

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN



BÁO CÁO PROJECT 3
NACHOS - ĐA CHƯƠNG VÀ LẬP LỊCH

Giảng viên hướng dẫn: Lê Viết Long

Lớp: 21_3

Nhóm sinh viên thực hiện:

Trần Minh Triết	21120578
Lê Anh Tú	21120585
Nguyễn Minh Tuấn	21120587
Nguyễn Thị Như Ý	21120600
Trần Phước Nhân	21120515

MỤC LỤC

1. Thông tin nhóm.....	3
2. Bảng phân chia công việc	4
3. Đánh giá mức độ hoàn thành đồ án	6
4. Yêu cầu đồ án	7
4.1. Thao tác với file trong NachOS	7
4.1.1. fileSYS.h và fileSYS.cc	7
4.1.2. file system.cc.....	7
4.2. Đa chương, đa tiến trình, lập lịch và đồng bộ hóa	7
4.2.1. Mô tả thiết kế	7
4.2.2. Cấu hình các thông số cần thiết	9
4.2.3. Cài đặt các syscall.....	11
4.3. Chương trình minh họa	17
4.3.1. Chương trình in ký tự A và B	17
4.3.2. Chương trình quản lý vôi nước	18
4.3.3. Đánh giá đồ án	19
5. Tài liệu tham khảo và trích dẫn:	20

1. Thông tin nhóm

STT	Họ và tên	MSSV	Địa chỉ email
1	Trần Minh Triết	21120578	21120578@student.hcmus.edu.vn
2	Lê Anh Tú	21120585	21120575@student.hcmus.edu.vn
3	Nguyễn Minh Tuấn	21120587	21120587@student.hcmus.edu.vn
4	Nguyễn Thị Như Ý	21120600	21120600@student.hcmus.edu.vn
5	Trần Phước Nhân	21120515	21120515@student.hcmus.edu.vn

2. Bảng phân chia công việc

STT	Yêu cầu	Đảm nhiệm	Đánh giá
1	Thao tác với file trong NachOS <ul style="list-style-type: none"> - filesys.h và filesys.cc - file system.cc 	21120578	Hoàn thành
2	Mô tả thiết kế <ul style="list-style-type: none"> - Lớp PCB (./code/userprog/pcb.h) - Lớp Stable(./code/userprog/stable.h) - Lớp Ptable (./code/userprog/ptable.h) - Lớp Sem(./code/userprog/stable.h) 	21120585	Hoàn thành
3	Cấu hình các thông số cần thiết <ul style="list-style-type: none"> - Thực hiện thêm một số biến toàn cục trong file ./threads/system.h - Điều chỉnh số lượng khung trang và kích thước của sector - Cài đặt trong file ./userprog/progtest.cc - Cài đặt lại lớp addrspac 	21120587	Hoàn thành
4	Cài đặt các syscall <ul style="list-style-type: none"> - System call Exec - System call Join - System call Exit - System call CreateSemaphore - System call Up - System call Down 	21120515	Hoàn thành
5	Chương trình in ký tự A và B	21120600	Hoàn thành

6	Chương trình quản lý vôi nước	21120600	Hoàn thành
7	Viết Report	21120578 21120587 21120515 21120600 21120585	Hoàn thành

3. Đánh giá mức độ hoàn thành đồ án

STT	Yêu cầu	Mức độ Hoàn thành	Ghi chú
1	Thao tác với file trong NachOS	100%	
2	Mô tả thiết kế	100%	
3	Cấu hình các thông số cần thiết	100%	
4	Cài đặt các syscall	100%	
5	Chương trình in ký tự A và B	100%	
6	Chương trình quản lý vôi nước	100%	
7	Viết Report	100%	
	Toàn bộ đồ án	100%	

4. Yêu cầu đồ án

4.1. Thao tác với file trong NachOS

4.1.1. `filesystem.h` và `filesystem.cc`

Thực hiện thêm các xử lý cho class `FileSystem` của khối lệnh sau `#else (FILESYS)` để cho phép mở file với 2 tác vụ là chỉ đọc (tương ứng với `type = 1`) và vừa đọc vừa ghi (tương ứng với `type = 0`).

Trong file `filesystem.h` tạo thêm biến `index` và biến `openState` để lưu tình trạng mở file (cho phép tối đa 10 file được mở). Ngoài ra còn khai báo phương thức `FindFreeSpace()` để tìm vị trí trống cho file mới được tạo.

Override phương thức `Open(char* name, int type)` với 1 tham số truyền vào là `name` tương ứng với tên file cần mở, và `type` tương ứng với loại file như đã đề cập bên trên.

Trong file `filesystem.cc` tiến hành viết các thao tác cho các phương thức đã khai báo:

- Đối với phương thức khởi tạo cần phải khởi tạo cho biến `openState` với 10 giá trị ban đầu bằng `NULL`.

