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 2 – HỆ ĐIỀU HÀNH

**CÁC SYSCALL NHẬP/XUẤT FILE,**

**ĐA CHƯƠNG, LẬP LỊCH VÀ ĐỒNG BỘ HÓA TRONG NACHOS**

*Thành phố Hồ Chí Minh, tháng 1 năm 2022*

# I – Thông tin nhóm và đánh giá

## Thông tin nhóm

|  |  |  |
| --- | --- | --- |
| STT | MSSV | Họ và tên |
| 1 | 19120390 | Trịnh Thị Thùy |
| 2 | 19120580 | Lê Đức Minh |
| 3 | 19120685 | Võ Ngọc Tín |

Mức độ hoàn thành của đồ án: 100%

## Đánh giá

* Những điều đã làm được
* Những điều chưa làm được

# II – Các system call nhập/ xuất file

## Cài đặt tổng quan

*Cách cài đặt các system call tương tự các bước như đồ án 1.*

* Bước 1: Define các syscall và khai báo prototype của hàm.
* Bước 2: Định nghĩa hàm trong file assembler, thêm các dòng sau vào file ./code/test/start.c và ./code/test/start.s .
* Bước 3: Trong ./code/userprog/exception.cc cài đặt các syscall theo yêu cầu.
* Bước 4: Viết các chương trình người dùng kiểm tra vào thư mục./code/test.
* Bước 5: Trong ./code/test/Makefile, thêm tên chương trình vào dòng: all: halt shell matmult sort.
* Bước 6: Biên dịch.
* Bước 7: Test syscall.

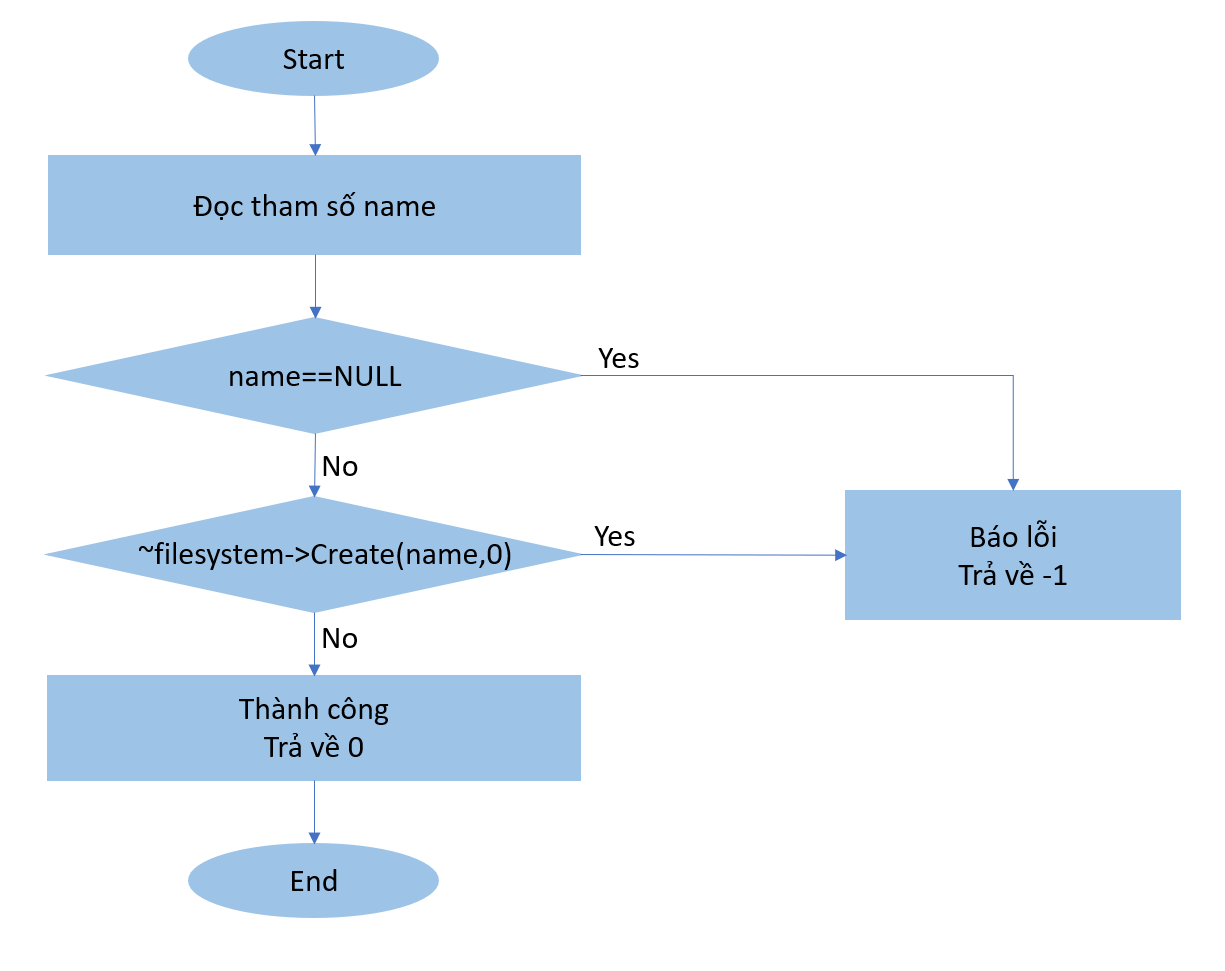
## Syscall: int CreateFile(char \*name)

* Yêu cầu:

+ Input: địa chỉ chứa tên file thuộc user space

+ Output: trả về 0 nếu thành công, -1 nếu thất bại.

* Cách thức cài đặt: Ở thanh ghi r4 lưu địa chỉ tham số name là tên file cần tạo, sao chép giá trị ở thanh ghi r4 từ user space sang system space bằng User2System(). Kiểm tra các điệu kiện như: tên file khác NULL, … Nếu tạo file thành công, trả về thanh ghi r2 giá trị 0, ngược lại trả về giá trị -1.

