**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**-------**



**BÁO CÁO PROJECT 2**

**MÔN HỌC: HỆ ĐIỀU HÀNH**

**Sinh viên thực hiện:** 20120443 - Nguyễn Tấn Chữ

20120444 - Nguyễn Chí Công

20120447 - Trịnh Quốc Cường

20120383 - Nguyễn Đức Tiến

20120446 - Nguyễn Đình Cường

**| Giáo viên hướng dẫn |**

Thầy Lê Viết Long

Thành Phố Hồ Chí Minh - 2022

MỤC LỤC

[Phần chia công việc 4](#_Toc118729185)

[I/ Tìm hiểu và cài đặt tổng quan 5](#_Toc118729186)

[1. Tổng quan về NachOS 5](#_Toc118729187)

[2. Cài đặt và biên dịch NachOS 5](#_Toc118729188)

[3. Quá trình biên dịch trên NachOS 5](#_Toc118729189)

[4. Cài đặt tổng quan 6](#_Toc118729190)

[II/ Exception và System Calls 7](#_Toc118729191)

[1. Xử lý các exception được liệt kê trong machine/machine.h 7](#_Toc118729192)

[2. Viết lại cấu trúc điều khiển của chương trình để nhận các NachOS system call 8](#_Toc118729193)

[3. Cài đặt hàm tăng program counter 8](#_Toc118729194)

[4. Cài đặt các system call 8](#_Toc118729195)

[III/Test system call 11](#_Toc118729196)

[1. System call readInt, printInt 11](#_Toc118729197)

[Code 11](#_Toc118729198)

[a) Một số nguyên dương 11](#_Toc118729199)

[b) Một chuổi kí tự 11](#_Toc118729200)

[c) Một số nguyên âm 12](#_Toc118729201)

[d) Một kí tự 12](#_Toc118729202)

[e) Một số thực 12](#_Toc118729203)

[2. System call readChar, printchar 13](#_Toc118729204)

[Code 13](#_Toc118729205)

[a) Một ký tự 13](#_Toc118729206)

[b) Nhiều kí tự 13](#_Toc118729207)

[3. System call readString ,printString 14](#_Toc118729208)

[Code 14](#_Toc118729209)

[a) Chuổi kí tự 14](#_Toc118729210)

[4. Chương trình help 14](#_Toc118729211)

[5. Chương trình ASCII 15](#_Toc118729212)

[6. Chương trình sort (bubble sort) 16](#_Toc118729213)

[Tài Liệu Tham Khảo 16](#_Toc118729214)

[link video demo 16](#_Toc118729215)

# Phần chia công việc

|  |  |  |  |
| --- | --- | --- | --- |
| **MSSV** | **Họ và tên** | **Công việc** | **Mức độ hoàn thành** |
| 20120443 | Nguyễn Tấn Chữ | * Viết lại file exception * Viết lại cấu trúc điều khiển chương trình * Viết báo cáo | 100% |
| 20120444 | Nguyễn Chí Công | * Tìm và cài đặt hàm tăng program counter * Tổng hợp mã nguồn | 100% |
| 20120447 | Trịnh Quốc Cường | * Viết system call ReadString và PrintString * Viết system call ReadChar và PrintChar * Viết system call ReadInt và PrintInt | 100% |
| 20120383 | Nguyễn Đức Tiến | * Viết chương trình Help * Viết chương trình Ascii * Viết chương trình sort (Bubble sort) | 100% |
| 20120446 | Nguyễn Đình Cường | * Viết chương trình test các system call * Viết báo cáo * Quay video demo | 100% |

# I/ Tìm hiểu và cài đặt tổng quan

## Tổng quan về NachOS

* NachOS (Not Another Completely Heuristic Operating System) là phần mềm mã nguồn mở giả lập một máy tính ảo và một số thành phần cơ bản của hệ điều hành chạy trên máy tính ảo này nhằm giúp cho việc tìm hiểu và xây dựng các thành phần phức tạp của hệ điều hành.

