
* Dùng hàm Read trong SynchConsole để đọc ký tự được nhập vào.
* Dùng hàm write để viết vào thanh kết quả.

1. **Mô tả cài đặt void PrintChar(char character):**

* Input: Ký tự
* Output: Ký tự.
* Công dụng: In một lý tự làm tham số truyền vào ra màn hình.
* Dùng hàm Write trong SynchConsole để viết ký tự ra màn hình.
* Sử dụng lớp SynchConsole để xuất một ký tự ra màn hình.
* Đọc giá trị từ thanh ghi r4 và dùng hàm Write thuộc lớp SynchConsole để in ký tự đọc được ra màn hình.

1. **Mô tả cài đặt void ReadString(char[]buffer, int lenght):**

* Input: Buffer và độ dài chuối nhập vào.
* Output: Không có.
* Công dụng: đọc vào một chuỗi và độ dài của nó.
* Sử dụng lớp SynchConsole để đọc một chuỗi ký tự vào trong buffer (chuỗi sẽ kết thúc khi người dùng nhấn **Enter**, hoặc có chiều dài lớn hơn hoặc bằng giá trị **lenght**).
* Chú ý rằng buffer là một vùng nhớ userspace, khi người dùng nhập chuỗi thì nội dung được lưu trữ ở kernel space -> cần viết một hàm tương ứng để chuyển dữ liệu từ kernelspace sang userspace.
* Lấy địa chỉ buffer từ r4 và độ dài chuỗi tối đa từ r5, đọc chuỗi bằng SynchConsole.Read sau đó copy chuỗi từ System space sang User space bằng hàm System2User.

1. **Mô tả cài đặt void PrintString (char[] buffer):**

* Input: Buffer
* Output: Chuối có địa chỉ buffer.
* Công dụng: In một chuỗi có địa chỉ buffer truyền vào ra màn hình.
* Chú ý tương tự như ReadString cần phải có hàm để chuyển đổi dữ liệu từ userspace qua kernel sace.
* Đưa buffer vào từ thanh ghi r4, chuyển dữ liệu từ User sang phía System bằng User2System, đếm độ dài thật của chuỗi cần in và gọi SynchConsole.Write để in chuỗi đó ra màn hình.

1. **Viết chương trình:**
2. **Chương trình help:**

* Dùng hàm **PrintString()** đã viết ở trên để xuất các chuỗi hướng dẫn ra màn hình.
* Chương trình được viết trong file **help.c.**

1. **Chương trình ascii:**

* In các kí tự trong bảng mã ASCII bằng các chuyển từ int sang char rồi dùng hàm **PrintChar()** để in các kí tự ra màn hình.
* Chương trình được viết trong file **ascii.c.**

1. **Chương trình sort:**

* Tạo một mảng một chiều, đọc các số nguyên vào các phần tử bằng hàm **ReadInt()**.
* Dùng thuật toán BubbleSort để sắp xếp.
* In các phần tử của mảng đã sắp xếp ra màn hình bằng hàm **PrintInt()**.

1. **Các hình ảnh minh họa kết quả:**

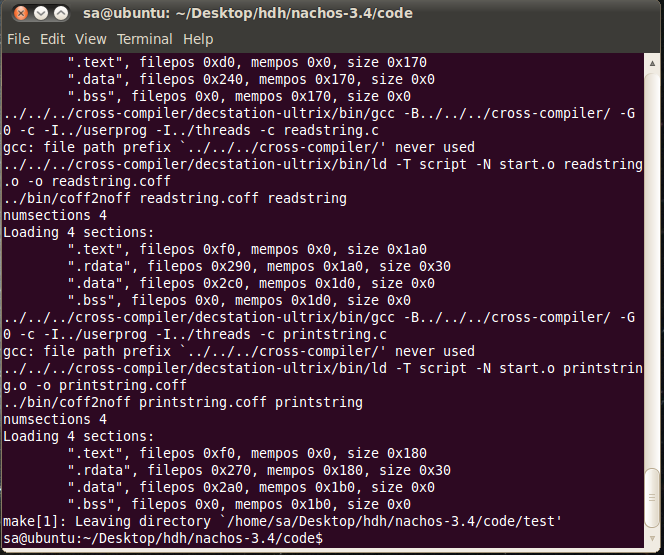


Figure 3: Make all thành công.

