**🙞🙞TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN🙜🙜**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🙣🕮🙡**

**BỘ MÔN HỆ ĐIỀU HÀNH**

ĐỒ ÁN 2:

SYSTEM CALL NACHOS

* Giảng viên: Lê Viết Long
* Sinh viên thực hiện: Võ Hoàng Bảo Duy – 19127027

Trần Ngọc Lam – 19127040

Lê Minh Sĩ – 19127064

C:\Users\tdqua_000\Dropbox\SS-Slides\DeCuong-CDIO\Template CDIO v4.2\Templates\Hinh anh\LogoTruong.png

Khoa Công nghệ Thông tin

Đại học Khoa học Tự nhiên TP HCM

Mục lục

[1. Thông tin chung 3](#_Toc77184315)

[a. Thông tin nhóm 3](#_Toc77184316)

[b. Bảng phân công công việc 3](#_Toc77184317)

[2. Cài đặt tổng quan 4](#_Toc77184318)

[a. Thêm lớp SynchConsole và khởi tạo biến toàn cục 4](#_Toc77184319)

[b. Cài đặt hai thủ tục User2System và System2User 4](#_Toc77184320)

[c. Cài đặt hàm IncreaseProgramCounter() 4](#_Toc77184321)

[d. Cài đặt lại Exception 5](#_Toc77184322)

[e. Hướng dẫn cài đặt System call và chương trình 5](#_Toc77184323)

[3. Cài đặt System Call và Exception 7](#_Toc77184324)

[a. Cài đặt System Call: int ReadInt() 7](#_Toc77184325)

[b. Cài đặt System Call: void PrintInt(int number) 8](#_Toc77184326)

[c. Cài đặt System Call: char ReadChar() 8](#_Toc77184327)

[d. Cài đặt System Call: void PrintChar(char character) 8](#_Toc77184328)

[e. Cài đặt System Call: void ReadString (char[] buffer, int length) 9](#_Toc77184329)

[f. Cài đặt System Call: void PrintString(char[] buffer) 9](#_Toc77184330)

[4. Cài đặt chương trình 10](#_Toc77184331)

[a. Chương trình help 10](#_Toc77184332)

[b. Chương trình ascii 10](#_Toc77184333)

[c. Chương trình sort (Bubble Sort) 10](#_Toc77184334)

[5. Một số hình ảnh demo chương trình 10](#_Toc77184335)

[a. Biên dịch chương trình 10](#_Toc77184336)

[b. Chương trình help 11](#_Toc77184337)

[c. Chương trình ascii 11](#_Toc77184338)

[d. Chương trình sort 12](#_Toc77184339)

[e. Một số chương trình khác kiểm tra các System Calll 12](#_Toc77184340)

[6. Tài liệu tham khảo 14](#_Toc77184341)

# Thông tin chung

## Thông tin nhóm

|  |  |  |
| --- | --- | --- |
| **MSSV** | **Họ Tên** | **Email** |
| *19127027* | Võ Hoàng Bảo Duy | [19127027@student.hcmus.edu.vn](mailto:19127027@student.hcmus.edu.vn) |
| *19127040* | Trần Ngọc Lam | [19127040@student.hcmus.edu.vn](mailto:19127040@student.hcmus.edu.vn) |
| *19127064* | Lê Minh Sĩ | [19127064@student.hcmus.edu.vn](mailto:19127064@student.hcmus.edu.vn) |

## Bảng phân công công việc

|  |  |  |  |
| --- | --- | --- | --- |
| **MSSV** | **Họ Tên** | **Phân công công việc** | **Mức độ hoàn thành** |
| 19127027 | Võ Hoàng Bảo Duy | * Viết báo cáo * Cài đặt tổng quan chương trình và tạo github nhóm * Kiểm tra các lỗi (nếu có) của syscall | 100% |
| 19127040 | Trần Ngọc Lam | * Cài đặt syscall **ReadInt** và **PrintInt** * Viết chương trình **help**, **sort**, **ascii** | 100% |
| 19127064 | Lê Minh Sĩ | * Cài đặt syscall **ReadChar** và **PrintChar** * Cài đặt syscall **ReadString** và **PrintString** | 100% |
| Toàn bộ project | | | 100% |