## Cài đặt và biên dịch NachOS

* Sử dụng VNware Workstation để tạo máy ảo và cài đặt hệ điều hành Ubuntu 14.04 lên trên máy ảo.
* Sau đó tiến hành cài trình biên dịch gcc/g++ trong Ubuntu
* Tiến hành biên dịch NachOS ( mã nguồn NachOS đã được cấu hình với cross-compiler-MIPS)

## Quá trình biên dịch trên NachOS

* Trong thư mục ../nachos-3.4/code/test có file hợp ngữ start.s, khi biên dịch, file này sẽ biên dịch các file mã nguồn <tenfile X>.c thành file hợp ngữ có đuôi <tenfile X>.s
* Sau đó file <tenfile X>.s này sẽ liên kết với file start.s tạo thành 1 file <tenfile X>.coff ( bao gồm file <tenfile X>.o và start.o ), đây là dạng file thực thi trên Linux vơi kiến trúc MIPS.
* File <tenfile X>.coff sẽ được công cụ coff2noff ( được viết sẵn trong NachOS) chuyển thành file <tenfile X>.noff , đây chính là dạng file thực thi trên NachOS với kiến trúc MIPS.
* Quá trình biên dịch này có thể mô tả qua 1 ví dụ với file ‘haft.c’ sau đây:

Diagram, schematic

Description automatically generated

## Cài đặt tổng quan

1. Cài đặt trước

* Để sao chép được vùng nhớ từ User sang System và từ System sang User, tiến hành cài đặt lần lượt 2 hàm: Char\* User2System(int virtAddr, int limit) và int System2User(int virtAddr, int len, char\* buffer) trong file ../code/userprog/exception.cc
* Để tránh việc NachOS sẽ bị vòng lặp gọi thực hiện system call này mãi mãi thì cần tăng program counter trước khi system call trả kết quả về. Tiến hành cài đặt hàm tăng Program counter trong file ../code/userprog/exception.cc

1. Cài đặt system calls:

* Trong file ../code/userprog/syscall.h, tiến hành khai báo các system call sẽ cài đặt.
* Để NachOS có thể call những system call đã khai báo, cần phải thêm 1 số mã lệnh MIPS cho từng system call vào 2 file ../code/test/start.c và ../code/test/start.s. 1 đoạn lệnh ví dụ về system call ReadInt:

Graphical user interface

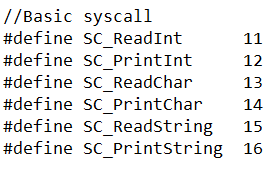
Description automatically generated with low confidence

* Các system call còn lại đều có cùng format như đoạn lệnh trên. Tiến hành cài đặt lần lượt vào 2 file ở trên

# II/ Exception và System Calls

## Xử lý các exception được liệt kê trong machine/machine.h

* Đầu tiên, xác định kiểu exception nào với biến which,
* Nếu exception thuộc vào các kiểu sau: PageFaultException, ReadOnlyException, BusErrorException,AddressErrorException,OverflowException, IllegalInstrException, NumExceptionTypes thì Halt hệ thống.
* Nếu exception là SyscallException thì lúc này, cần xác định giá trị thanh ghi loại 2, để biết được exception của system call nào. Thêm vào file userprog/syscalls.h các define sau đây để kiểm tra và xử lí exception:



## Viết lại cấu trúc điều khiển của chương trình để nhận các NachOS system call

* Tiến hành cài đặt lại hàm ExceptionHandler trong file userprog/exception.cc. Từ cấu trúc if-else chuyển sang cấu trúc switch-case.

## Cài đặt hàm tăng program counter

* + Bước 1: Lấy giá trị program hiện tại gán vào previous program counter.
  + Bước 2: Đọc next program counter và gán vào program counter
  + Bước 3: Cập nhật lại giá trị cho next program counter

## Cài đặt các system call

1. Cài đặt system call **int ReadInt()**
   * Bước 1: lấy buffer chứa chuỗi số nguyên đọc được, để lấy được buffer này, sử dụng hàm **Read(buffer, 256)** từ class **SynchConsole** để nạp chuỗi số vào buffer đồng thời trả ra độ dài của chuỗi số
   * Bước 2: Tiến hành lặp qua tất cả phần tử trong buffer, nếu có bất kỳ phần tử nào không thuộc khoảng ‘0’ đến ‘9’ thì trả về 0, tăng programcounter, giải phóng buffer, ngược lại thì tiếp tục lặp
   * Bước 3: Chuyển buffer thành số nguyên, rồi ghi vào thanh ghi số 2 với hàm **WriteRegister** từ class **Machine**. ( vì theo mô tả của NachOS thì thanh ghi số 2 trả ra giá trị trả về của system call ).
   * Bước 4: Giải phóng **buffer** và tăng programcounter
2. Cài đặt system call **char PrintInt(int number)**