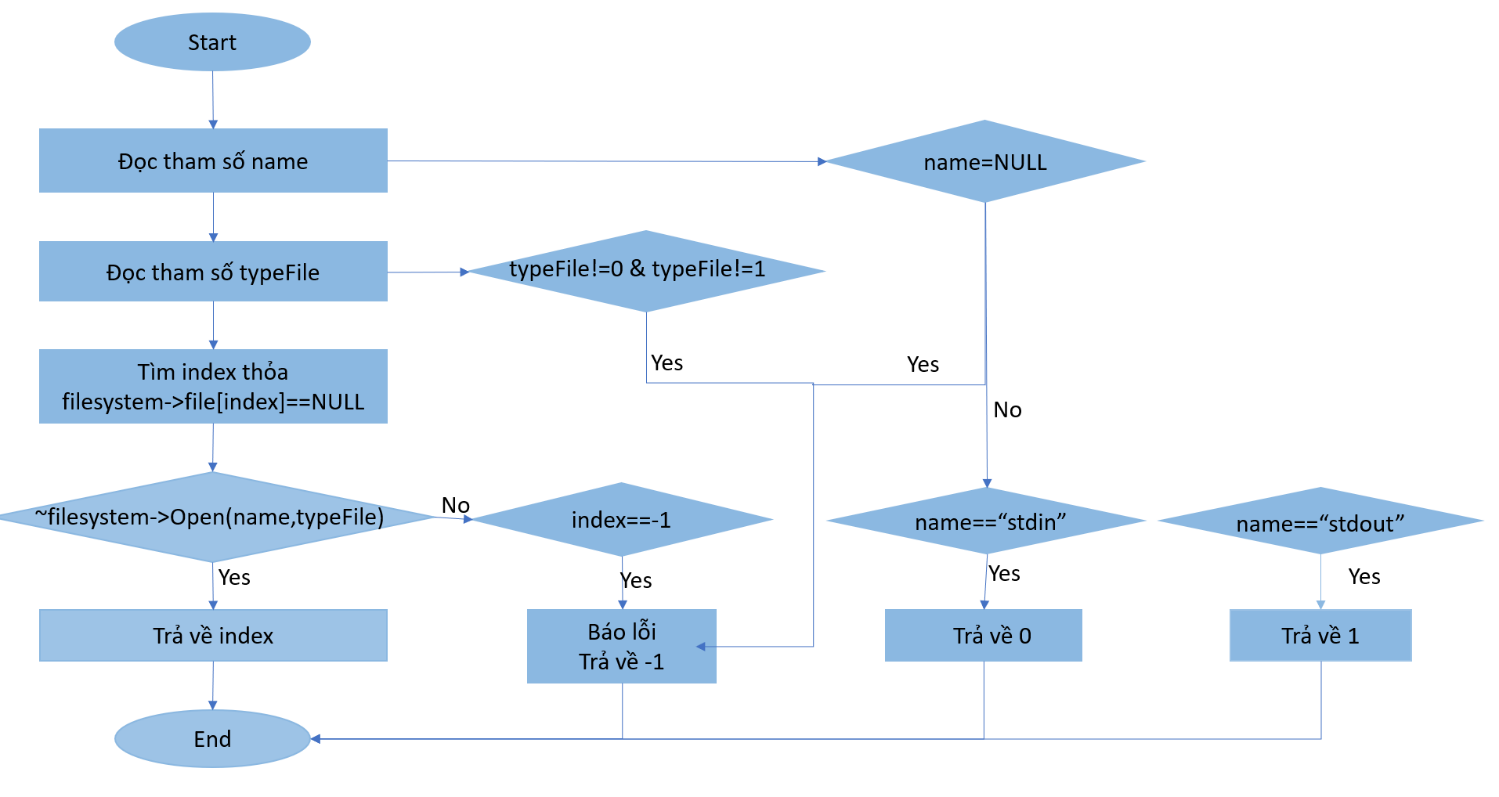
****

## Syscall: OpenFileID Open(char \*name, int type)

* Yêu cầu:

+ Input: địa chỉ tên file thuộc user space, chế độ cần thao tác (type)

+ Output: trả về id file nếu thành công, -1 nếu thất bại hoặc lỗi.

* **** Cách thức cài đặt: Ở thanh ghi r4, r5 lưu địa chỉ tham số name và type, kiểm tra giá trị type hợp lệ, kiểm tra file có tồn tại, có đủ ô nhớ trong bảng mô tả file không, … Nếu tất cả hợp lệ, sao chép giá trị ở r4 từ user space sang system space bằng User2System(). Nếu mở file thành công, trả về thanh ghi r2 id file, ngược lại trả về -1.

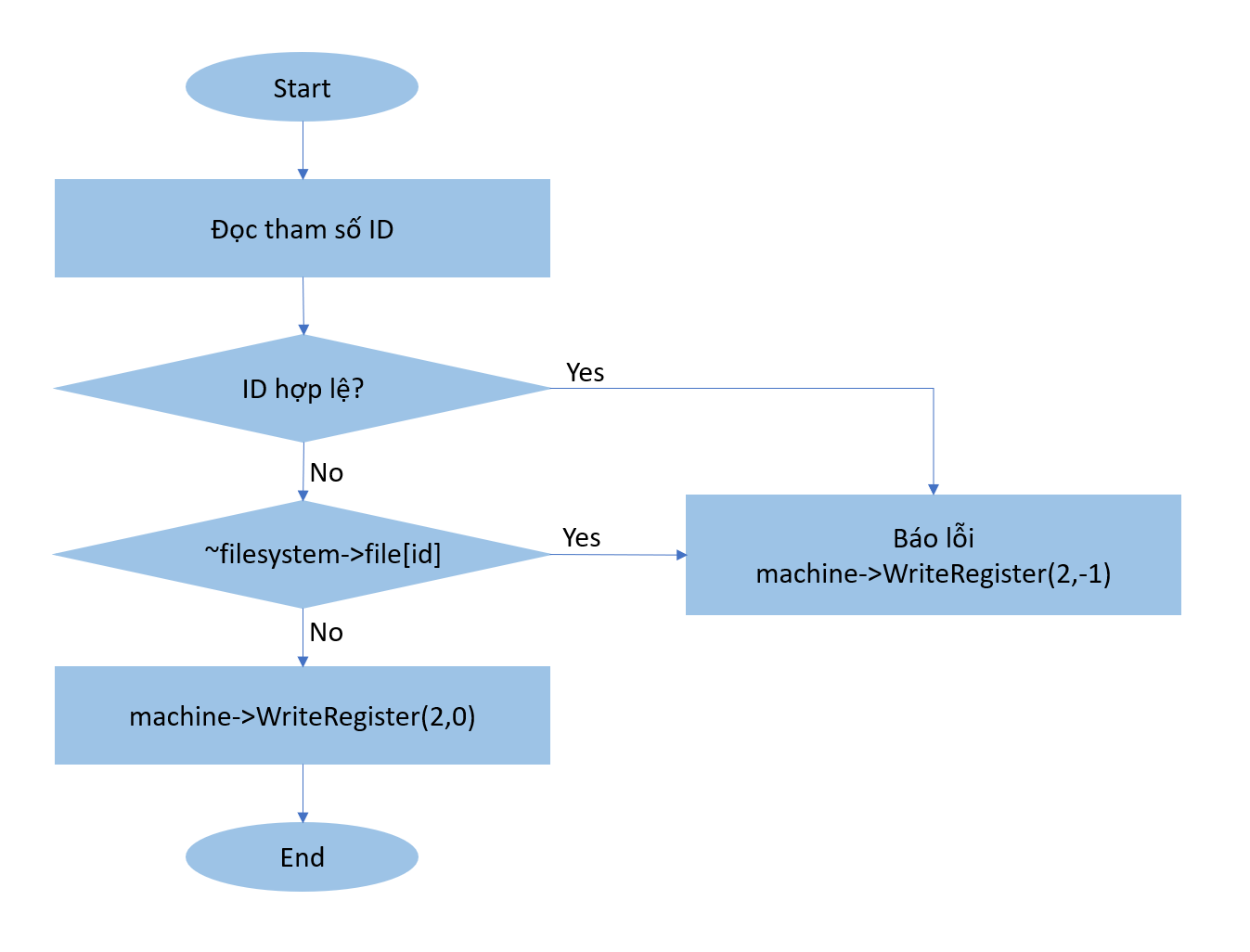
## Syscall: int Close(OpenFileID id)

* Yêu cầu:

+ Input: ID của file (một số nguyên)

+ Output: trả về 0 nếu thành công, -1 nếu thất bại hoặc lỗi.