- Ngoài ra sẽ dành ra 2 vị trí đầu tiên để đặt cho console input và console output và gán `type` cho chúng lần lượt là 2 (`stdin`) và 3 (`stdout`) => có cả phương thức `Create` và `Open`.

4.1.2. `file system.cc`

Khởi tạo các giá trị format cho của `FILESYS_NEEDED = TRUE` để khi biên dịch thì class `FileSystem` của `FILESYS` sẽ được sử dụng. Như vậy mới có thể thực hiện thao tác với 2 loại file là chỉ đọc(1) và đọc kết hợp với ghi(0).

4.2. Đa chương, đa tiến trình, lập lịch và đồng bộ hóa

4.2.1. Mô tả thiết kế

4.2.1.1. Lớp `PCB(./code/userprog/pcb.h)`

- Có chức năng là lưu thông tin quản lý các process, mô tả một tiến trình và quản lý hành động của tiến trình đó.

- Một số thuộc tính quan trọng:

- + Semaphore `*joinsem, *exitsem, *multex` lần lượt là 3 semaphore quản lý cho các quá trình `Join`, `Exit` và truy xuất độc quyền.

- + Int `numwait` để cho biết được số lượng các tiến trình đã `join`.

- + Char `filename[32]` cho biết tên của tiến trình.

- + Thread *thread cho biến tiến trình của chương trình.
- + Int parentID: cho biến id của tiến trình cha của tiến trình đang chạy.
- Một số phương thức quan trọng:
 - + Int Exec(char* filename, int pid): có chức năng gọi thực thi một thread với tên truyền vào filename và id là pid.
 - + Void JoinWait(): thực hiện Join một.

4.2.1.2. Lớp Stable(./code/userprog/stable.h)

- Kết hợp với lớp sem để thực hiện công việc quản lý semaphore. Lớp Stable sẽ quản lý toàn bộ các semaphore hiện đang có trong hệ điều hành.
- Một số thuộc tính quan trọng:
 - + BitMap *bm để quản lý các slot còn trống.
 - + MAX_SEMAPHORE = 10 cho biết số tiến trình tối đa được nạp vào.
 - + Sem *semtabl[MAX_SEMAPHORE] quản lý các đối tượng semaphore.
- Một số phương thức quan trọng:
 - + Int Create(char* name, inti nit) : tạo một đối tượng semaphore mới với tên và giá trị được truyền vào ban đầu.
 - + Int Wait(char* name): thực hiện dùng semaphore và đợi.
 - + Int Signal(char* name) cho phép semaphore đang ở trạng thái đợi được phép thực hiện tiếp.

4.2.1.3. Lớp Ptable (./code/userprog/ptable.h)

- Có chức năng quản lý toàn bộ tiến trình trong hệ điều hành.
- Có các thuộc tính quan trọng:
 - + Hằng số MAX_PROCESS = 10 cho biết số tiến trình tối đa mà PTable quản lý.
 - + BitMap *bm để đánh dấu cho các vị trí đã được sử dụng trong PCB.
 - + Semaphore *bmsem: quản lý việc nạp tiến trình, giúp ngăn chặn 2 tiến trình được nạp vào cùng một lúc.
- Một số phương thức quan trọng:

- + Int ExecUpdate(char* name): phương thức này tạo ra một thread mới và được dùng trong khi xử lý syscall SC_Exit.
- + Int ExitUpdate(int ec): hàm này có chức năng thoát thread hiện tại với tham số exitcode được truyền vào giúp xử lý cho syscall SC_Exit.
- + Int JoinUpdate(int id) : hàm này để join vào thread con có id được truyền vào từ thread hiện tại và chờ cho thread con chạy xong. Được dùng trong syscall SC_Join.
- + Int GetFreeSlot(): hàm này để tìm vị trí còn trống để lưu thông tin cho một tiến trình mới.

4.2.1.4. Lớp Sem(./code/userprog/stable.h)

- Có vai trò quản lý các semaphore.
- Có các thuộc tính quan trọng:
 - + Char name[50] cho biết tên của semaphore
 - + Semaphore *sem cho biết đối tượng semaphore đang được quản lý
- Một số phương thức quan trọng:
 - + void Wait() giảm giá trị của semaphore. Trường hợp mà giá trị semaphore đó giảm tới 0 thì không thể giảm được nữa, lúc đó thread hiện tại sẽ rơi vào trạng thái chờ
 - + void Signal() Tăng giá trị của semaphore. Nếu thread đó đang ở trạng thái chờ thì sẽ được gọi để thực hiện tiếp

4.2.2. Cấu hình các thông số cần thiết

4.2.2.1. Thực hiện thêm một số biến toàn cục trong file ./threads/system.h

```
#include "bitmap.h"
extern BitMap* gPhysPageBitMap;

#include "ptable.h"
extern PTable *pTab;

#include "stable.h"
extern STable *semTab;
extern Semaphore *addrLock;
```

```
addrLock = new Semaphore("addrLock", 1);
gPhysPageBitMap = new BitMap(256);
pTab = new PTable(10);
semTab = new STable();
```

Trong đó:

- Biến BitMap *gPhysPageBitMap để quản lý cho các frame.
- Biến Semaphore *addrLock.
- Biến Stable *semTab để quản lý cho các semaphore.
- Biến PTable *pTab để quản lý cho bảng các tiến trình.