# Cài đặt tổng quan

## Thêm lớp SynchConsole và khởi tạo biến toàn cục

* Copy 2 tập tin **synchcons.h** và **synchcons.cc** vào thư mục ./code/threads
* Ở thư mục ./code tìm file Makefile.common thực hiện:
  + Cuối đoạn USERPROG\_H thêm **../threads/synchcons.h**
  + Cuối đoạn USERPROG\_C thêm **../threads/synchcons.cc**
  + Cuối đoạn USERPROG\_O thêm **synchcons.o**
* Để hỗ trợ việc nhập/ xuất với màn hình console ta cần sử dụng lớp **SynchConsole**
* Ở thư mục ./code/threads/ thực hiện:
  + Trong file **system.h** ở phần USER\_PROGRAM thêm:

#include "synchcons.h"

extern SynchConsole\* gSynchConsole;

* + Trong file **system.cc:** thực hiện khởi tạo, cấp phát, thu hồi bộ nhớ của biến cục bộ **gSynchConsole**

## Cài đặt hai thủ tục User2System và System2User

* Ở thư **mục ./code/machine** vào file **machine.h** khai báo 2 hàm trong class Machine:
  + **char\* User2System (int virtAddr, int limit)**: có chức năng sao chép vùng nhớ của user sang vùng nhớ của system, kết quả trả về kiểu char\*.
  + **int System2User (int virtAddr, int len, char\* buffer):**  có chức năng sao chép vùng nhớ của system sang vùng nhớ của user, kết quả trả về là số byte đã sao chép được.
* Việc cài đặt 2 hàm trên được thực hiện ở **./code/machine/translate.cc**

## Cài đặt hàm IncreaseProgramCounter()

* Để tránh tình trạng hệ điều hành Nachos bị lặp vô tận khi thực hiện system call ta cần tăng program counter để nạp lệnh tiếp theo trước khi kết quả trả về, công việc này sẽ do hàm **IncreaseProgramCounter** thực hiện.
* Ở thư mục **./code/machine** vào file **machine.h** khai báo hàm trong class Machine
* Việc cài đặt hàm **IncreaseProgramCounter** được thực hiện ở **./code/machine/machine.cc**

## Cài đặt lại Exception

* Ở thư mục ./code/machine vào file **machine.h** để tìm tất cả các exception trong này (hầu hết là lỗi do run-time, không thể phục hồi được).
* Ở thư mục ./code/userprog vào file **exception.cc** thực hiện switch/case với các exception:
  + **No exception:** trả quyền về cho hệ điều hành
  + **SyscallException:** thực hiện switch/ case với từng loại type của System call, sau đó tăng program counter **(machine->IncreaseProgramCounter() )** để nạp lệnh tiếp theo.
  + **Các exception còn lại:** thực hiện việc thông báo người dùng và dừng hệ điều hành (**interrupt -> Halt()**)
* Các system call tiếp theo được cài đặt và thêm các case vào switch/case của **SyscallException.**

## Hướng dẫn cài đặt System call và chương trình

* Nội dung của các thanh ghi:
  + **R2**: lưu mã system call đồng thời lưu kết quả trả về của system call (nếu có).
  + **R4, R5, R6, R7:** lưu tham số thứ nhất, hai, ba, tư.
* Để tiện cho việc hướng dẫn cài đặt system call và chương trình, ở đây sẽ hướng dẫn cài đặt syscall Sub và chương trình Sub.
* Ở thư mục ./code/userprog vào file syscall.h thêm :

#define SC\_Sub 43

int Sub(int a, int b);

* Ở thư mục ./code/userprog vào file exception.cc thêm case SC\_Sub vào SyscallException:

int op1,op2,result;

op1 = machine->ReadRegister (4);

op2 = machine->ReadRegister (5);

result = op1 - op2;

machine->WriteRegister (2, result);

printf("\nSub is tested");

* Ở thư mục **./code/test** trong các file **start.c** và **start.s** thêm các dòng sau:

.globl Sub

.ent Sub

Sub:

addiu $2,$0,SC\_Sub

syscall

j $31

.end Sub

* Ở phía người dùng, ta sẽ viết chương trình để gọi System Call: thư mục **./code/test:**
  + Thêm file **subTest.c** với đoạn mã gọi hàm Sub:

#include "syscall.h"

int main()

{

int a,b,result;

a= 43;

b=23;

result = Sub(a, b);

return 0;

}

Text

Description automatically generated**Lưu ý:** Do sử cross compiler để biên dịch các chương trình ở thư mục Test ở phiên bản khá thấp so với phiên bản hiện tại nên cần chú ý: việc thực hiện đoạn mã như trên khi biên dịch sẽ không báo lỗi, còn đoạn mã dưới đây sẽ báo lỗi khi biên dịch