* Cách thức cài đặt: Từ thanh ghi r4, đọc tham số id, kiểm tra xem file tồn tại bằng cách sử dụng FileSystem Class, kiểm tra id nằm trong bảng mô tả file không. Nếu kiểm tra hợp lệ, xóa đi dữ liệu file tại id đó và trả về thanh ghi r2 giá trị 0, ngược lại báo lỗi và trả về thanh ghi r2 giá trị -1.

****

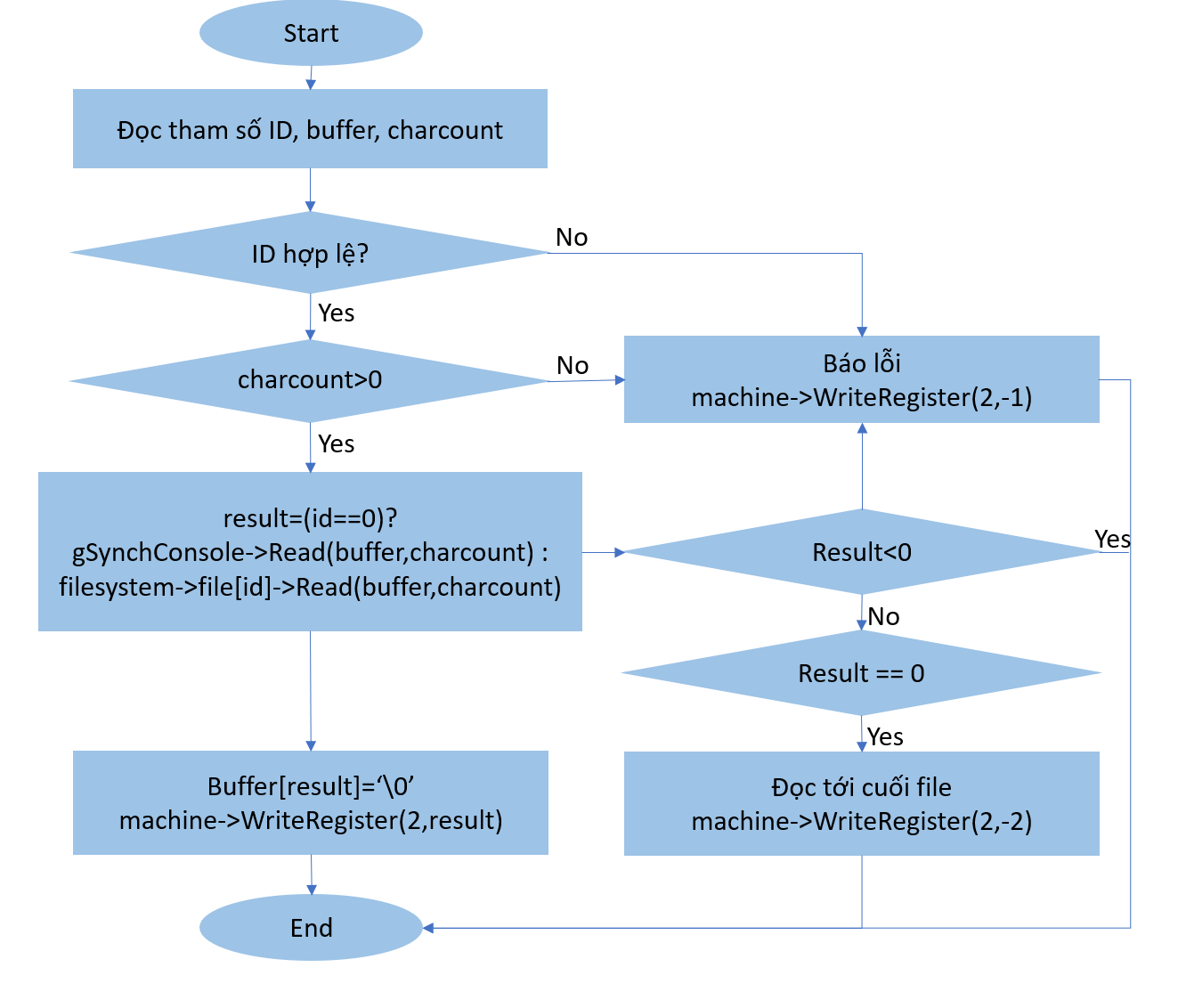
## Syscall: int Read(char \*buffer, int charcount, OpenFileID id)

* Yêu cầu:

+ Input: buffer, số ký tự cần đọc charcount và id file (số nguyên)

+ Output: trả về số ký tự đọc được nếu thành công, -1 nếu lỗi, -2 nếu đọc tới cuối file

* Cách thức cài đặt: Từ thanh ghi r4, r5, r6 đọc lần lượt 3 tham số buffer, charcount, id. Kiểm tra file có tồn tại, id file có nằm trong bảng mô tả file, kiểm tra charcount hợp lệ (>0), … Nếu kiểm tra hợp lệ, sử dụng fileSystem object để đọc file với độ dài biết trước và trả về thanh ghi r2 số ký tự đọc được, nếu đã tới cuối file trả về -2, ngược lại báo lỗi trả về -1.



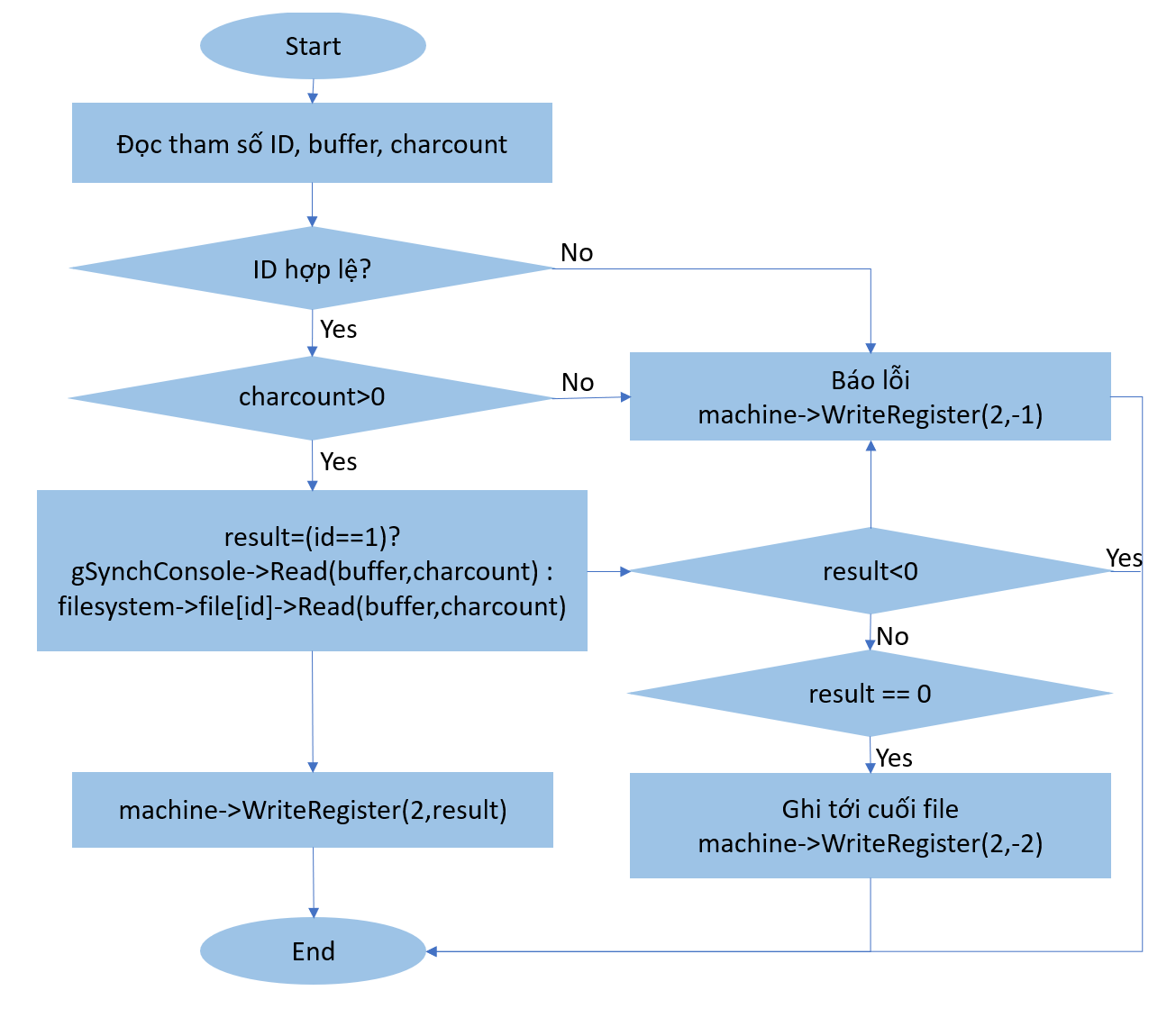
## Syscall: int Write(char \*buffer, int charcount, OpenFileID id)

