ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN 02

HỆ ĐIỀU HÀNH

| Đề tài |

EXCEPTIONS VÀ CÁC SYSTEM CALLS ĐƠN GIẢN

| Thành viên nhóm|

Nguyễn Văn Hào – 20120470

Lê Kim Hiếu – 20120474

Thái Nguyễn Việt Hùng – 20120488

Võ Phi Hùng – 20120489

Nguyễn Cảnh Huy – 20120496

| Giáo viên hướng dẫn |

Ths. Lê Viết Long

Thành phố Hồ Chí Minh – 202

MỤC LỤC

PHẦN 1: HIỂU MÃ CHƯƠNG TRÌNH NACHOS:

I. KHÁI QUÁT VỀ NACHOS

Nachos (Not another completely heuristic operating system) là phần mềm để hướng dẫn giảng dạy về hệ điều hành. Nachos được phát triển tại Đại học Califonia, Berkeley. Nachos được giả theo kiến trúc MIPS với hầu hết các thành phần và chức năng của một máy thật như: thanh ghi, bộ nhớ, bộ xử lý, bộ lệnh, chu kỳ thực thi lệnh, cơ chế ngắt, chu kỳ đồng hồ,…

II. CÀI ĐẶT TỔNG QUAN

1. Cài đặt Nachos

Cài đặt gói Nachos trên Moodle do giáo viên cung cấp và các thành phần sau:

- Môi trường cài đặt: Ubuntu 14.4 (32-bit)
- Trình biên dịch: gcc/g++
- Trong Makefile tại "./code/test" sửa "MAKE = gmake" thành "MAKE = make"
- Trong thư mục code thực thi đoạn script "make all" trên terminal
- Chạy thử chương trình trên Nachos: di chuyển vào thư mục threads thực thi đoạn script "./nachos -rs 1234 -x ./test/halt"
- Nếu chương trình được cài đặt và chạy thành công sẽ được kết quả như sau:

```
canhhuy116@ubuntu:~/Desktop/nachos/nachos-3.4/code$ cd threads/
canhhuy116@ubuntu:~/Desktop/nachos/nachos-3.4/code/threads$ ./nachos -rs 1234 -x
 ../test/halt
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 10, idle 0, system 10, user 0
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

Cleaning up...
canhhuy116@ubuntu:~/Desktop/nachos/nachos-3.4/code/threads$
```

2. Các giá trị thanh ghi

+ R2: Lưu mã syscall đồng thời lưu kết quả trả về của mỗi syscall nếu có.
+ R4: Lưu tham số thứ nhất.
+ R5: Lưu tham số thứ hai.
+ R6: Lưu tham số thứ ba.
+ R7: Lưu tham số thứ tư.

3. Cài đặt SynchConsole:

- Thêm 2 tập tin: synchcons.h và synchcons.cc vào thư mục $/code/threads
- Mở file Makefile.common (nachos-3.4/code). Thêm các script:
    - USERPROG_H: "../threads/synchcons.h"
    - USERPROG_C: "../threads/synchcons.cc"
    - USERPROG_O: "synchcons.o"

Chức năng: Hỗ trợ nhập xuất từ màn hình console. Để nhập và xuất dữ liệu từ màn hình console ta phải sử dụng lớp **SynchConsole** được khởi tạo qua biến toàn cục gSynchConsole. Gồm 2 hàm chính:
- int Read(char *into, int numBytes): Cho phép nhập một chuỗi từ màn hình console và lưu biến thuộc vùng nhớ hệ điều hành Nachos.
- int Write(char *from, int numBytes): Cho phép xuất một chuỗi từ biến thuộc vùng nhớ hệ điều hành Nachos ra màn hình console.