* Bước 1: lấy số nguyên từ thanh ghi số 4 ( vì theo mô tả của **NachOS** thì thanh ghi số 4 chính là tham số thứ 1 của system call).
* Bước 2: xử lý trường hợp: **INT\_MIN** và **0**, in ra màn hình
* Bước 3: lưu lại dấu của số và chuyển số thành số dương.
* Bước 4: đếm số chữ số của số nguyên, sau đó khai báo 1 mảng char để lưu chuỗi số
* Bước 5: lặp để chuyển từng ký tự số vào trong mảng
* Bước 6: nếu số là số âm thì gán phần tử đầu tiên của mảng là “-“
* Bước 7: dung hàm **Write** trong class **SynchConsole** để in ra màn hình
* Bước 8: giải phóng mảng và tăng program counter

1. Cài đặt system call **char** **ReadChar**()

* Bước 1: đọc ký tự từ buffer, lấy ký tự bằng hàm **Read(buffer, 1)** từ class **SynchConsole** để nạp chuỗi số vào buffer đồng thời trả ra độ dài của chuỗi số.
* Bước 2: nếu độ dài chuỗi số là -1 tức là chuỗi rỗng tiến hành in ra màn hinh là chuỗi rỗng, kết thúc.
* Bước 3: nếu có 1 kí tự thì tiến hành ghi vào thanh ghi số 2
* Bước 4: xóa buffer và tăng program counter

1. Cài đặt system call **void** **PrintChar(char character)**

* Bước 1: đọc giá trị từ thanh ghi số 4.
* Bước 2: in ra giá trị bằng hàm **Write** từ class **SynchConsole**.
* Bước 3: tăng program counter.

1. Cài đặt System call **void** **ReadString( char[] buffer, int length)**

* Bước 1: lấy địa chỉ chuỗi kí tự từ thanh ghi số 4, và lấy độ dài ‘length’ của chuỗi từ thanh ghi số 5
* Bước 2: chuẩn bị sẵn 1 mảng buffer có độ dài bằng độ dài của chuỗi , đọc chuỗi mà User nhập vào từ bàn phím bằng hàm **Read(buffer, length)** từ class **SynchConsole**.
* Bước 3: dùng hàm **System2User** chuyển chuỗi từ vùng nhớ System sang User
* Bước 4: giải phóng buffer và tăng program counter.

1. Cài đặt system call **void PrintString (char[] buffer)**

* Bước 1: lấy địa chỉ chuỗi kí tự từ thanh ghi số 4 gán vào virtAd
* Bước 2: chuyển chuỗi từ vùng nhớ User sang vùng nhớ System bằng hàm **User2System( virtAd, 255)** và gán vào mảng buffer đã chuẩn bị sẵn
* Bước 3: đếm độ dài của chuỗi
* Bước 4: in chuỗi ra màn hình bằng hàm **Write** từ class **SynchConsole**
* Bước 5: giải phóng buffer và tăng program counter.

1. Cài đặt chương trình help

* Sử dụng system call printString để in ra các dòng giới thiệu và mô tả

1. Cài đặt chương trình ascii

* Sử dụng vòng lặp kết hợp với system call printInt để in giá trị và system call printChar để in kí tự tương ứng với giá trị đó

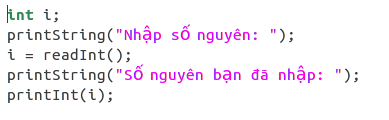
1. Cài đặt chương trình sort (Bubble sort)