* Yêu cầu:

+ Input: buffer, số ký tự cần ghi charcount và id file (số nguyên)

+ Output: trả về số ký tự ghi được nếu thành công, -1 nếu lỗi, -2 nếu ghi tới cuối file.

* Cách thức cài đặt: Tương tự như system call Read, từ thanh ghi r4, r5, r6 đọc 3 tham số và kiểm tra các điều kiện. Nếu kiểm tra hợp lệ, nếu openFileId là 1, sử dụng biến gSynchConsole để ghi, ngược lại dùng fileSystem. Sau khi ghi thành công, trả về thanh ghi r2 giá trị là số ký tự ghi được, ngược lại trả về -1 nếu thất bại hoặc -2 nếu đã tới cuối file.



# III – Đa chương, lập lịch và đồng bộ hóa trong Nachos

## Cài đặt tổng quan

***Cà đặt đa tiến trình:*** cần thay đổi trong file addrspace.h và addrspace.cc để chuyển hệ thống đơn chương sang đa chương:

* Sử dụng biến toàn cục Bitmap\* gPhysPageBitMap quản lý các frames.
* Xử lý giải phóng bộ nhớ khi user program kết thúc.
* Tạo pageTable = new TranslationEntry[numPages], tìm trang trống bằng phương thức Find() của lớp Bitmap, nạp chương trình lên bộ nhớ chính

***Thiết kế đa chương:*** cần cài đặt 4 lớp

* PCB: lưu các thông tin quản lý proces.
* Ptable: quản lý các tiến trình chạy được, gồm thuộc tính pcb là một bảng mô tả tiến trình, có cấu trúc mảng một chiều (tối đa 10).
* Stable: gồm thuộc tính semTab là một bảng mô tả các semaphore, có cấu trúc mảng một chiều (tối đa 10).
* Sem: quản lý các semaphore.

*Tiến hành:*

* Bước 1: khai báo các biến toàn cục trong /threads/system.h và tạo các object trong file system.cc

// file system.h

extern Semaphore \*addrLock;

extern BitMap \*gPhysPageBitMap;

extern Stable\* semTab;

extern PTable \*pTab;

. // file system.cc

addrLock = new Semaphore("AddrLock", 1);

gPhysPageBitMap = new BitMap(NumPhysPages);

semTab = new STable();

pTab = new PTable(10);

PTable \*pTab;

* Bước 2: Cài đặt lớp PCB và Ptable, Stable đồng thời khai báo trong file “Makefile.common”.

THREAD\_H =…

../threads/pcb.h\

../threads/ptable.h\

../threads/stable.h\

THREAD\_C =…

../threads/pcb.cc\

../threads/ptable.cc\

../threads/stable.cc\

THREAD\_O =… pcb.o stable.o ptable.o …

* Bước 3: Tiến trình cài đặt các system call theo các bước tương tự phần 1.

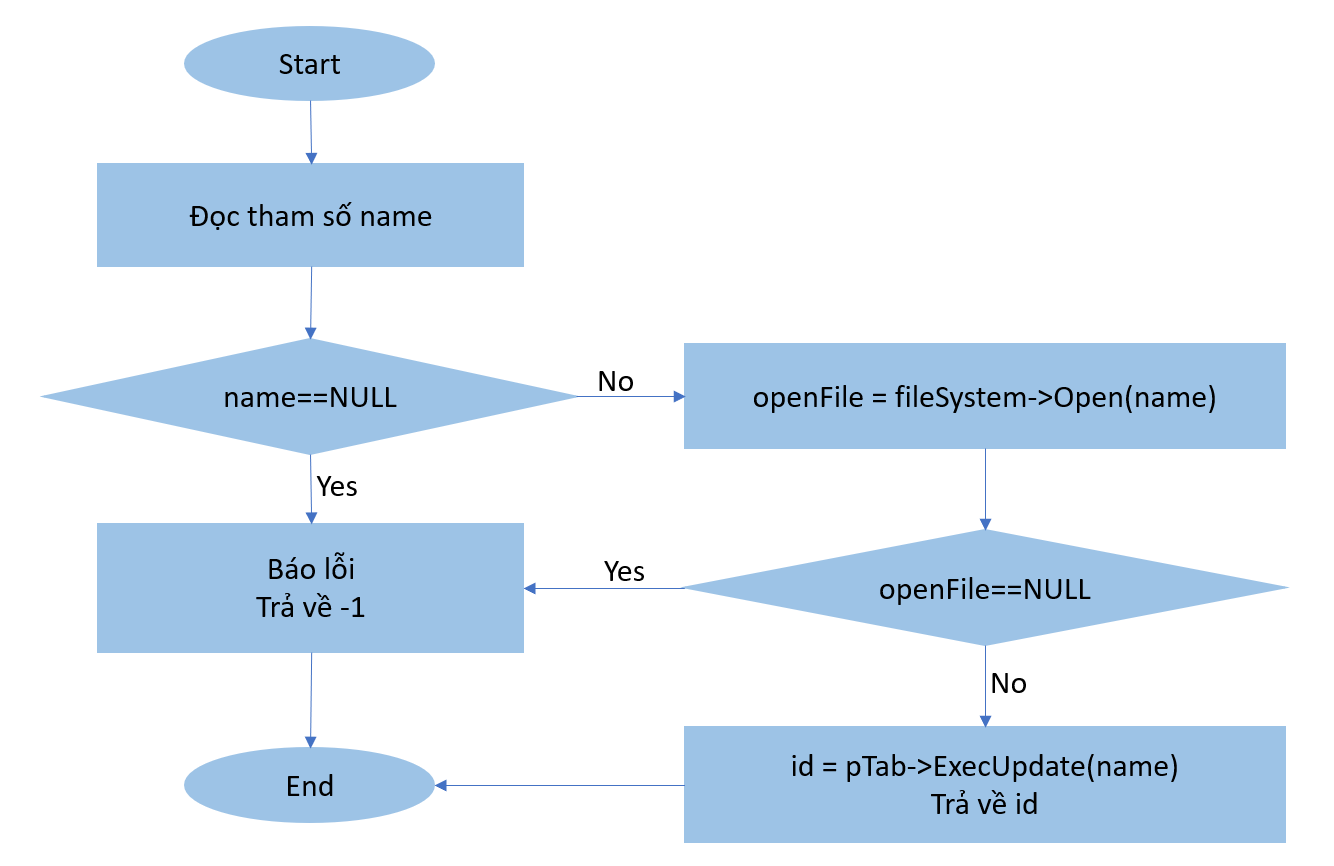
## Syscall: SpaceID Exec(char\* name)