4.2.2.2. Điều chỉnh số lượng khung trang và kích thước của sector

- Trong file ./machine/machine.h thực hiện thay đổi giá trị số lượng khung trang.

```
#define NumPhysPages    128
```

- Trong file ./machine/disk.h thực hiện thay đổi kích thước của sector.

```
#define SectorSize      512 // number of bytes per disk sector
```

4.2.2.3. Cài đặt trong file ./userprog/progtest.cc

- Tạo ra một phương thức StartProcessWithId để hàm Folk có trở hàm này tới vùng nhớ của tiến trình con.

```
void StartProcessWithID(int id)
{
    char* fileName = pTab->GetFileName(id);

    AddrSpace *space;
    space = new AddrSpace(fileName);

    if(space == NULL)
    {
        printf("\nPCB::Exec : Can't create AddrSpace.");
        return;
    }

    currentThread->space = space;

    space->InitRegisters();
    space->RestoreState();

    machine->Run();
    ASSERT(FALSE);
}
```

4.2.2.4. Cài đặt lại lớp addrspace

Trong phương thức khởi tạo của lớp addrspace cần phải giải quyết được vấn đề cấp phát các frame cho bộ nhớ vật lý sao cho có thể nạp được nhiều chương trình cùng lúc vào đó thông qua biến toàn cục đã được khai báo là gPhysPageBitMap.

Trước khi nạp tiến trình sẽ cần phải kiểm tra số lượng trang trống bằng phương thức NumClear() trong lớp BitMap. Sau đó thực hiện tạo pageTable với phương thức new TranslationEntry. Tiếp đến sẽ đi tìm kiếm các trang còn trống thông qua phương thức Find() sau đó nạp chương trình đó lên bộ nhớ chính.

Cần phải cài đặt thêm phương thức phá hủy để dọn dẹp, cập nhật lại bộ nhớ khi tiến trình kết thúc.