4. Các bước cài đặt system call

Bước 1: /code/userprog/syscall.h
• Define syscall code để dùng trong switch-case để xử lý exception
• Khai báo prototype của hàm

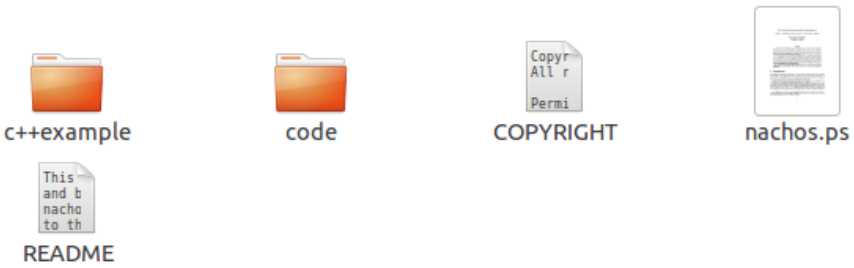
Bước 2: ./code/test/start.s thêm các lệnh của tiêu chuẩn MIPS
• Hỗ trợ ngôn ngữ hợp ngữ để thực hiện các cuộc gọi hệ thống tới Nachos kernal.
• Đặt mã cho lệnh gọi hệ thống vào thanh ghi r2 và để lại các đối số cho lệnh gọi hệ thống (nói cách khác, arg1 trong r4, arg2 trong r5, arg3 trong r6, arg4 ở trong r7)

Bước 3: ./code/userprog/exception.cc xử lí các exception handle

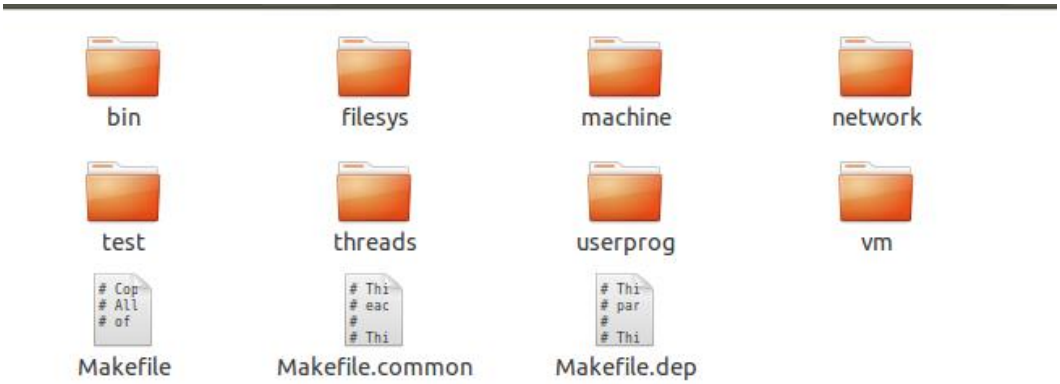
Bước 4: Viết file .c ở mức người dùng trong ./code/test sử dụng hàm như đã khai báo prototype ở code/userprog/syscall.h.

Bước 5: ./code/test/Makefile để xây dựng các chương trình người dùng chạy trên NachOS link các file object, biên dịch thành file thực thi duy nhất.

III. TÌM HIỂU CÁC FILE TRONG NACHOS



Ta cần quan tâm đến thư mục code (là thư mục chính trong nachos). Dưới đấy là các file và thư mục con của code



1. Thư mục bin

Coff.h	COFF là định dạng tệp đối tượng tiêu chuẩn được GCC sử dụng.
Noff.h	NOFF là định dạng tệp đối tượng tiêu chuẩn được Nachos sử dụng
Coff2noff	biến các tệp coff thành định dạng noff

2. Thư mục filesys

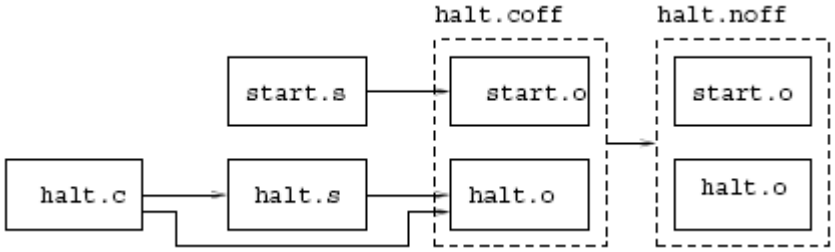
filesys.h	Định nghĩa các lớp để biểu diễn file hệ thống
openfile.h	Định nghĩa các hàm trong hệ thống file Nachos

