

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN



Project 03:

Đa chương và đồng bộ hoá

Môn: Hệ điều hành

Giảng viên hướng dẫn: ThS. Lê Việt Long

Thành viên nhóm:

21120279 – Lê Trần Minh Khuê

21120289 – Diệp Quốc Hoàng Nam

21120290 – Hoàng Trung Nam

21120348 – Nguyễn Trần Trình

21120354 – Lương Thanh Tú

Thành phố Hồ Chí Minh, tháng 1 / 2023

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành đến Khoa Công nghệ Thông tin, Trường Đại học Khoa học Tự nhiên Thành phố Hồ Chí Minh đã tạo điều kiện thuận lợi cho chúng em học tập và hoàn thành bài đồ án này. Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc đến ThS. Lê Viết Long, người thầy đã dày công truyền đạt kiến thức và hướng dẫn chúng em trong quá trình làm bài.

Chúng em đã cố gắng vận dụng những kiến thức đã học được để hoàn thành bài đồ án. Nhưng do kiến thức hạn chế và không có nhiều kinh nghiệm thực tiễn nên khó tránh khỏi những thiếu sót trong quá trình nghiên cứu và trình bày. Rất kính mong sự góp ý của quý Thầy để bài báo cáo của chúng em được hoàn thiện hơn.

Một lần nữa, chúng em xin trân trọng cảm ơn sự quan tâm giúp đỡ của thầy trong suốt quá trình thực hiện đồ án này.

Xin trân trọng cảm ơn!

MỤC LỤC

LỜI CẢM ƠN.....	2
MỤC LỤC	3
DANH MỤC BẢNG	4
1. TỔNG QUAN	5
1.1. THÔNG TIN NHÓM.....	5
1.2. BẢNG PHÂN CÔNG CÔNG VIỆC	5
1.3. CÁC NỘI DUNG ĐÃ HOÀN THÀNH	5
1.4. TỔ CHỨC BÀI LÀM VÀ CÁC GHI CHÚ LẬP TRÌNH.....	6
1.4.1. Tổ chức bài làm.....	6
1.4.2. Môi trường thực nghiệm:	6
2. TỔNG QUAN CÀI ĐẶT	6
2.1. Các cài đặt sẵn có:.....	6
2.2. Các cài đặt của project trước:.....	7
2.3. Các giá trị thanh ghi:	7
3. NỘI DUNG BÁO CÁO	8
3.1. System call nhập xuất file	8
3.1.1. Cài đặt syscall CreateFile.....	8
3.1.2. Cài đặt syscall Open và Close:.....	9
3.1.3. Cài đặt syscall Read và Write:	10
3.2. Đa chương, lập lịch và đồng bộ hoá trong NachOS	12
3.2.1. Chuyển hệ thống từ đơn chương thành đa chương:	12
3.2.2. Cài đặt system call Exec	15
3.2.3. Cài đặt system call Join và Exit	15

3.2.4. Cài đặt system call CreateSemaphore	17
3.2.5. Cài đặt system call Wait và Signal:	17
3.2.6. Cài đặt system call Seek.....	19
3.3. Chương trình minh hoạ	19
3.3.1. Demo chương trình:	20
3.3.2. Hướng dẫn sử dụng chương trình:	21
4. TÀI LIỆU THAM KHẢO.....	21

DANH MỤC BẢNG

Bảng 1. Danh sách thành viên	5
Bảng 2. Bảng phân công công việc	5
Bảng 3. Mức độ hoàn thành.....	5

1. TỔNG QUAN

1.1. THÔNG TIN NHÓM

STT	MSSV	Họ và Tên	Email
1	21120279	Lê Trần Minh Khuê	21120279@student.hcmus.edu.vn
2	21120289	Diệp Quốc Hoàng Nam	21120289@student.hcmus.edu.vn
3	21120290	Hoàng Trung Nam	21120290@student.hcmus.edu.vn
4	21120348	Lương Thanh Tú	21120348@student.hcmus.edu.vn
5	21120354	Nguyễn Trần Trình	21120354@student.hcmus.edu.vn

Bảng 1. Danh sách thành viên

1.2. BẢNG PHÂN CÔNG CÔNG VIỆC

STT	Mô tả công việc	Người thực hiện
1	Code phần syscall nhập xuất file	Trung Nam, Hoàng Nam
2	Code phần đa chương, lập lịch và đồng bộ hoá	Trần Trình, Thanh Tú
3	Code phần chương trình vôi nước	Cả nhóm
4	Viết hoàn thiện báo cáo	Minh Khuê

Bảng 2. Bảng phân công công việc

1.3. CÁC NỘI DUNG ĐÃ HOÀN THÀNH

Yêu cầu	Công việc	Tỉ lệ hoàn thành
1	Code phần syscall nhập xuất file	100%
2	Code phần đa chương, lập lịch và đồng bộ hoá	100%
3	Code phần chương trình vôi nước	100%
4	Viết hoàn thiện báo cáo	95%