4.2.3. Cài đặt các syscall

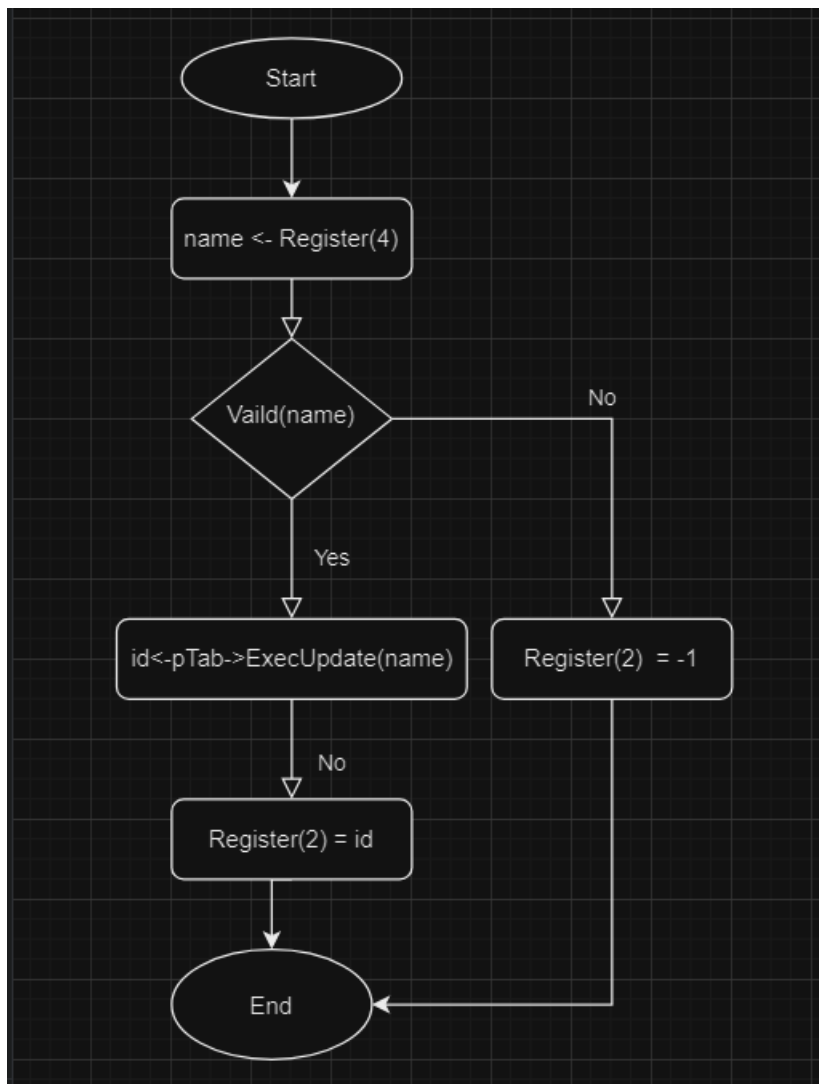
4.2.3.1. System call Exec

- Dùng để gọi thực thi cho một chương trình mới trong system thread mới.
- Trước khi thực hiện định nghĩa cho syscall Exec cần:
 - + Khai báo cho syscall trong file ./userprog/syscall.h

```
SpaceId Exec(char *name);
```

- + Cài đặt phương thức Exec(char* name, int pid) ở lớp PCB
- + Cài đặt phương thức ExecUpdate(char* name) ở lớp PTable

- Lưu đồ cho quá trình xử lý system call Exec.



4.2.3.2. System call Join

- Có chức năng thực hiện đợi và block dựa trên tham số SpaceID

- Trước khi cài đặt system call Join cần phải

+ Khai báo cho syscall trong file ./userprog/syscall.h

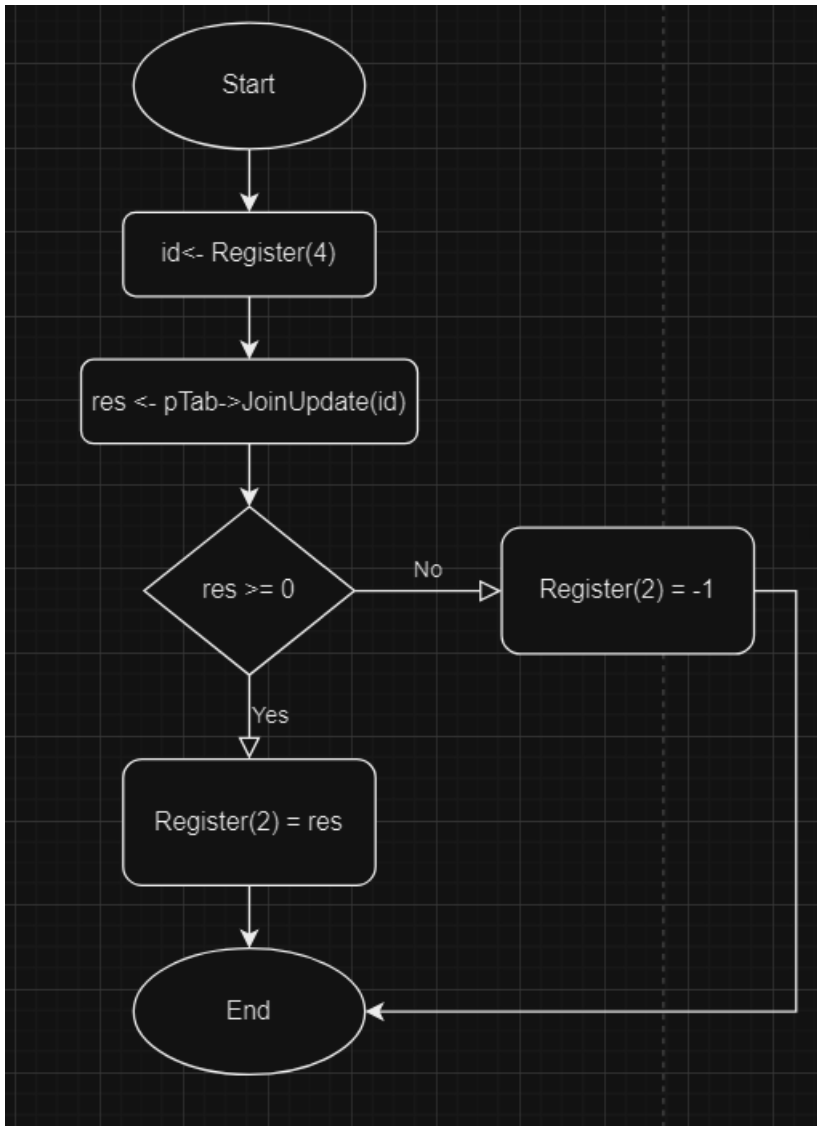
```
int Join(SpaceId id);
```

+ Cài đặt phương thức JoinWait() ở lớp PCB

+ Cài đặt phương thức ExitRelease() ở lớp PCB

+ Cài đặt phương thức JoinUpdate(int id) ở lớp Ptable

- Lưu đồ xử lý system call Join.