3. Thư mục machine

mipssim.cc	Mô phỏng tập lệnh của MIPS R2/3000 processor
mipssim.h	Định nghĩa bảng opcodes và decoding
machine	Mô phỏng các thành phần của máy tính khi thực thi chương trình người dùng: bộ nhớ chính, thanh ghi,...

console	Mô phỏng thiết bị đầu cuối sử dụng UNIX files. Một thiết bị có các đặc tính: đơn vị dữ liệu theo byte, đọc và ghi các bytes cùng một thời điểm, các bytes đến bất đồng bộ
translate	Mỗi địa chỉ ảo được giả sử giống như địa chỉ vật lý, điều này giới hạn chúng ta chỉ chạy 1 chương trình tại một thời điểm.
timer	Định nghĩa lớp Timer và mô phỏng bộ đếm thời gian phần cứng bằng cách tạo ra CPU ngắt mỗi TimerTicks
interrupt	Duy trì các cấu trúc để biểu diễn ngắt, quản lý thời gian mô phỏng và xác định mức ngắt

4. Thư mục test
   các chương trình C sẽ được biên dịch theo MIPS và chạy trong Nachos. Test/start.s cài đặt mã hợp ngữ sơ khai cho tất cả các lệnh gọi hệ thống trong userprog/syscall.h và có thể thay thế cho C trong đồ án này



5. Thư mục userprog

protest.cc	Kiểm tra các thủ tục để chạy chương trình người dùng
syscall.h	System call interface: các thủ tục ở kernel mà chương trình người dùng có thể gọi
exception.cc	Xử lý system call và các exception khác ở mức user, ví dụ như lỗi trang,…
bitmap	Các hàm xử lý cho lớp bitmap (hữu ích cho việc lưu vết các ô nhớ vật lý)

PHẦN 2: HIỂU THIẾT KẾ

Mỗi chương trình trong hệ thống bao gồm thông tin cục bộ của nó: program, counters, registers, stack poiner và file system handler. Mặc dù user program truy cập các thông tin cục bộ của nó nhưng HDH điều khiển các truy cập này, HDH đảm bảo các yêu cầu từ user program tới kernel không làm cho HDH sụp đổ. Việc chuyển quyền điều khiển từ user mode thành system mode được thực hiện thông qua systemcall, sofware interrupt.

Trước khi gọi một lệnh trong hệ thống thì các tham số cần thiết phải được nạp vào các thanh ghi của CPU. Để chuyển một biến mang giá trị, tiến trình chỉ việc ghi các giá trị vào thanh ghi, Để chuyển một biến tham chiếu thì giá trị lưu trong thanh ghi được xem như là user space pointer. Ta cần chuyển nội dung từ user space vào kernel để có

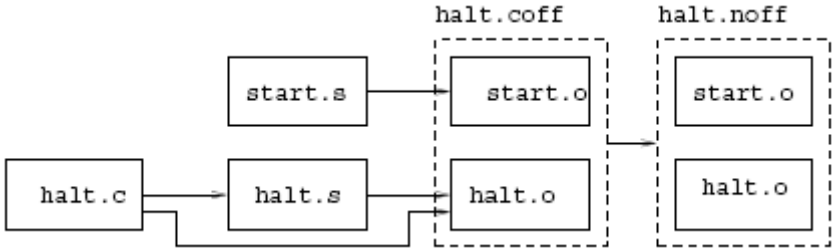
thể xử lý dữ liệu này. Khi trả thông tin từ system về user space các giá trị phải đặt trong các thành ghi của CPU.

Nachos cung cấp một CPU giả lập, thực tế CPU giả lập này giống hệt CPU thật (MIPS chip), nhưng chúng ta không thể chỉ thực thi chương trình như một tiến trình bình thường của UNIX, bởi vị chúng ta muốn kiểm soát có bao nhiêu lệnh được thực hiện trong một đơn vị tgian, không gian địa chỉ làm việc như thế nào, các interrupt và exception(system calls) được xử lý như thế nào.