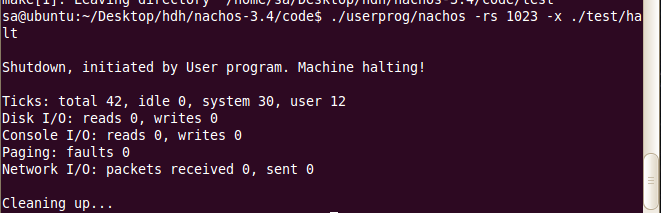


Figure 4:Demo halt.

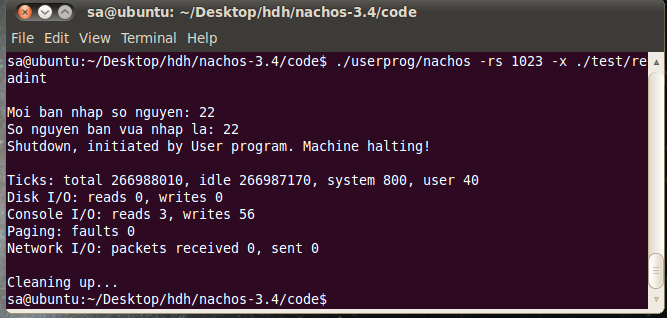


Figure 5: Demo readint.

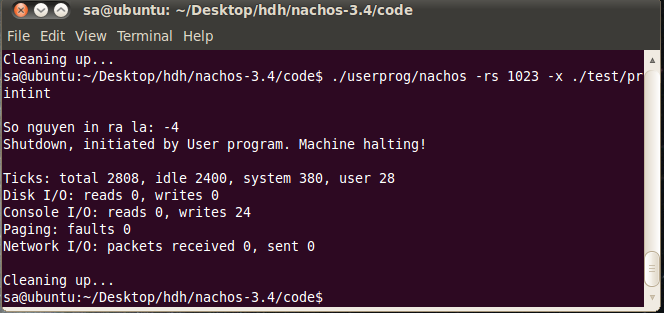


Figure 6: Demo printint.

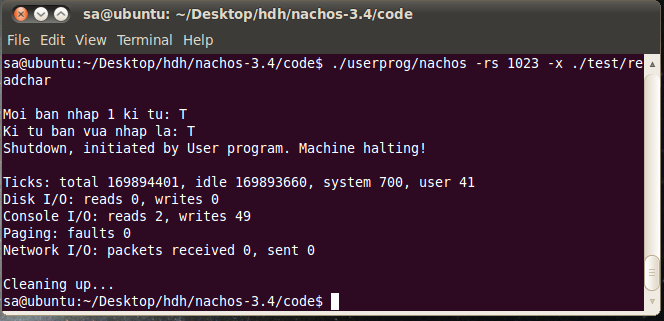


Figure 7: Demo readchar.

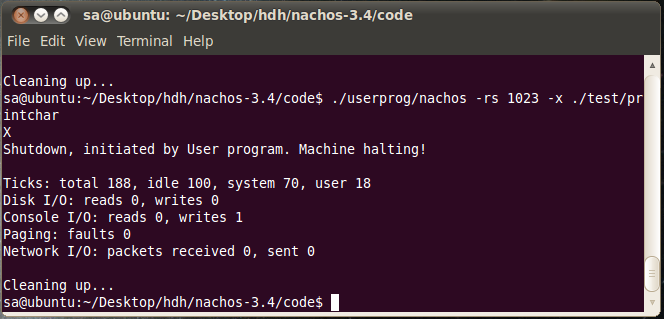


Figure 8: Demo printchar.

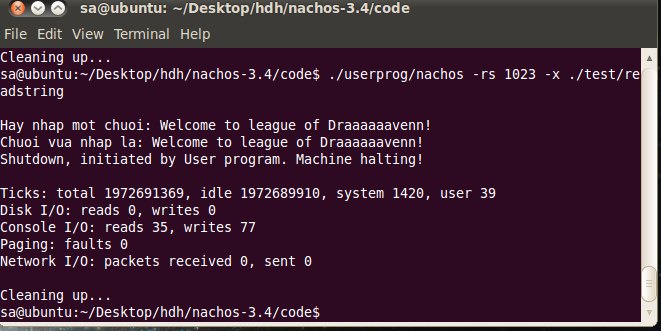


Figure 9: Demo readstring.