* Mô tả: system call Exec sử dụng lớp PCB và Ptable để gọi thực thi một chương trình mới trong một system thread mới.
* Cài đặt:

+ Cài đặt hàm Exec(char\* name, int pid) ở lớp PCB.

+ Cài đặt hàm ExecUpdate(char\* name) ở lớp Ptable.

+ Syscall Exec: từ thanh ghi r4, đọc tham số name, từ userpace chuyển sang systemspace bằng User2System. Kiểm tra lỗi không mở được file, trả về thanh ghi r2 giá trị -1, ngược lại gọi pTab->ExecUpdate(name) và trả về Process SpaceID chương trình người dùng vừa được tạo.



## Syscall: Int Join(SpaceID id)

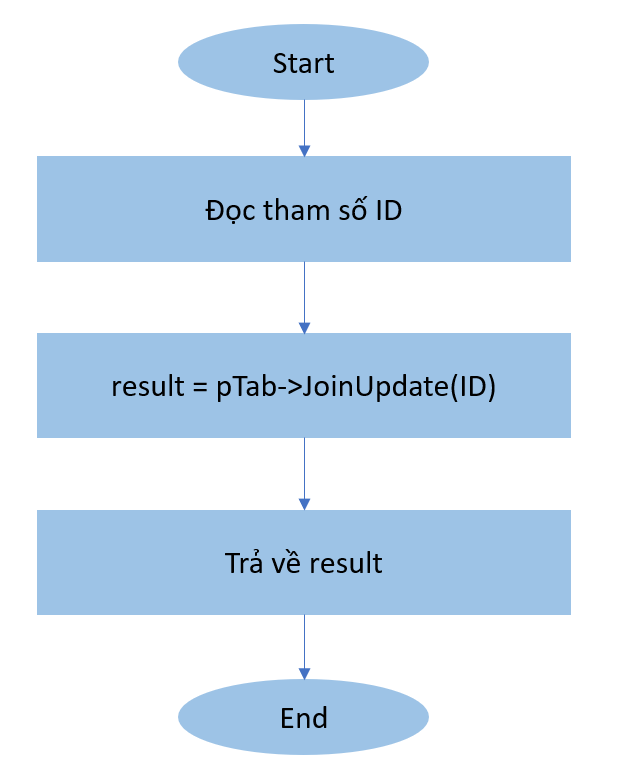
* Mô tả: system call Join sử dụng lớp PCB và Ptable để thực hiện và block dựa trên tham số Space ID id.
* Cài đặt:

+ Cài đặt hàm JoinWait() ở lớp PCB

+ Cài đặt hàm ExitRelease() ở lớp PCB

+ Cài đặt hàm JoinUpdate(int id) ở lớp Ptable

+ Syscall Join: từ thanh ghi r4, đọc tham số id. Gọi phương thức pTab->JoinUpdate(id) và lưu vào thanh ghi r2 giá trị kết quả thực hiện.



## Syscall: Void Exit(int exitCode)

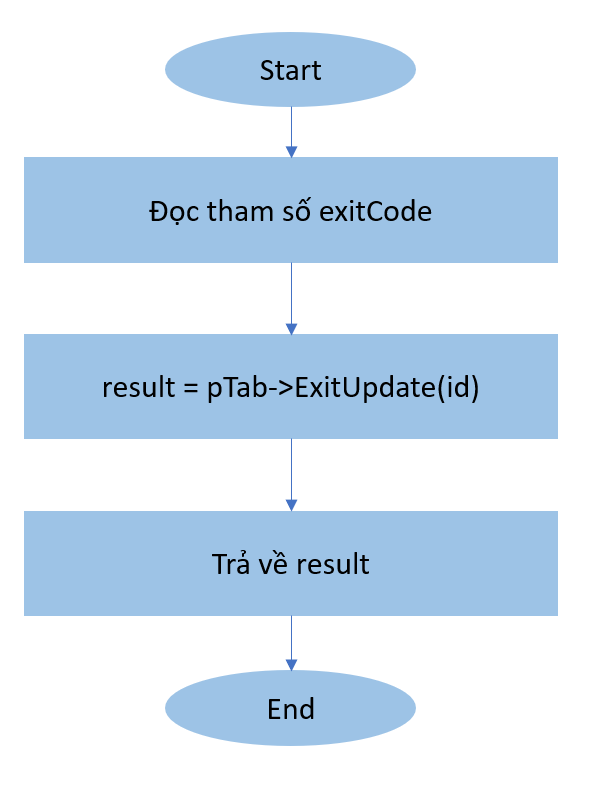
* Mô tả: system call Exit sử dụng lớp PCB và Ptable để thực hiện thoát tiến trình mà nó đã Join.
* Cài đặt:

+ Cài đặt hàm JoinRelease() ở lớp PCB

+ Cài đặt hàm ExitWait() ở lớp PCB

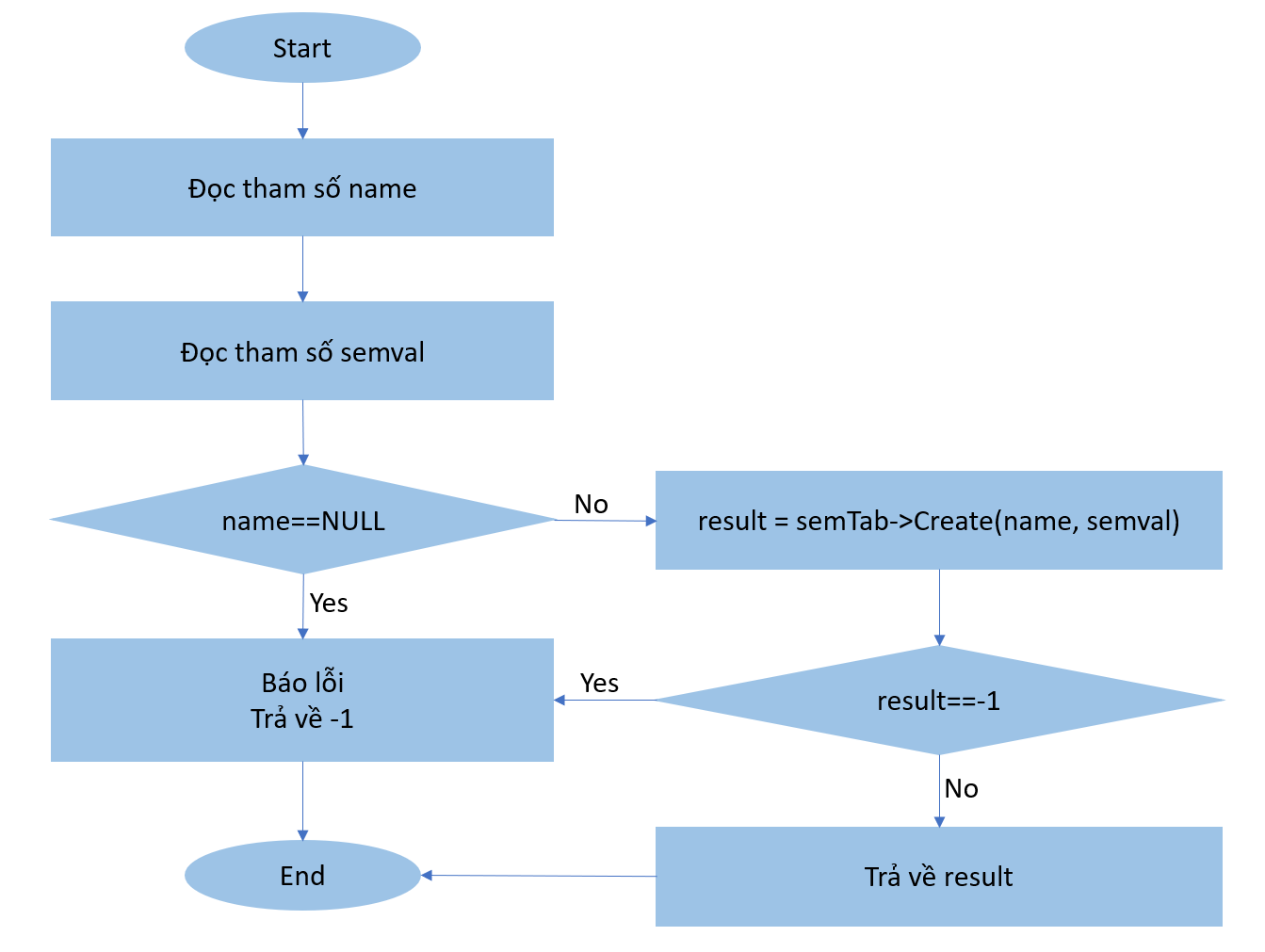
+ Cài đặt hàm ExitUpdate(int exitcode) ở lớp Ptable

+ Syscall Exit: từ thanh ghi r4, đọc tham số exitCode. Gọi phương thức pTab->ExitUpdate(exitCode) và lưu vào thanh ghi r2 giá trị kết quả thực hiện.



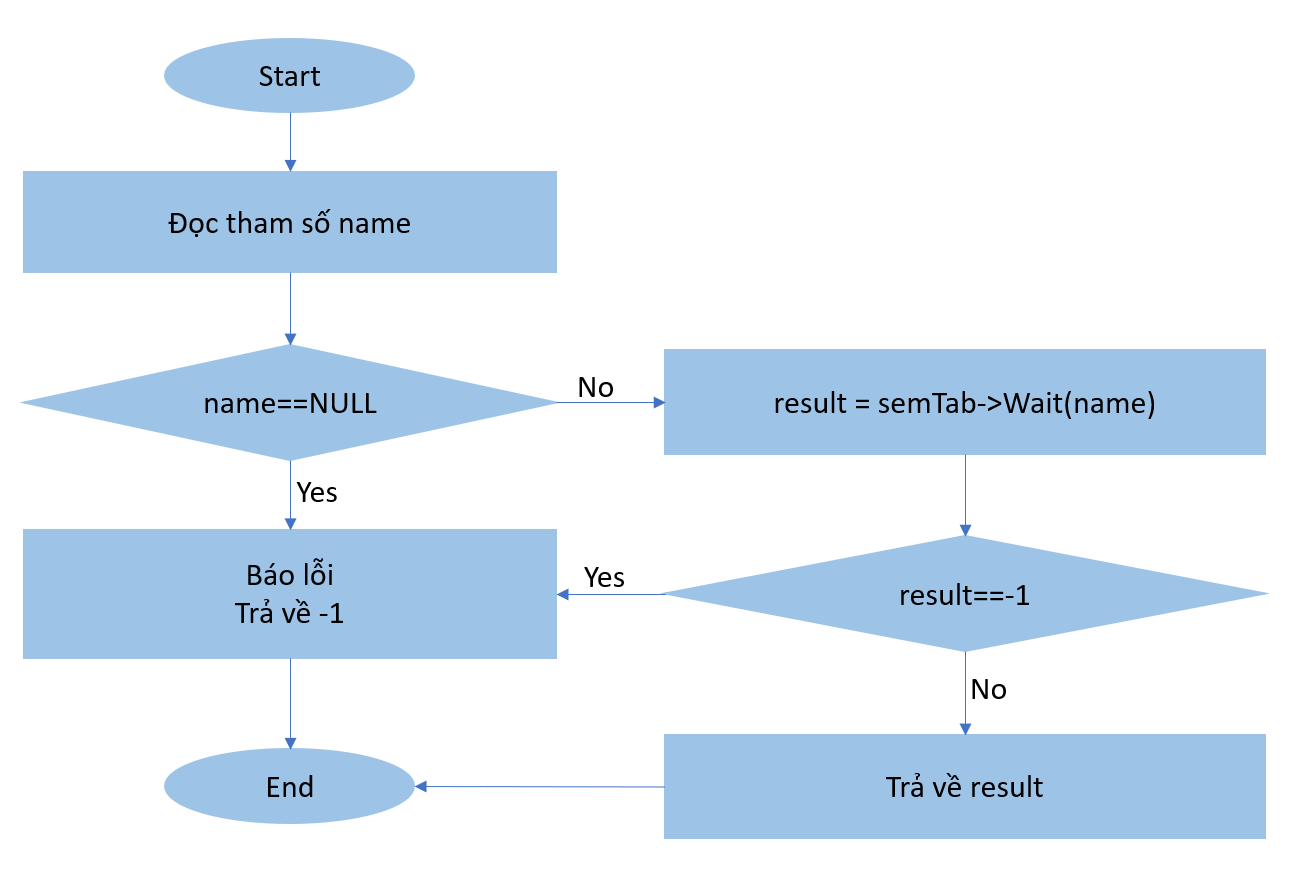
## Syscall: Int CreateSemaphore(char\* name, int semval)

* Mô tả: system call Create Semaphore để tạo một semaphore mới.
* Cài đặt: từ thanh ghi r4, r5, đọc 2 tham số name và semval, chuyển từ user space sang system space bằng hàm User2System. Thực hiện phương thức semTab->Create(name, semval) để tạo Semaphore. Nếu thành công, lưu vào thanh ghi r2 giá trị 0, ngược lại báo lỗi và trả về -1.