Nachos cung cấp môi trường giả lập để chạy các chương trình bằng C, xem Makefile trong thư mục test, chương trình biên dịch phải link với vài flag, sao đó chuyển sang định dạng đặc biệt của Nachos, dùng CT "coff2noff"

Quá trình thực thi các system call trên NachOS

Để thực thi được các program hay system call thì đầu tiên nó cần phải được biên dịch. Quá trình biên dịch gồm 3 giai đoạn, ta dùng minh họa dưới đây



1. Halt.c được biên dịch bằng cross-complier gcc (đã được cài đặt trên máy) thành file halt.s là mã assembly chạy trên kiến trúc MIPS
2. Halt.s sẽ được liên kết với start.s để tạo ra định dạng COFF là halt.coff
3. Halt.coff sẽ dùng coff2noff (trong thư mục $code/bin/) để chuyển sang halt.noff, định dạng thực thi trên hệ điều hành Nachos sử dụng kiến trúc MIPS

Sau khi biên dịch thành công ta được tập tin halt.noff lúc này ta có thể thực thi system call thông qua terminal đã hướng dẫn ở Phần 1.

PHẦN 3: CÀI ĐẶT EXCEPTIONS VÀ SYSTEM CALLS

I. CÀI ĐẶT LẠI CÁC EXCEPTIONS

Ta lấy danh sách các system calls trong file "machine.h", từ đó viết lệnh xử lý các ExceptionType trong file "exception.cc". Nếu là NoException, ta kết thúc chương trình mà không làm gì cả. Đối với SyscallException, ta xử lí theo từng loại cụ thể. Các exceptions còn lại ta in ra thông báo lỗi và halt.

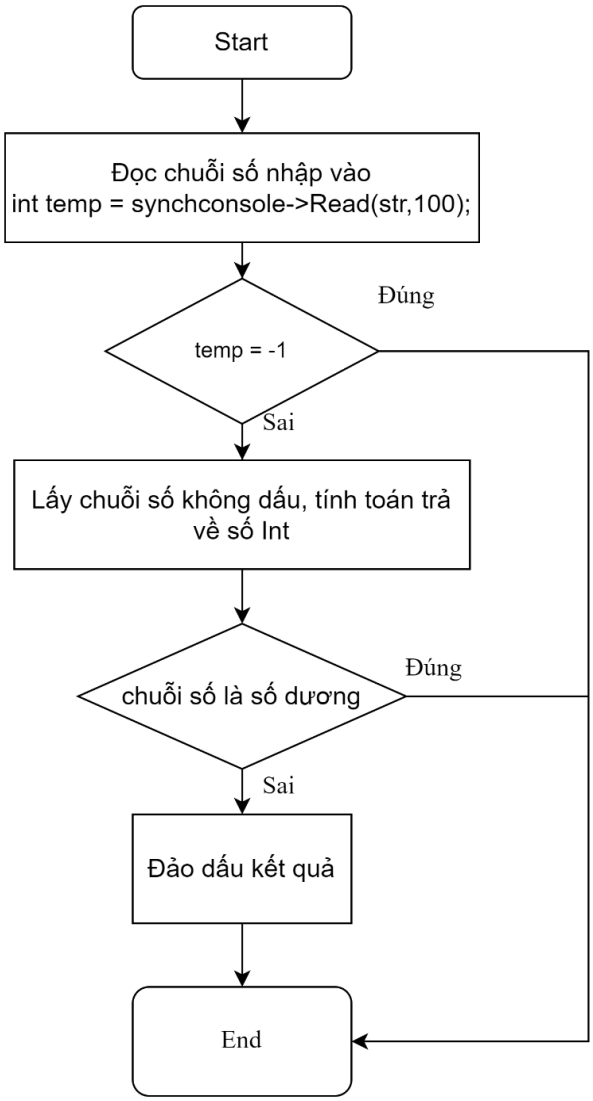
II. CẶT ĐẶT HÀM TĂNG PROGRAM COUNTER

Để Nachos không bị lặp gọi một system call mãi mãi, ta cần tăng program counter trước khi system call trả kết quả về. Ta lần lượt lưu PC trước là PC hiện tại, PC hiện tại là PC kế tiếp, còn PC kế tiếp được cộng thêm 4.