* + Trong **Makefile:**
    - Thêm **subTest** vào dòng **all**:

all: halt shell matmult sort subTest

* + - Sau đoạn matmult thêm vào đoạn code sau để tạo các file cần thiết để chạy chương trình:

subTest.o: subTest.c

$(CC) $(CFLAGS) -c subTest.c

subTest: subTest.o start.o

$(LD) $(LDFLAGS) start.o subTest.o -o subTest.coff

../bin/coff2noff subTest.coff subTest

* **Biên dịch chương trình:** tại thư mục ./code ta open terminal hoặc cd vào thư mục ./code: **code$ make all**
* **Chạy chương trình**:

**code$ ./userprog/nachos -rs 1023 -x ./test/subTest**

* Sau khi chạy chương trình ta sẽ thấy màn hình hiện dòng **“Sub is tested”** – như thế là việc cài đặt system call đã thành công và chương trình cũng đã được chạy.
* Các system call tiếp theo và chương trình theo yêu cầu cũng được thực hiện như các bước trên.

# Cài đặt System Call và Exception

## Cài đặt System Call: int ReadInt()

* System Call **ReadInt** sẽ sử dụng lớp SynchConsole để đọc một số nguyên do người dùng nhập vào. Nếu giá trị người dùng nhập không phải là số nguyên hay những giá trị không hợp lệ sẽ thì trả về zero (0) và thông báo cho người dùng biết.

***Mô tả cài đặt SC\_ReadInt:***

* **Input:** Không có.
* **Output:** số nguyên người dùng vừa nhập hoặc 0 (đồng thời thông báo lỗi).
* **Mục đích:** dùng để đọc số nguyên do người dùng nhập vào từ màn hình Console.
* Ta dùng biến gSynchConsole (thuộc lớp SynchConSole) để đọc các chữ số từ màn hình console (hàm Read), lưu vào **buffer và** kết quả trả về số bytes thực tế đã đọc được.
* Do phạm vi giá trị kiểu **INTEGER(INT)**: từ -2147483648 đến 2147483647. Tối đa có 11 kí tự nên buffer hợp lệ sẽ có tối đa 11 kí tự: số âm tối đa là 11, số dương tối đa là 10.
* Các trường hợp khiến system call trả về 0 và thông báo lỗi về cho người dùng:
  + Buffer có kích thước không hợp lệ.
  + Buffer có chứa các kí tự khác các chữ số.
  + Số vừa nhập không thuộc phạm vi giá trị của kiểu int.
* Ngược lại, system call sẽ trả về số nguyên vừa nhập (lưu ở thanh ghi số 2) (123.0000 cũng là số nguyên).

## Cài đặt System Call: void PrintInt(int number)

* PrintInt systemcall sẽ sử dụng lớp SynchConsole để xuất một số nguyên ra màn hình.

***Mô tả cài đặt SC\_PrintInt:***

* **Input:** số nguyên kiểu int.
* **Output**: không có.
* **Mục đích:** xuất số nguyên ra màn hình của người dùng.
* Ta thực hiện đọc **number** từ thanh ghi số 4 (r4), sau đó sử dụng các phép biến đổi cơ bản từ kiểu int đưa về kiểu char\* buffer, và dùng biến gSynchConsole (thuộc lớp SynchConsle) để gọi hàm Write với mục đích in ra màn hình một số nguyên.

## Cài đặt System Call: char ReadChar()

* **ReadChar** system call sẽ sử dụng lớp SynchConsole để đọc một ký tự do người dùng nhập vào.
* ***Mô tả cài đặt SC\_ReadChar:***
* Input: không có
* Output: ký tự của người dùng vừa nhập từ màn hình hoặc NULL nhập lỗi.
* Mục đích: dùng để đọc 1 ký tự do người dùng nhập vào từ màn hình console
* Sử dụng hàm Read của lớp SynchConsole để đọc các ký tự từ màn hình và lưu vào buffer.
* Các trường hợp system call thông báo lỗi và trả về NULL:
  + Nhập quá 1 ký tự.
  + Không nhập gì cả.
* Ngược lại, trả về ký tự vừa nhập.

## Cài đặt System Call: void PrintChar(char character)

* **PrintChar** system call sẽ sử dụng lớp SynchConsole để xuất một ký tự ra màn hình.

***Mô tả cài đặt SC\_PrintChar:***

* **Input:** một ký tự kiểu char
* **Output**: không có.
* **Mục đích:** xuất một ký tự ra màn hình của người dùng.
* Ta thực hiện đọc **character** từ thanh ghi số 4 (r4), sau đó dùng biến gSynchConsole (thuộc lớp SynchConsle) để gọi hàm Write với mục đích in ra màn hình ký tự ở input.