Bảng 3. Mức độ hoàn thành

1.4. TỔ CHỨC BÀI LÀM VÀ CÁC GHI CHÚ LẬP TRÌNH

1.4.1. Tổ chức bài làm

Chúng em bố trí bài làm thành các file như sau:

- **Report:** Chứa file báo cáo về bài làm của nhóm (Report.pdf – file này).
- **Source:** chứa source code toàn bộ chương trình (có ghi chú các phần đã chỉnh sửa). Gồm folder nachos được nén lại sau khi nhóm đã thực hiện cài đặt xong các yêu cầu đồ án.

1.4.2. Môi trường thực nghiệm:

Chương trình được cài đặt và chạy thử trên máy ảo sử dụng:

- Hệ điều hành: Linux – Ubuntu 14.04
- Trình biên dịch:
 - *gcc (Ubuntu 4.8.4-2ubuntu1~14.04.4) 4.8.4*
 - *g++ (Ubuntu 4.8.4-2ubuntu1~14.04.4) 4.8.4*

2. TỔNG QUAN CÀI ĐẶT

2.1. Các cài đặt sẵn có:

- **progtest.cc:** kiểm tra các thủ tục để chạy chương trình người dùng
- **syscall.h:** Định nghĩa giao diện thủ tục hệ thống (system call interface) mà các chương trình người dùng có thể gọi để tương tác với kernel.
- **exception.cc:** Xử lý system call và các exception khác ở mức người dùng
- **bitmap.*:** các hàm xử lý cho lớp bitmap (thường dùng để theo dõi trạng thái của các bit bằng cách đánh dấu việc đang được sử dụng hay không của các tài nguyên như các ô nhớ vật lý, các block trên đĩa,...)
- **filesys.h:** định nghĩa các hàm trong hệ thống tệp NachOS. Thường chứa các hàm quản lý như mở, đóng, đọc, ghi, di chuyển con trỏ,...
- **openfile.h:** định nghĩa các hàm trong hệ thống file đang mở trong NachOS. Thường tập trung vào các file đang được mở.
- **translate.*:** Dùng để quản lý chuyển địa chỉ ảo sang địa chỉ vật lý và ngược lại.

- **machine.***: mô phỏng các thành phần của máy tính khi thực thi chương trình người dùng (Bộ nhớ chính, thanh ghi,...)
- **mipssim.cc**: mô phỏng tập lệnh của MIPS R2/3000 processor.
- **console.***: chứa mã nguồn mô phỏng một thiết bị đầu cuối sử dụng các tệp tin của UNIX.

Đặc điểm chung của thiết bị:

- Đơn vị dữ liệu theo byte: các hoạt động đọc và ghi sẽ thực hiện trên cấp độ byte.
- Đọc và ghi các bytes cùng một thời điểm: có khả năng đọc và ghi nhiều bytes cùng lúc, tối ưu hiệu suất.
- Các bytes đến bất đồng bộ: khi chương trình yêu cầu đọc dữ liệu từ thiết bị, nó không bị chặn và có thể tiếp tục thực hiện các công việc khác trong khi đợi dữ liệu đến. Khi dữ liệu sẵn sàng, nó có thể được đọc mà không làm chậm chương trình chính.
- **synchconsole.***: Nhóm hàm cho việc quản lý nhập xuất I/O theo dòng trong NachOS.
- **../test/***: các chương trình C được biên dịch theo MIPS và chạy trong NachOS.

2.2. Các cài đặt của project trước:

- **User2System**: Sao chép dữ liệu từ người dùng, chuyển cho hệ thống.
- **System2User**: Sao chép dữ liệu từ hệ thống, chuyển cho người dùng.
- **readInt & printInt**: hàm đọc và hàm in số nguyên
- **readChar & printChar**: hàm đọc và hàm in một ký tự
- **readString & printString**: Hàm đọc và hàm in một chuỗi

2.3. Các giá trị thanh ghi:

- **Register 2**: Lưu mã syscall và lưu kết quả trả về của mỗi syscall (nếu có).
- **Register 4**: Lưu tham số thứ nhất
- **Register 5**: Lưu tham số thứ hai
- **Register 6**: Lưu tham số thứ ba
- **Register 7**: Lưu tham số thứ tư