III. CÀI ĐẶT CÁC SYSTEM CALLS

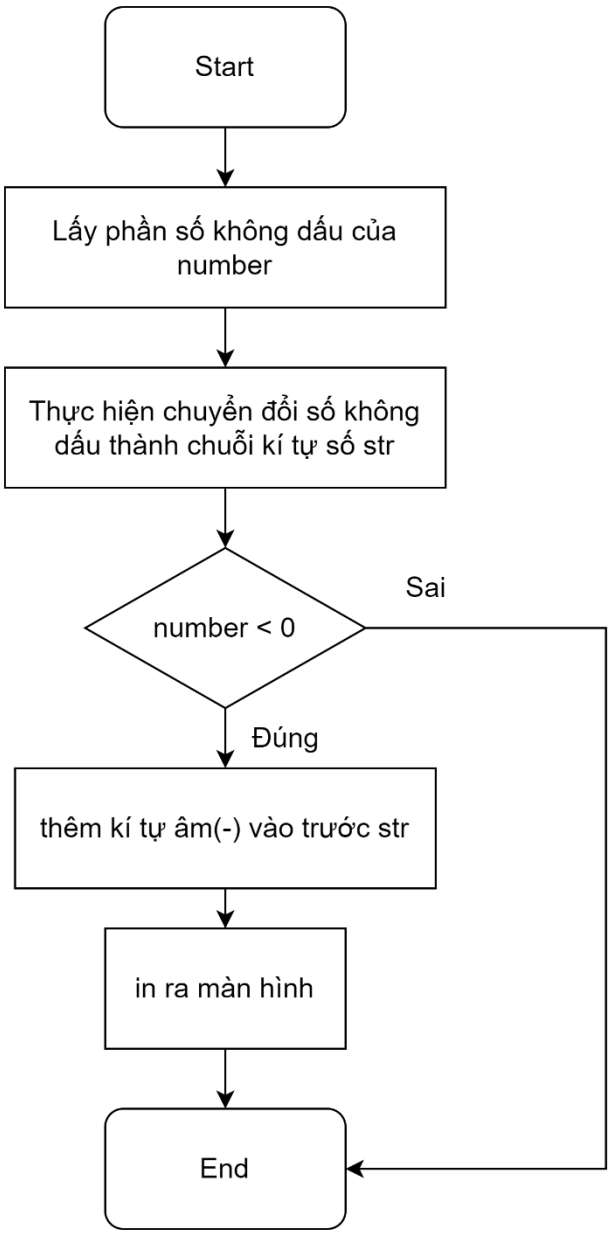
Cài đặt các system calls theo các bước đã được trình bày ở trên.

1. **Cài đặt system call *int ReadInt()***
   - Chức năng: sử dụng lớp SynchConsole để đọc một số nguyên do người dùng nhập vào.
   - Khai báo ở "syscall.h"
   - Cài đặt ở "exception.cc"
   - Sơ đồ thuật toán:



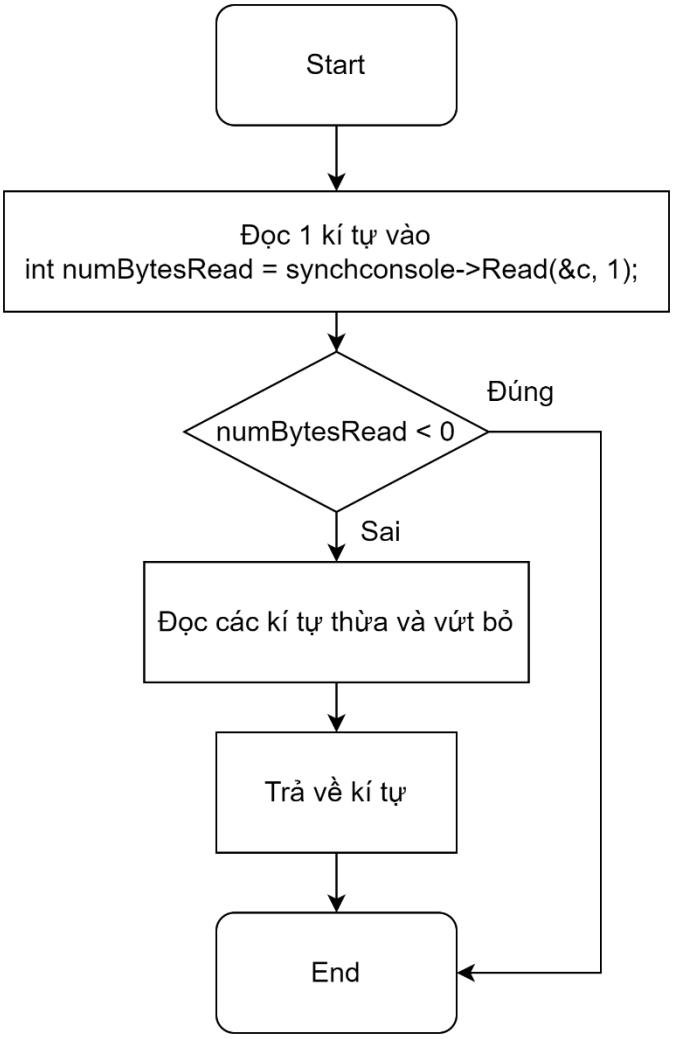
2. **Cài đặt system call *void PrintInt(int number)***
   - Chức năng: sử dụng lớp SynchConsole để xuất một số nguyên ra màn hình.
   - Khai báo ở "syscall.h"
   - Cài đặt ở "exception.cc"

- Sơ đồ thuật toán:

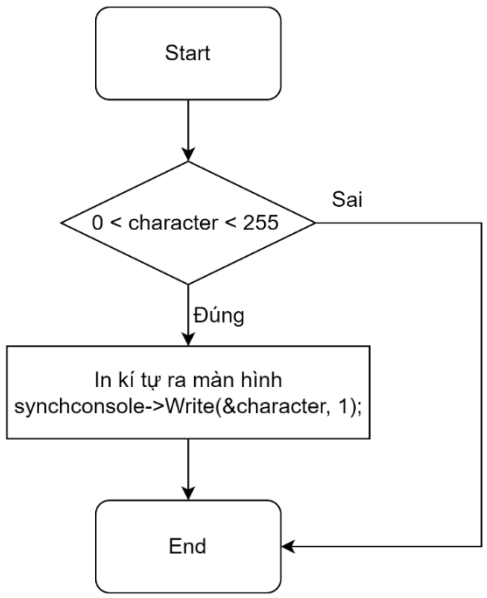


3. Cài đặt system call *char ReadChar()*

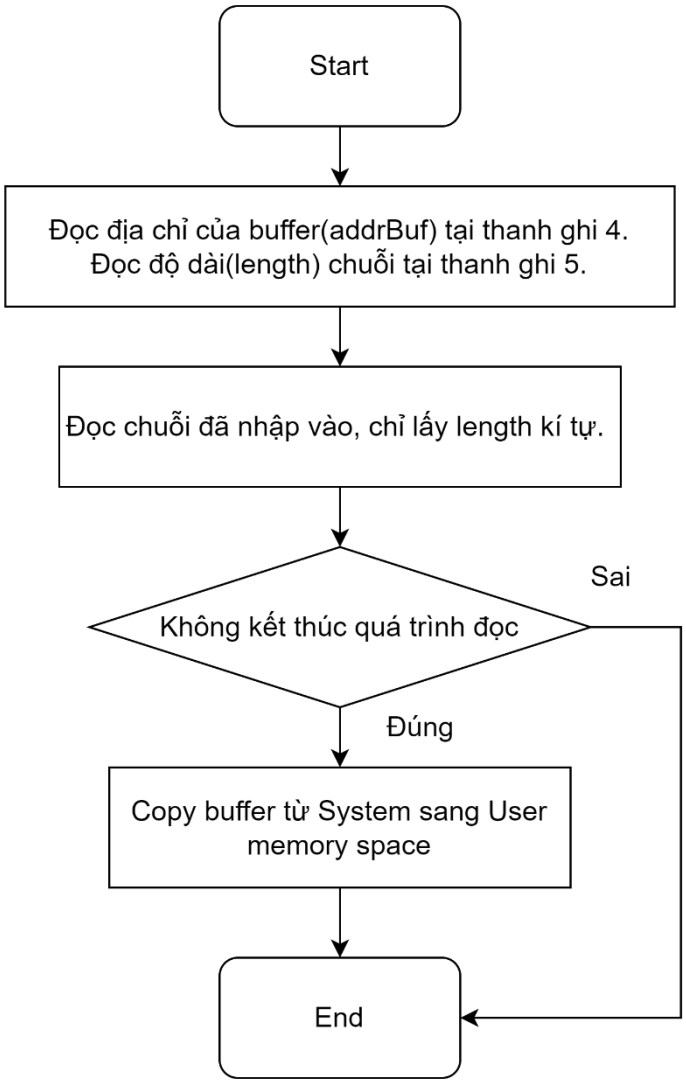
- Chức năng: sử dụng lớp SynchConsole để đọc một ký tự do người dùng nhập vào.
- Khai báo ở “syscall.h”
- Cài đặt ở “exception.cc”
- Sơ đồ thuật toán:



4. **Cài đặt system call *void PrintChar(char character)***
   - Chức năng: sử dụng lớp SynchConsole để xuất một ký tự ra màn hình.
   - Khai báo ở “syscall.h”
   - Cài đặt ở “exception.cc”
   - Sơ đồ thuật toán:

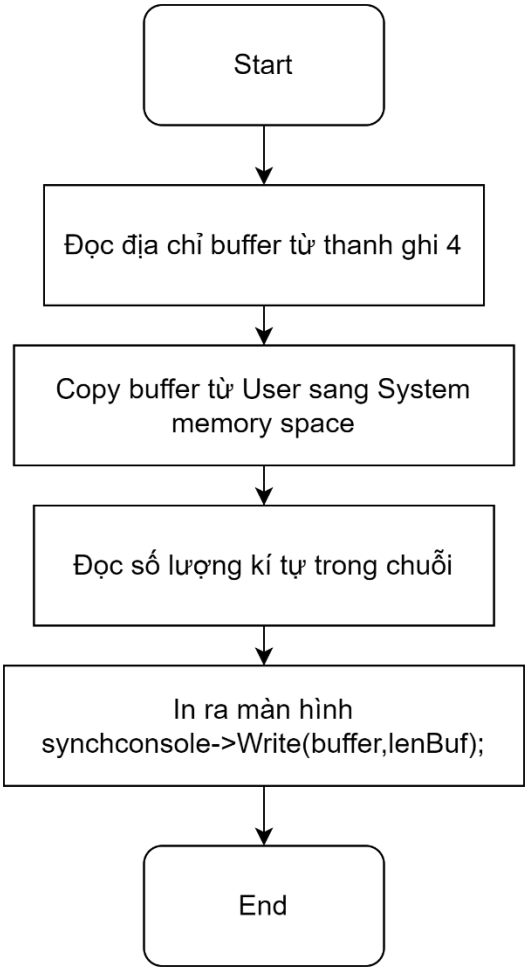


5. **Cài đặt system call *void ReadString(char *buffer, int length)***
   - Chức năng: sử dụng lớp SynchConsole để đọc một chuỗi ký tự vào trong buffer (chuỗi sẽ kết thúc khi người dùng ấn **Enter** hoặc có chiều dài lớn hơn hoặc bằng **length**). Vì buffer là vùng nhớ thuộc userspace, khi người dùng nhập chuỗi thì nội dung được lưu trữ ở kernelspace nên ta sẽ viết hàm **System2User** để chuyển dữ liệu từ kernelspace sang userspace (được cung cấp trong file hướng dẫn thực hành).
   - Khai báo ở "syscall.h"
   - Cài đặt ở "exception.cc"
   - Sơ đồ thuật toán:



6. **Cài đặt system call *void PrintString(char *buffer)***
   - Chức năng: sử dụng lớp SynchConsole để xuất chuỗi ký tự trong buffer ra màn hình. Tương tự như **ReadString** ta có hàm **User2System** để chuyển dữ liệu từ userspace sang kernelspace (được cung cấp trong file hướng dẫn thực hành).
   - Khai báo ở "syscall.h"