* Một số system call cần cài đặt thêm
* readArray
* sử dụng một vòng lặp do while để yêu cầu người dùng nhập số lượng phần tử, sử dụng system call readInt để đọc số lượng phần tử mà người dùng nhập. Nếu số lượng phần tử không hợp lệ (n < 1 hoặc n > 100 và n!= char) thì sai điều kiện vòng lặp và phải nhập lại
* sử dụng vòng for kết hợp với system call readInt để nhập từng phần tử cho mảng
* printArray
* sử dụng vòng lặp for và kết hợp với system call printInt để in từng phần tử của mảng
* swap
* system call được dùng để hoán vị hai số nguyên
* bubbleSort
* Sử dụng vòng lặp for thứ nhất chạy từ phần tử thứ i = 0 đến phần tử thứ i = n-2
* Sử dụng vòng lặp for thứ hai chạy từ phần tử j = 0 đến phần tử thứ j = n- i -2
* Nếu phần tử thứ j > phần tử thứ i thì tiến hành dùng system call swap
* Chương trình
* Bước 1 : Sử dụng system call readArray để nhập số phần tử và từng phần tử của mảng .
* Bước 2: sử dụng system call printArray để in mảng mới nhập
* Bước 3 : Sử dụng system call bubbleSort để sắp xếp mảng tăng dần
* Bước 4 : sử dụng system call print Array để in mảng đã sắp xếp

# III/Test system call

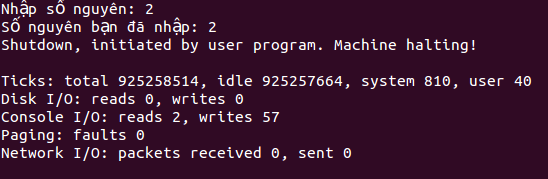
## System call readInt, printInt

### Code



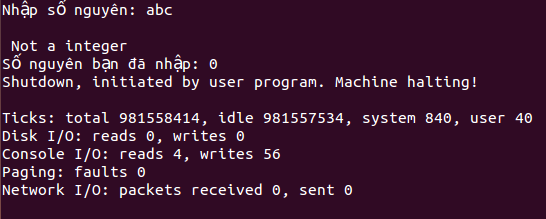
### Một số nguyên dương

* Input: 2
* Output: 2
* Kết quả :