3. NỘI DUNG BÁO CÁO

3.1. System call nhập xuất file

3.1.1. Cài đặt syscall CreateFile

- Mô tả cài đặt:
 - Input: char *name (tên file muốn tạo)
 - Output:
 - Trả về 0 nếu thành công
 - Trả về -1 nếu có lỗi
 - Chức năng: Sử dụng Nachos FileSystem Object để tạo một file rỗng.
- Ý tưởng: Chuyển dữ liệu **name** từ system to user, sau đó kiểm tra **name** có hợp lệ không
 - Hợp lệ: tiến hành tạo file mới, nếu tạo thành công thì **return 0**.
 - Không hợp lệ: báo lỗi
 - Hợp lệ nhưng không đủ vùng nhớ để tạo file hoặc tạo gặp lỗi thì **return -1**.
- Các tập tin có liên quan:
 - **filesystem/filesys.cc**: Cung cấp hàm '**bool FileSystem::Create(char* name, int initialSize)**' nhằm tạo file với tên và kích thước ban đầu. Trả về **true** nếu tạo thành công và **false** cho trường hợp tạo thất bại.
 - **userprogs/exception.cc**: Khai báo system call **ex_CreateFile**.
 - **userprogs/syscall.h**: Khai báo hàm **Create()**
 - **machine/machine.cc**: Sử dụng hàm **User2System**.
- Các bước cài đặt:
 - *Bước 1*: Nhận filename từ thanh ghi 4
 - *Bước 2*: Dùng hàm **User2System** chuyển vùng nhớ của filename từ user space sang system space.
 - *Bước 3*: Kiểm tra tính hợp lệ của filename
 - Không hợp lệ: trả về -1 (gán giá trị -1 cho thanh ghi 2) và báo lỗi.
 - Hợp lệ: Tiếp tục Bước 4.
 - *Bước 4*: Tiến hành tạo file với filename
 - Tạo thành công: return 0 (gán giá trị 0 cho thanh ghi 2) và báo thành công

- Không đủ vùng nhớ hoặc tạo gặp lỗi: return -1 (gán giá trị -1 cho thanh ghi 2) và báo lỗi.

3.1.2. Cài đặt syscall Open và Close:

3.1.2.1. Systemcall Open:

- Mô tả cài đặt:
 - Input: char *filename, int type (type = 0: Mở đọc và ghi, type = 1: Mở chỉ đọc)
 - Output: Trả về OpenfileID nếu thành công. Trả về -1 nếu có lỗi
 - Chức năng: Mở file
- Ý tưởng: Chuyển dữ liệu **filename** từ system to user, đọc giá trị **type** từ thanh ghi 5. Kiểm tra tính hợp lệ của filename, sau đó tiến hành mở file, nếu mở được thì chọn cách mở theo type.
- Các tập tin có liên quan:
 - **filesystems/filesys.cc**: Cung cấp hàm '**OpenFile * FileSystem::Open(char *name, int type)**' để mở file theo filename và type tương ứng. Trả về *con trỏ OpenFile** (nếu tìm thấy và mở được). Ngược lại, trả về **NULL**.
 - **userprogs/exception.cc**: Khai báo system call **ex_Open**.
 - **userprogs/syscall.h**: Khai báo hàm **OpenFileId Open(char *name, int type)**
 - **machine/machine.cc**: Sử dụng hàm **User2System**.
- Các bước cài đặt:
 - *Bước 1*: Nhận filename từ thanh ghi 4 và type từ thanh ghi 5.
 - *Bước 2*: Dùng hàm **User2System** chuyển vùng nhớ của filename từ user space sang system space. Sau đó kiểm tra tính hợp lệ của filename
 - Không hợp lệ: trả về -1
 - *Bước 3*: Dùng hàm FindFreeSlot() để tìm khe trống trong bảng mô tả/ bảng quản lý các file mở **¹fileSystem->files**
 - Nếu không tìm được vị trí trống thì trả về -1 (lỗi không thể mở)
 - *Bước 4*: Mở file với type đã chọn.

¹ files là một mảng gồm FILE_MAX phần tử (được định nghĩa trước) chứa danh sách file opening. Nếu index còn trống để cấp cho file thì sẽ return index (~openFileID), ngược lại return -1

- Truy cập vị trí trống vừa tìm được trong mảng **files** của '**fileSystem**', tiến hành gọi hàm **Open(char *name, int type)** để mở file với type tương ứng.
 - Mở thành công: trả về OpenFileID (gán giá trị của khe trống đã tìm thấy cho thanh ghi 2)
 - Không đủ vùng nhớ hoặc tạo gặp lỗi: return -1 (gán giá trị -1 cho thanh ghi 2).