- Cài đặt ở “exception.cc”
- Sơ đồ thuật toán:



IV. CÀI ĐẶT CÁC CHƯƠNG TRÌNH CƠ BẢN

1. Chương trình Help

- Chức năng: hiển thị mô tả ngắn gọn về nhóm và các chức năng khác như **ascii**, **BubbleSort**.
- Cài đặt:
  - Viết chương trình thực thi trong “code/test/Help.c”
  - Sửa lại Makefile trong thư mục “code”
  - Biên dịch lại Nachos. Để chạy chương trình, ta nhập câu lệnh: ***./userprog/nachos -rs 1023 -x ./test/Help***

2. Chương trình ascii

- Chức năng: in ra bảng mã ascii.
- Cài đặt:
  - Viết chương trình thực thi trong “code/test/ascii.c”
  - Sửa lại Makefile trong thư mục “code”
  - Biên dịch lại Nachos. Để chạy chương trình, ta nhập câu lệnh: ***./userprog/nachos -rs 1023 -x ./test/ascii***

3. Chương trình BubbleSort

- Chức năng: BubbleSort sẽ cho người dùng nhập số lượng phần tử trong mảng, sau đó người dùng sẽ nhập từng giá trị trong mảng đó và kết quả trả về là mảng được sắp xếp theo chiều tăng dần của mảng người dùng nhập.
- Cài đặt:
    - Viết chương trình thực thi trong "code/test/BubbleSort.c"
    - Sửa lại Makefile trong thư mục "code"
    - Biên dịch lại Nachos. Để chạy chương trình, ta nhập câu lệnh: ***./userprog/nachos -rs 1023 -x ./test/BubbleSort***

MỘT SỐ HÌNH ẢNH MINH HOẠ

```
phihungtf@phihungtf:~/hcmus-nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/Help
My group has 5 people
20120470 - Nguyen Van Hao
20120474 - Le Kim Hieu
20120488 - Thai Nguyen Viet Hung
20120489 - Vo Phi Hung
20120496 - Nguyen Canh Huy

ReadPrintInt: In directory, enter './userprog/nachos -rs 1023 -x ./test/ReadPrintInt' to run program
ReadPrintChar: In directory, enter './userprog/nachos -rs 1023 -x ./test/ReadPrintChar' to run program
ReadPrintString: In directory, enter './userprog/nachos -rs 1023 -x ./test/ReadPrintString' to run program
ASCII: In directory, enter './userprog/nachos -rs 1023 -x ./test/ascii' to run program
BubbleSort: In directory, enter './userprog/nachos -rs 1023 -x ./test/BubbleSort' to run program
Shutdown, initiated by user program.
Machine halting!

Ticks: total 72516, idle 65000, system 7420, user 96
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 650
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
```

Hình 1: Chạy chương trình Help

```
phihungtf@phihungtf:~/hcmus-nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/halt
Shutdown, initiated by user program.
Machine halting!

Ticks: total 42, idle 0, system 30, user 12
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

Cleaning up...
```

Hình 2: Chạy chương trình halt

```
phihungtf@phihungtf:~/hcmus-nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/BubbleSort
Enter the number size of array [0,50] n = 5
Enter arr[0] = 6
Enter arr[1] = 3
Enter arr[2] = 2
Enter arr[3] = 9
Enter arr[4] = 3
2 3 3 6 9 Shutdown, initiated by user program.
Machine halting!

Ticks: total 313167923, idle 313164563, system 2340, user 1020
Disk I/O: reads 0, writes 0
Console I/O: reads 12, writes 127
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
```

Hình 3: Chạy chương trình BubbleSort

Hình 4: Chạy chương trình ascii

TÀI LIỆU THAM KHẢO

1. Các files tài liệu hướng dẫn thực hành của Trường đại học Khoa học Tự nhiên
2. Các clip về Nachos của anh Thành Chung: https://www.youtube.com/watch?v=t0jtY1C129s&list=PLRgTVtca98hUgCN2_2vzsAAXPiTFbvHpO