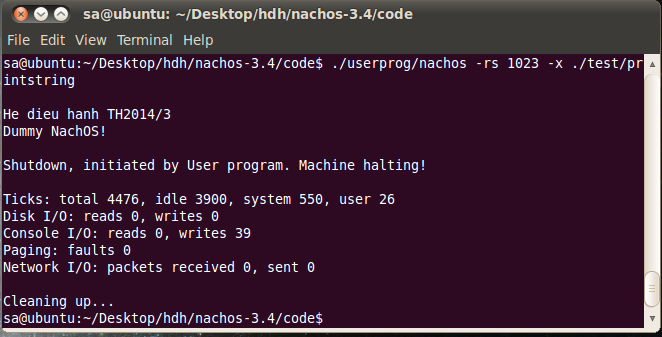


Figure 10: Demo printstring.

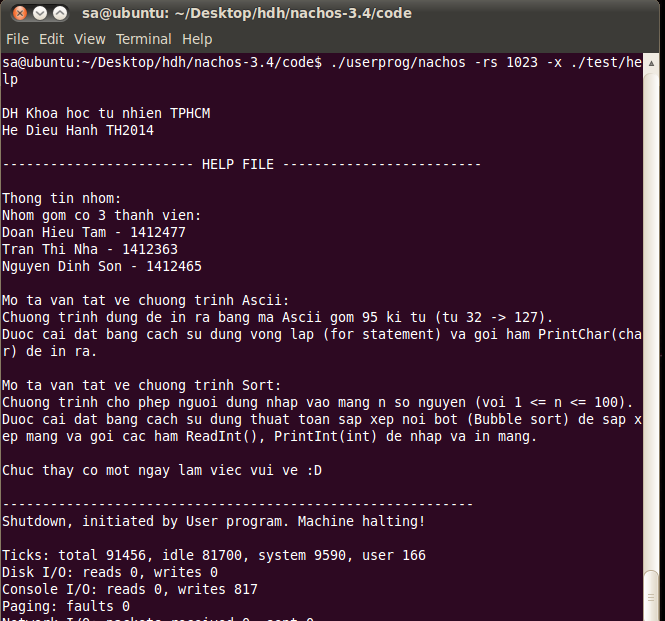


Figure 11: Demo help.

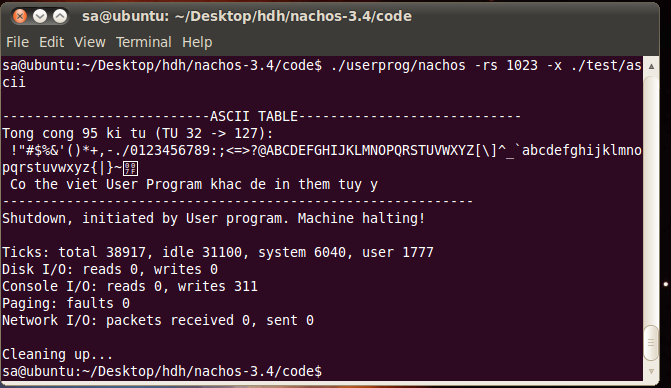


Figure 12: Demo ascii.

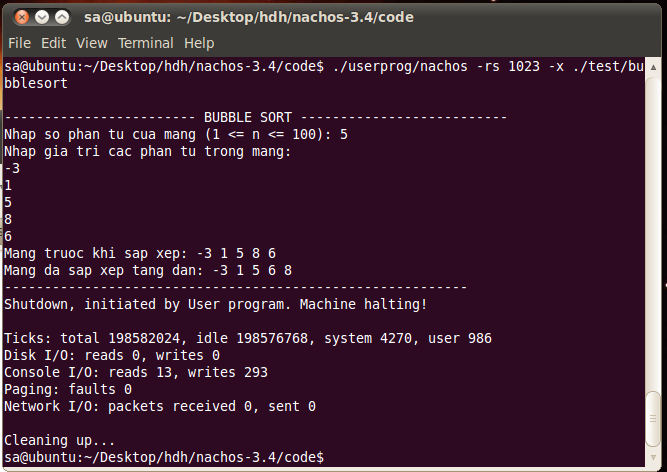


Figure 13: Demo bubblesort.

1. **Tài liệu tham khảo:**

* Slide[1] Biên dịch và cài đặt Nachos.
* Slide[2] Giao tiếp giữa hệ điều hành Nachos và người dùng.
* Slide[3] Cách viết một system call.
* Slide[4] Cách thêm một lớp vào Nachos.
* *http://dotrungduchd.blogspot.com/2013/05/system-call-of-nachos.html*