4.2.3.3. System call Exit

- Có chức năng thực hiện thoát cho tiến trình đã được Join

- Trước khi cài đặt system call Exit cần phải

+ Khai báo system call này trong file ./userprog/syscall.h

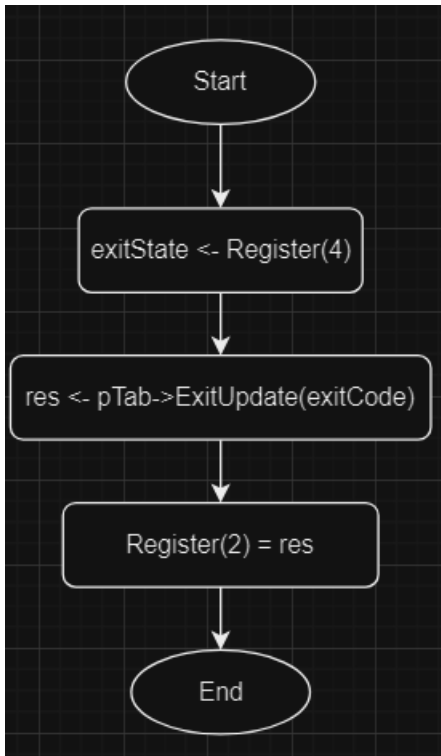
```
void Exit(int status);
```

+ Cài đặt phương thức JoinRelease() ở lớp PCB

+ Cài đặt phương thức ExitWait() ở lớp PCB

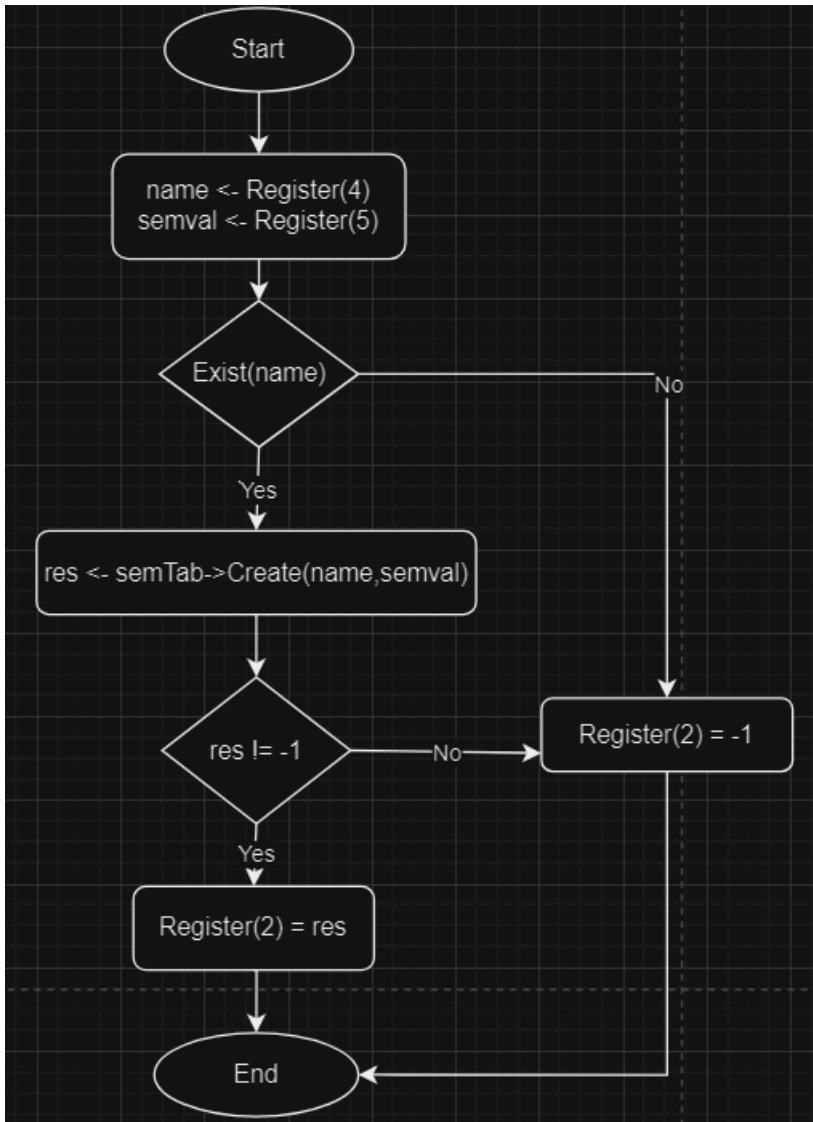
+ Cài đặt phương thức ExitUpdate(int exitcode) ở lớp PTable