## Cài đặt System Call: void ReadString (char[] buffer, int length)

* ReadString system call sẽ sử dụng lớp SynchConsole dùng để đọc một chuỗi ký tự vào trong buffer. Tuy nhiên buffer hiện tại là vùng nhớ của kernel (system), ta cần chuyển buffer này từ vùng nhớ của kernel (system) sang vùng nhớ của người dùng (user)

***Mô tả cài đặt SC\_ReadString:***

* **Input:**
  + Địa chỉ chứa chuỗi nằm ở vùng nhớ của user.
  + Độ dài của chuỗi cần đọc.
* **Output**: không có.
* **Mục đích:** đọc một chuỗi ký tự được nhập từ màn hình và lưu vào vùng nhớ của người dùng.
* Ta thực hiện đọc 2 tham số:
  + Địa chỉ chứa chuỗi nằm ở vùng nhớ của user (virtual address): ở thanh ghi số 4.
  + Độ dài của chuỗi cần đọc (length): ở thanh ghi số 5
* Buffer ban đầu được cấp phát với limit là 255, nên khi length có giá trị lớn 255 thì buffer chỉ nhận 255 ký tự.
* Sử dụng hàm Read của SynchConsole để đọc chuỗi ký tự vừa nhập vào buffer (thuộc vùng nhớ của system), sau đó sử dụng hàm System2User để sao chép vùng nhớ từ system sang vùng nhớ của user.

## Cài đặt System Call: void PrintString(char[] buffer)

* **PrintString** system call dùng để in chuỗi ký tự trong buffer ra màn hình. Tuy nhiên, tương tự như **ReadString** chúng ta cần phải chuyển dữ liệu từ vùng nhớ của user qua vùng nhớ của system.

***Mô tả cài đặt SC\_PrintString:***

* **Input:** Địa chỉ chứa chuỗi nằm ở vùng nhớ của user.
* **Output:** Không có.
* **Mục đích:** In ra chuỗi ký tự trong buffer ra màn hình console.
* Ta thực hiện đọc tham số địa chỉ chứa chuỗi nằm ở vùng nhớ của user ở thanh ghi số 4 (r4). Sử dụng hàm User2System để sao chép vừng nhớ từ user sang vùng nhớ của system với số bytes tối đa là 255 bytes.
* Gọi hàm Write của lớp SynchConsole để in ra từng kí tự trong chuỗi vừa được sao chép, khi nào gặp kí tự kết thúc ‘\0’ hay ‘\n’ thì dừng lại.

# Cài đặt chương trình

## Chương trình help

* Chương trình được viết ở phía người dùng, với chương trình này sử dụng system call **PrintString** để in ra thông tin trường, môn học, giảng viên và thông tin sinh viên tham gia thực hiện đồ án. Kế tiếp in các các mô tả ngăn gọn về chương trình **ascii** và chương trình **sort**.

## Chương trình ascii

* Với chương trình này in ra bảng mã ASCII: với định dạng int – char tương ứng.
* Chương trình sử dụng 2 system call: **PrintInt** và **PrintChar.**

## Chương trình sort (Bubble Sort)

* Chương trình cho phép người dùng nhập vào số phần tử của mảng (<100), các phần tử tương ứng và thực hiện sắp xếp mảng theo thứ tự tăng dần theo thuật toán cơ bản **bubble sort.**
* Chương trình sử dụng các systemcall chính: **ReadInt, PrintInt, PrintChar, PrintString.**

# Một số hình ảnh demo chương trình

## Biên dịch chương trình

Text

Description automatically generated

***Text

Description automatically generatedBiên dịch thành công***

## Chương trình help

Text

Description automatically generated

## Chương trình ascii

Text

Description automatically generatedText

Description automatically generated

## Chương trình sort

## Text Description automatically generatedMột số chương trình khác kiểm tra các System Calll

* **Chương trình PrintInt (System call: ReadInt, PrintInt, PrintString)**

**Các trường hợp nhập số nguyên int hợp lệ:**

**Text

Description automatically generatedText

Description automatically generated**

**Các trường hợp không hợp lệ:**

* Text

  Description automatically generated**Có chứa các ký tự**
* Text

  Description automatically generated**Vượt quá phạm vi lưu trữ của biến int**
* Text

  Description automatically generated**Số thập phân**

# Tài liệu tham khảo

[Dang Khoa: nachOS (dangkhoahome.blogspot.com)](http://dangkhoahome.blogspot.com/p/nachos.html)

Bien\_dich\_va\_Cai\_dat\_Nachos.pdf

Giao\_tiep\_giua\_HDH\_Nachos\_va\_chuong\_trinh\_nguoi\_dung.pdf