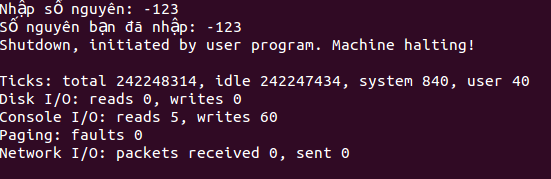
### Một chuổi kí tự

* Input: abc
* Output: 0
* Kết quả



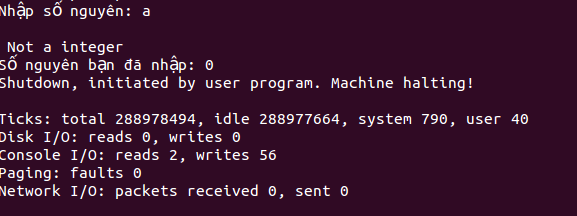
### Một số nguyên âm

* Input: -123
* Output: -123
* Kết quả



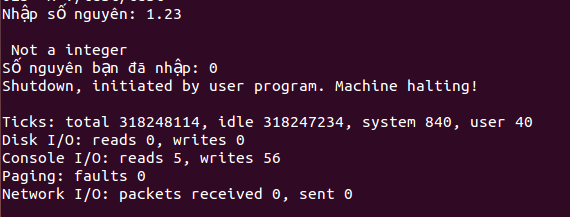
### Một kí tự

* Input: a
* Output: 0
* Kết quả



### Một số thực

* Input: 1.23
* Output: 0
* Kết quả



## System call readChar, printchar

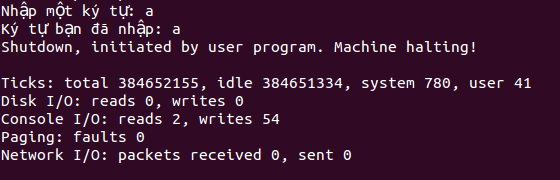
### Code

Ảnh có chứa văn bản

Mô tả được tạo tự động

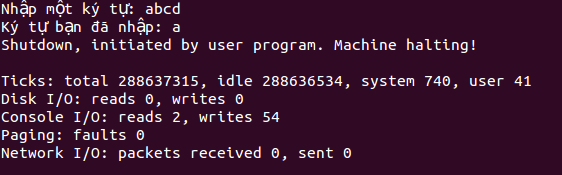
### Một ký tự

* Input: a
* Output: a
* Kết quả



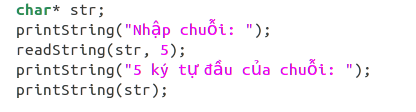
### Nhiều kí tự

* Intput: abcd
* Output: a
* Kết quả



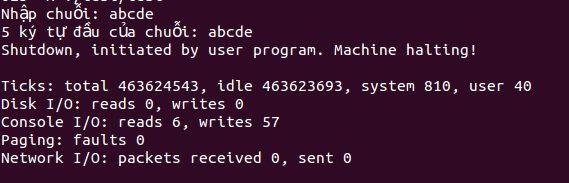
## System call readString ,printString

### Code

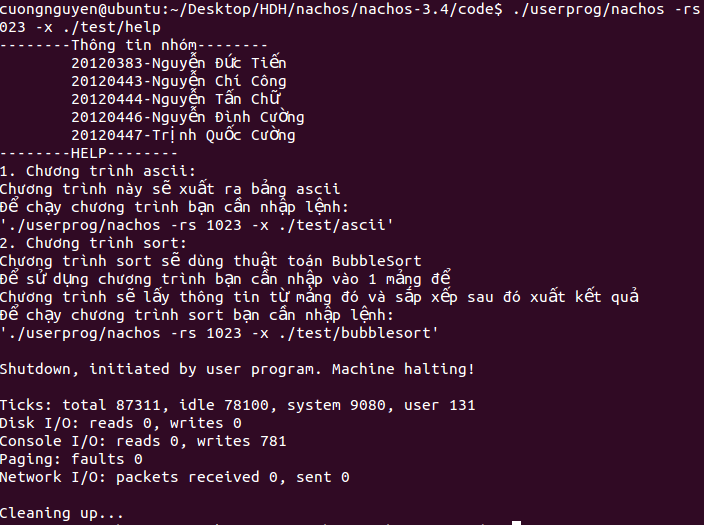


### Chuổi kí tự

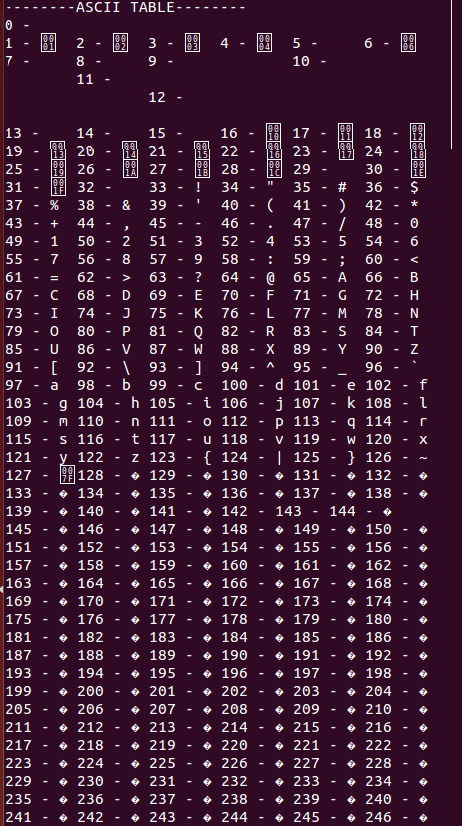
* Input: abcde
* Output: abcde
* Kết quả



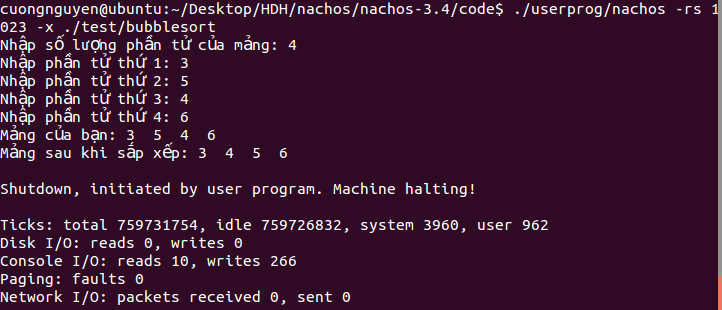
## Chương trình help



## Chương trình ASCII



## Chương trình sort (bubble sort)



# Tài Liệu Tham Khảo

1. Biên dịch và cài đặt NachOS – file được cung cấp bởi giáo viên hướng dẫn.
2. <https://blog.luyencode.net/thuat-toan-sap-xep-bubble-sort/#y-tuong-cua-thuat-toan-bubble-sort>

# link video demo

<https://www.youtube.com/watch?v=0NfgTtdwLa8>