- Lưu đồ cài đặt syscall Exit



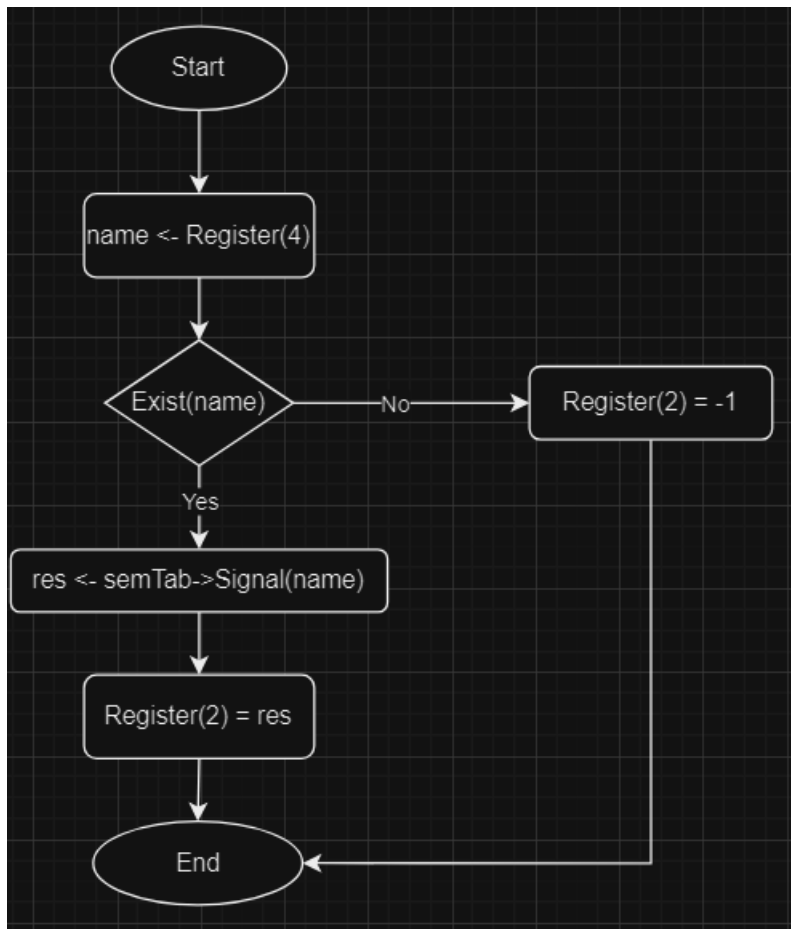
4.2.3.4. System call CreateSemaphore

- Có chức năng tạo ra một semaphore mới
- Lưu đồ cài đặt syscall CreateSemaphore



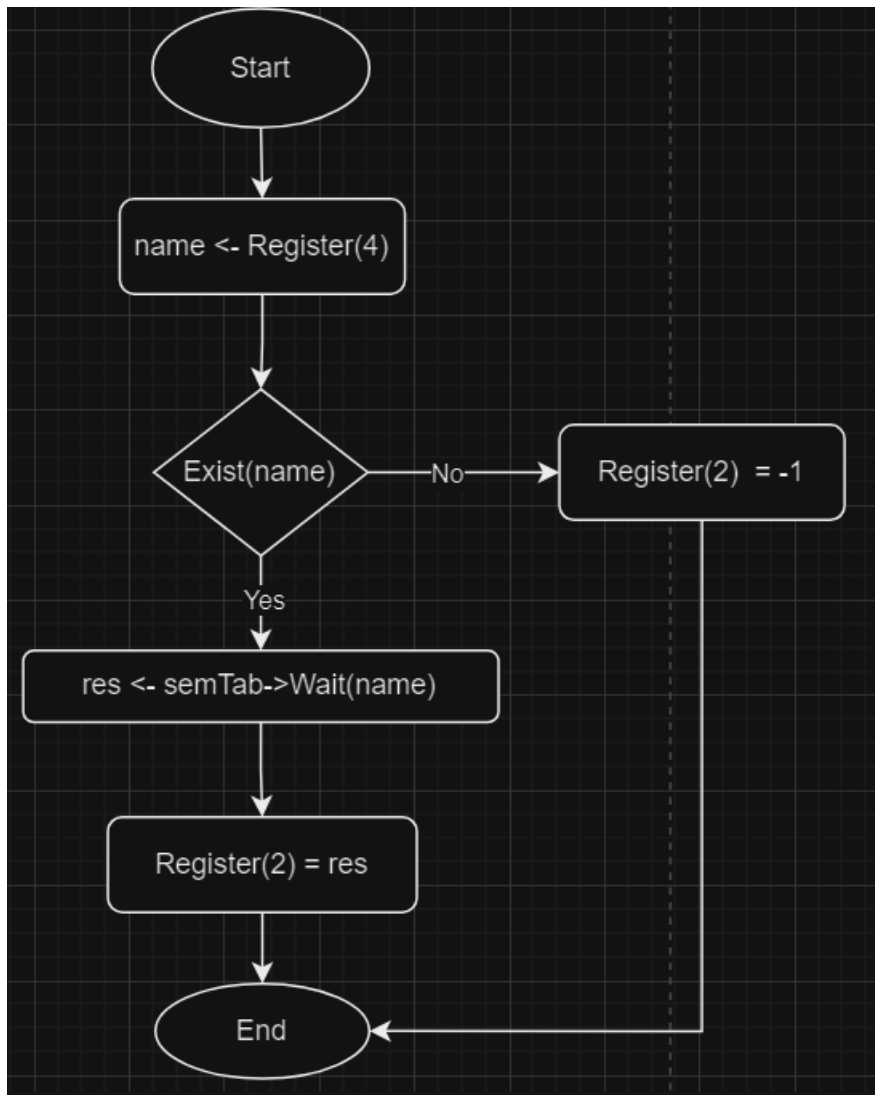
4.2.3.5. System call Up

- Dừng để giải phóng cho tiến trình đang chờ.
- Lưu đồ cài đặt System call Up.



4.2.3.6. System call Down

- Dùng để thực hiện thao tác chờ
- Lưu đồ cho quá trình cài đặt System call Down



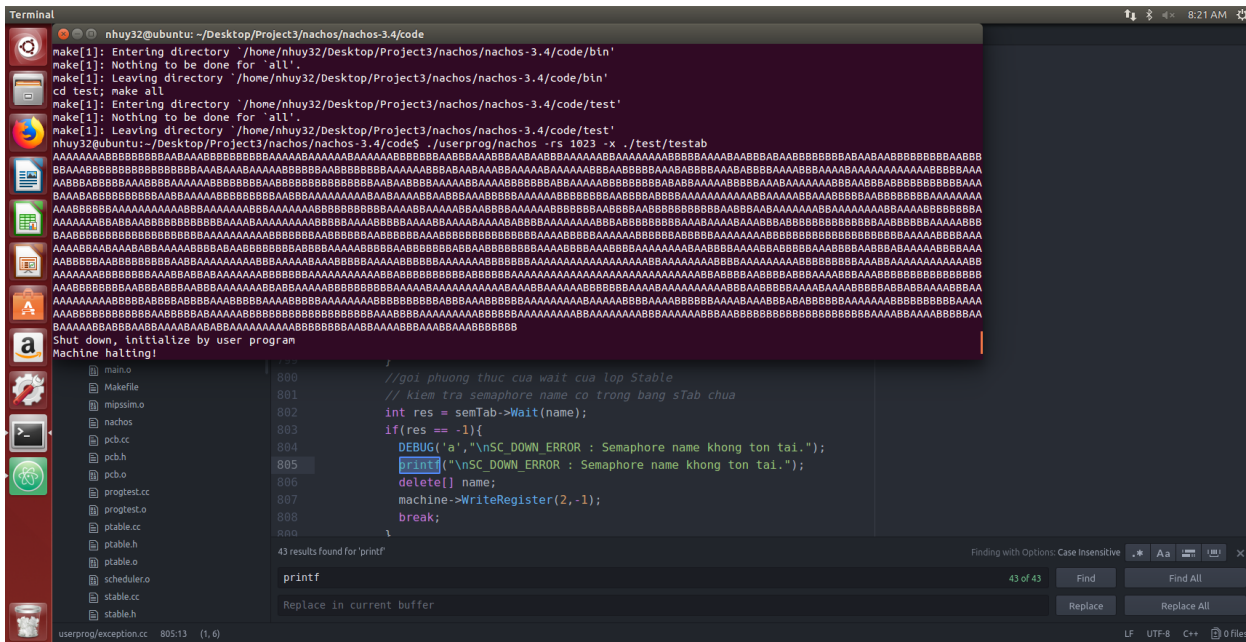
4.3. Chương trình minh họa

4.3.1. Chương trình in ký tự A và B

- Mô tả: Chương trình sẽ có chức năng in ra 1000 ký tự 'A' và 1000 ký tự 'B' với các ký tự 'A', 'B' xuất hiện xen kẽ một cách ngẫu nhiên.
- Để thực hiện chạy thử chương trình gọi lệnh sau.

```
./userprog/nachos -rs 1023 -x ./test/testtab
```

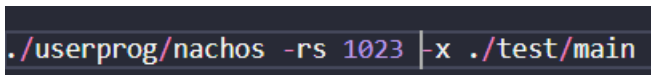
- Kết quả sau khi thực hiện chạy chương trình



4.3.2. Chương trình quản lý vòi nước

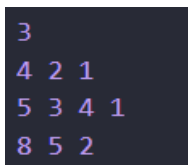
- Mô tả: lấy input đầu vào là dung tích bình nước của một số sinh viên trong cùng một thời điểm nhất định. Chương trình sẽ tính toán và đưa ra kết quả xem sinh viên với bình nước có dung tích nào sẽ rót nước ở vòi nào