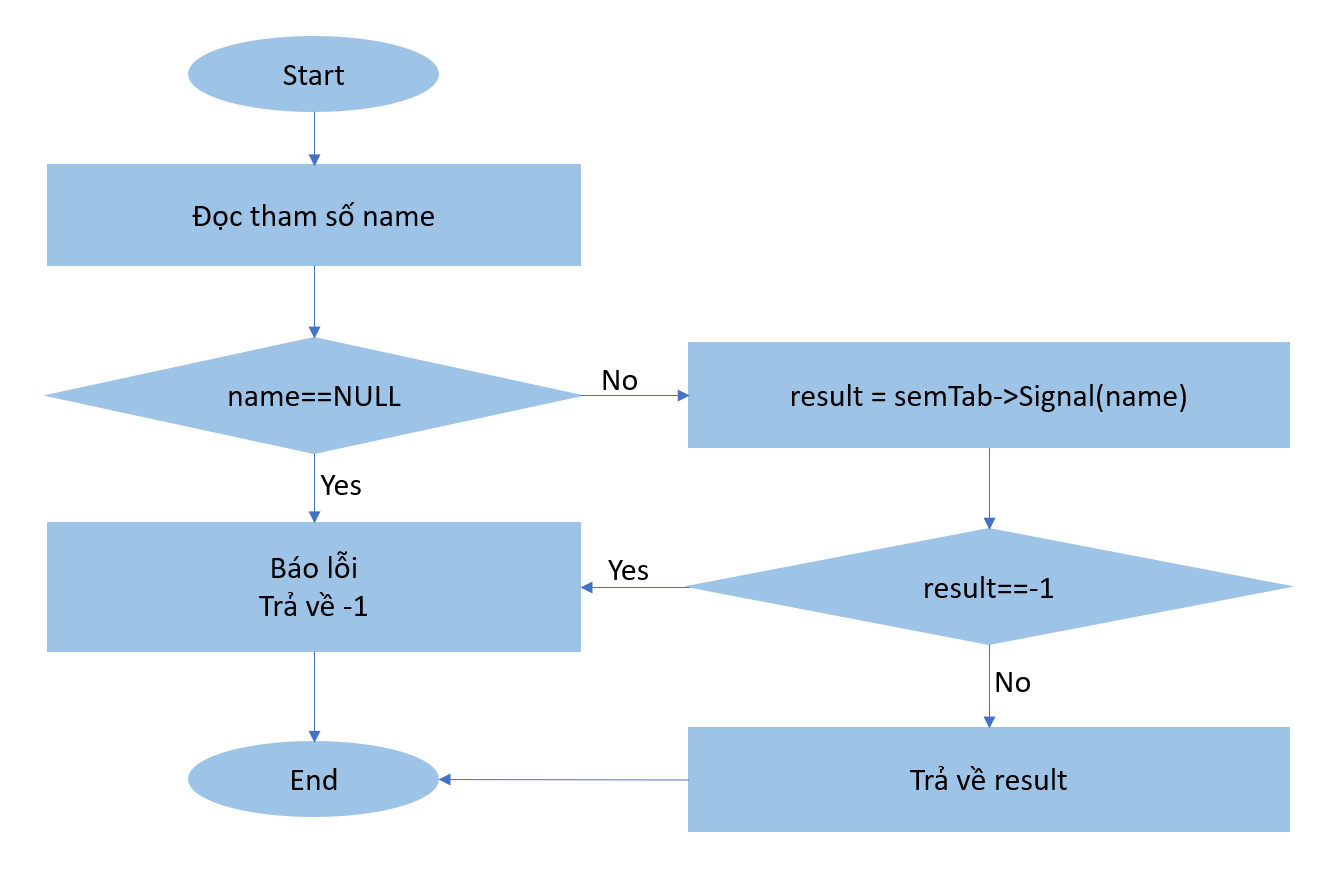
## Syscall: Int Wait(char\* name)

* Mô tả: system call Wait sử dụng lớp Stable để thực hiện thao tác chờ
* Cài đặt: từ thanh ghi r4, đọc tham số name, chuyển từ user space sang system space bằng hàm User2System. Kiểm tra semaphore “name” có chứa trong bảng Stab, sau đó gọi phương thức Wait() của lớp Stable. Thực hiện thành công, lưu vào thanh ghi r2 giá trị 0, ngược lại báo lỗi và trả về -1.



## Syscall: Int Signal(char\* name)

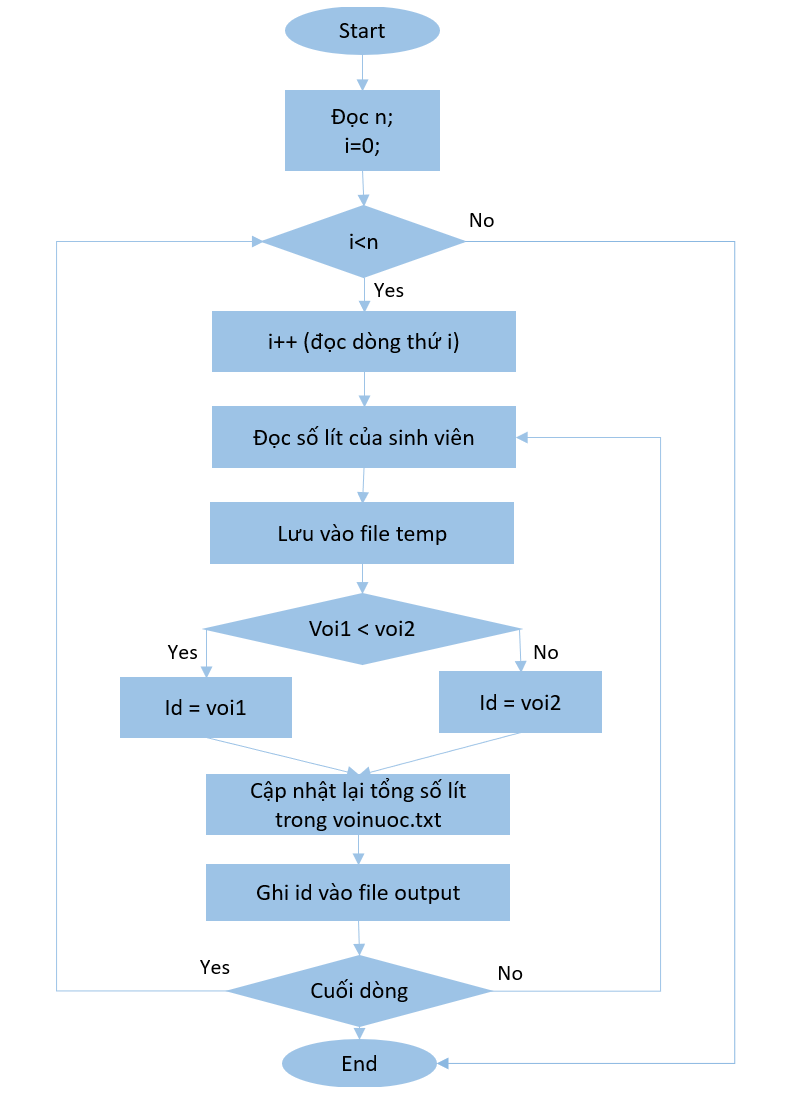
* Mô tả: system call Signal sử dụng lớp Stable để giải phóng tiến trình chờ.
* Cài đặt: từ thanh ghi r4, đọc tham số name, chuyển từ user space sang system space bằng hàm User2System. Kiểm tra semaphore “name” có chứa trong bảng Stab, sau đó gọi phương thức Signal() của lớp Stable. Thực hiện thành công, lưu vào thanh ghi r2 giá trị 0, ngược lại báo lỗi và trả về -1.