- Để thực hiện chạy chương trình ta gọi lệnh sau

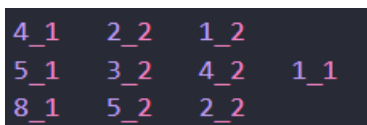


- Kết quả thực hiện

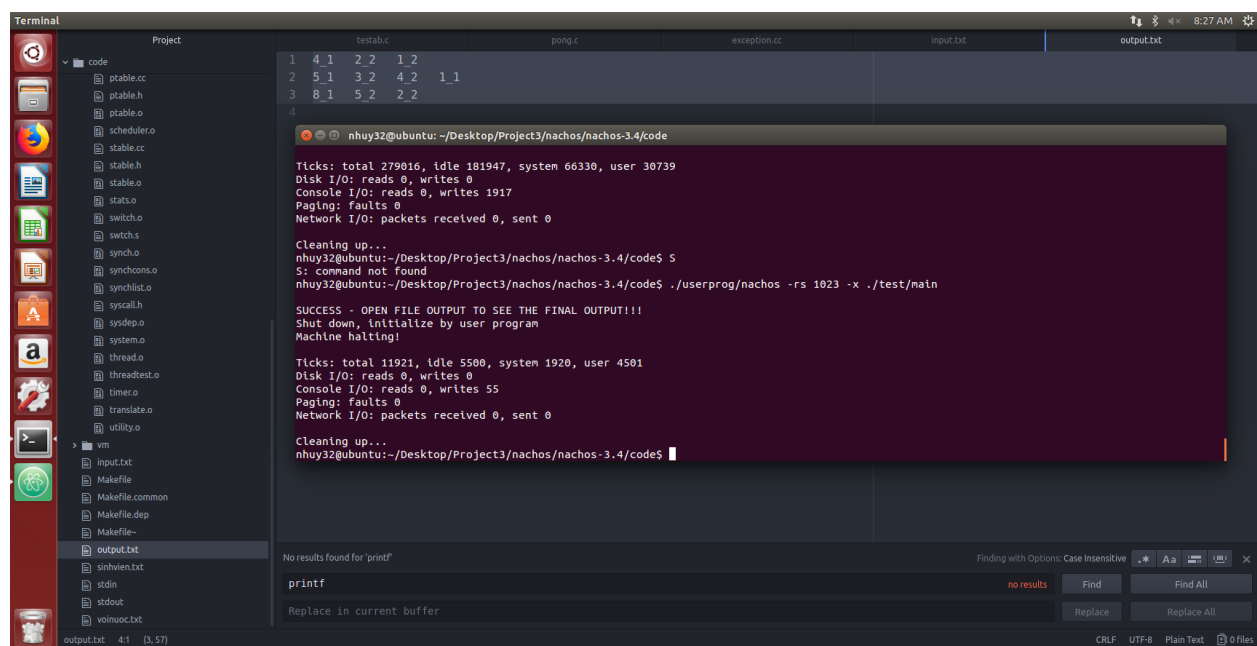
+ Với file input như sau:



+ Ta sẽ có file output như sau:



+ Và kết quả hiển thị trên màn hình console:



```
1 4 1 2 2 1 2
2 5 1 3 2 4 2 1 1
3 8 1 5 2 2 2
4

nhuy32@ubuntu: ~/Desktop/Project3/nachos/nachos-3.4/code

Ticks: total 279016, idle 181947, system 66330, user 30739
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 1917
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
nhuy32@ubuntu:~/Desktop/Project3/nachos/nachos-3.4/code$ S
S: command not found
nhuy32@ubuntu:~/Desktop/Project3/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/main

SUCCESS - OPEN FILE OUTPUT TO SEE THE FINAL OUTPUT!!!
Shut down, initialize by user program
Machine halting!

Ticks: total 11921, idle 5500, system 1920, user 4501
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 55
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
nhuy32@ubuntu:~/Desktop/Project3/nachos/nachos-3.4/code$
```

4.3.3. Đánh giá đồ án

Mọi thứ đã được hoàn thành theo yêu cầu của đề bài.

Các phần chưa hoàn thành: không.

5. Tài liệu tham khảo và trích dẫn:

[1] ThS. Lê Viết Long, (2023). Slide và tài liệu lập trình Nachos

[2] Nguyễn Thành Chung, (30-November-2023), Lập trình NaChos HCMUS,
<https://www.youtube.com/@thanhchungnguyen2618/playlists>

[3] Trường CNTT & TT, (2023), Hệ điều hành, <https://tailieuhust.com/tai-lieu-mon-he-dieu-hanh-hust/>

[4] Sukarna Barua, (2023), tanviramin, <http://www.tanviramin.com/documents/nachos1.pdf>