# IV – Chương trình minh họa

## Cài đặt chương trình “Thống kê sử dụng máy nóng lạnh”

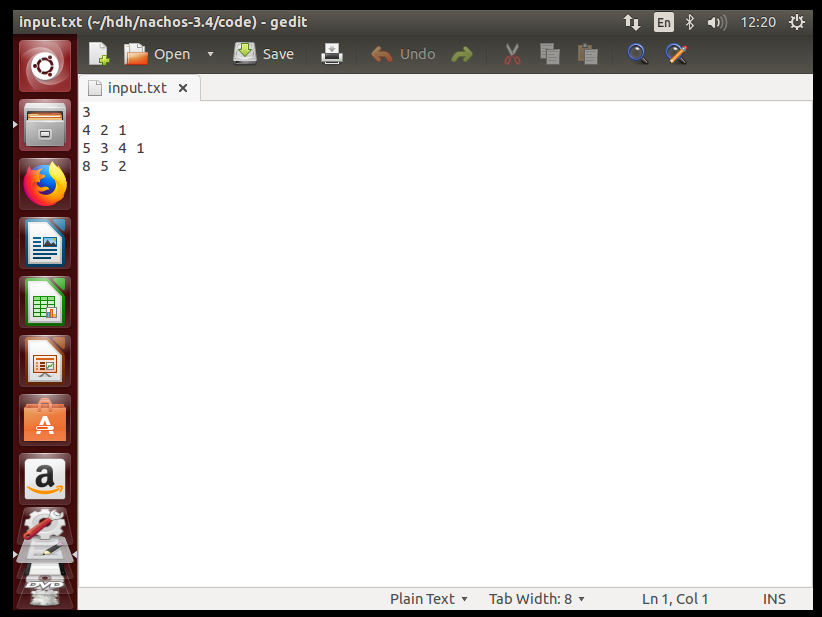
Ý tưởng: Ban đầu cả 2 vòi nước trống, sinh viên đầu tiên sẽ sử dụng vòi 1. Chúng ta sẽ cập nhật tổng số lít nước mà các sinh viên trong hàng đợi cần lấy của mỗi vòi trong 1 file tạm thời tên voinuoc.txt. Cứ mỗi một sinh viên mới đến, sẽ kiểm tra xem tổng số lít nước của vòi nào ít hơn, sinh viên sẽ được xếp vào hàng đợi của vòi đó, nếu 2 vòi bằng nhau thì xếp vào hàng của vòi 1. Sau khi xếp sinh viên mới vào, sẽ cập nhật lại tổng số lít của các vòi. Thực hiện lặp lại như vậy cho tới khi hết sinh viên.



## Hướng dẫn sử dụng chương trình

Demo chương trình:

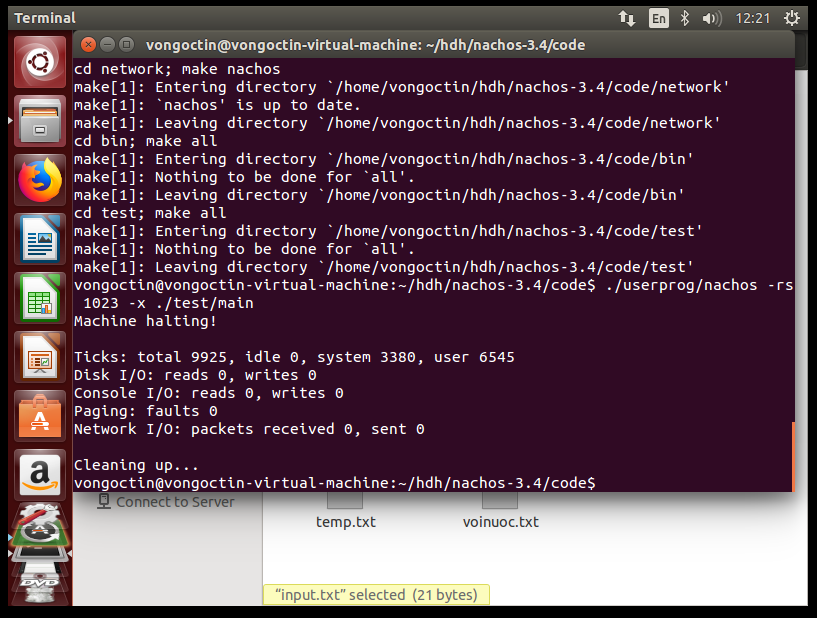
Giả sử chúng ta có file input.txt như sau: gồm 3 tiến trình, mỗi tiến trình là mỗi dòng với số lít nước của các sinh viên cần lấy nước.



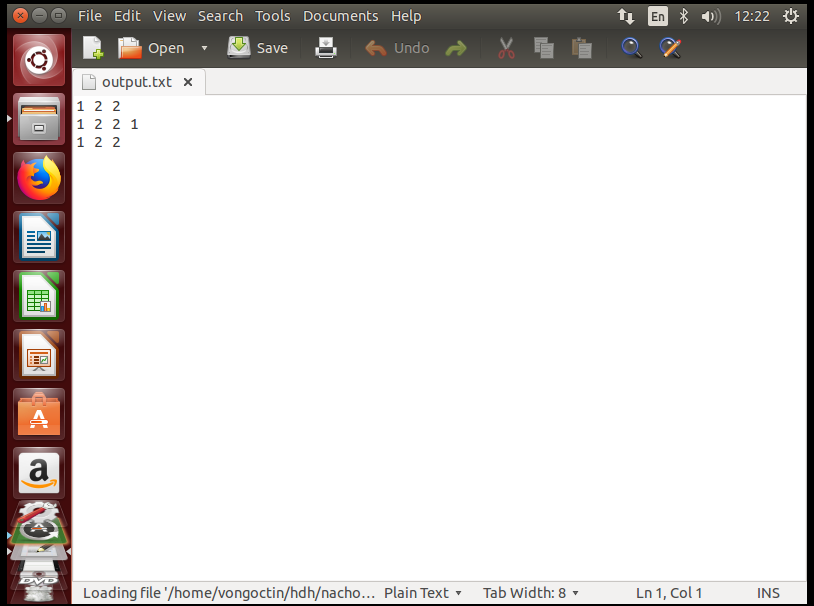
Chạy chương trình main**Text

Description automatically generated**

Chương trình chạy thành công

****

Kết quả trả về file output.txt. 3 dòng kết quả ứng với 3 tiến trình, tương ứng từng sinh viên nên sử dụng vòi nước nào.

****

# V – Tài liệu tham khảo

* Bộ tài liệu hướng dẫn đồ án:

[5] Da Chuong Dong Bo Hoa.doc

Constructor\_Cua\_AddrSpace.pdf

## Huong Dan Cac Syscall Ve Da Chuong.dpf

* Series video hướng dẫn đồ án nachos Nguyễn Thành Chung: [*https://www.youtube.com/watch?v=t0jtY1C129s&list=PLRgTVtca98hUgCN2\_2vzsAAXPiTFbvHpO*](https://www.youtube.com/watch?v=t0jtY1C129s&list=PLRgTVtca98hUgCN2_2vzsAAXPiTFbvHpO)