3.1.2.2. Systemcall Close:

- Mô tả cài đặt:
 - Input: fileID
 - Output: Trả về 0 nếu thành công. Trả về -1 nếu có lỗi
 - Chức năng: Đóng file
- Ý tưởng: lấy fileID từ thanh ghi 4, nếu fileID < 0 hoặc lớn hơn số file đã được định nghĩa ban đầu thì báo lỗi. Nếu fileID hợp lệ nhưng không xóa fileID ra khỏi mảng **files** được thì đóng file không thành công.
- Các tập tin có liên quan:
 - **userprogs/exception.cc**: Khai báo system call **ex_Close**.
 - **userprogs/syscall.h**: Khai báo hàm **void Close(OpenFileId id)**
- Các bước cài đặt:
 - *Bước 1*: Nhận fileID từ thanh ghi 4
 - *Bước 2*: Kiểm tra fileID có nằm trong đoạn [0, FILE_MAX²] hay không
 - Không hợp lệ: Lỗi, return -1 (gán giá trị -1 cho thanh ghi 2).
 - *Bước 3*: fileID hợp lệ, tiến hành xóa phần tử fileID trong mảng files (**delete fileSystem->files[id]**)
 - Nếu xóa không thành công: Lỗi, return -1 (gán giá trị -1 cho thanh ghi 2).
 - Xóa thành công: return 0 (gán giá trị 0 cho thanh ghi 2)

3.1.3. Cài đặt systemcall Read và Write:

- Mô tả cài đặt (chung cho cả Read & Write):

² Số lượng phần tử tối đa của mảng file, được định nghĩa trước (Trong cài đặt, nhóm định nghĩa FILE_MAX = 10)

- Input: char* buffer (Để đọc/ ghi), int size (số lượng ký tự cần đọc/ghi), OpenFileID id (file cần mở)
- Output:
 - Cả read và write trả số ký tự đọc, ghi thật sự.
 - Cả Read và Write trả về -1 nếu bị lỗi và -2 nếu cuối file.
- Chức năng: Đọc/ ghi các ký tự vào file
- Các tập tin có liên quan:
 - Filesys/openfile.cc: Các hàm sau
 - `int OpenFile::Read(char *into, int numBytes)`
 - `int OpenFile::Write(char *into, int numBytes)`
 - Giải thích:
 - Trong Class OpenFile, các hàm trên được cung cấp trước để đọc/ ghi một số lượng byte vào file (từ vị trí cho trước)
 - **userprogs/exception.cc**: Khai báo system call **ex_Read** và **ex_Write**.
 - **userprogs/syscall.h**: Khai báo
 - `void Write(char *buffer, int size, OpenFileId id);`
 - `int Read(char *buffer, int size, OpenFileId id);`
 - **machine/machine.cc**: Sử dụng hàm **User2System** và **System2User**
- Các bước cài đặt:
 - *Bước 1*: Đọc addr, sz, id lần lượt từ thanh ghi 4, 5, 6. Xét điều kiện của addr, sz, id
 - Nếu không hợp lệ thì báo lỗi (gán giá trị -1 cho thanh ghi 2).
 - Đồng thời kiểm tra loại tệp có phải stdout (đối với Read) và stdin (đối với Write), nếu đúng thì báo lỗi không thể thực hiện (gán giá trị -1 cho thanh ghi 2).
 - *Bước 2*: Xác định vị trí bắt đầu (start = vị trí bắt đầu của con trỏ trong tệp).
 - *Bước 3*: Chuyển địa chỉ buffer từ user space sang system space
 - *Bước 4 (Đối với Read)*:
 - 4.1. Đọc:
 - Nếu tệp là stdin:
 - Sử dụng 'synchcons->Read()' để đọc từ bàn phím

- Nếu tệp thông thường:
 - Sử dụng **Read** (của OpenFile) để đọc
- 4.2. Ghi số byte thực sự đọc được vào buffer và chuyển địa chỉ buffer từ system space sang user space.
- Bước 4 (Đối với Write):
 - Nếu tệp là stdout:
 - Sử dụng ‘synchron->Write()’ để ghi từ buffer ra màn hình.
 - Nếu tệp đọc/ghi:
 - Sử dụng **Write** (của OpenFile) để ghi vào tệp
- Bước 5: Ghi số ký tự thực tế đã đọc/ghi vào thanh ghi R2
 - Nếu đọc đến cuối file thì trả về -2

3.2. Đa chương, lập lịch và đồng bộ hoá trong NachOS

3.2.1. Chuyển hệ thống từ đơn chương thành đa chương:

3.2.1.1. Trong file “/machine/disk.h”:

Thay đổi định nghĩa ‘SectorSize’ thành 512.

3.2.1.2. Trong file “/userprogs/addrspace.cc”:

Viết hàm khởi tạo AddrSpace(char* filename), thay đổi so với hàm khởi tạo trước đó:

- Thay việc truyền trực tiếp ‘OpenFile*’ thành mở file thông qua ‘fileSystem->Open(filename)’
- Thêm khoá địa chỉ (addrLock) để đảm bảo tránh xung đột địa chỉ khi nhiều tiến trình cùng thực hiện
- Thêm điều kiện kiểm tra số trang trống trong **bitmap** (gPhysPageBitMap). Nếu không đủ trang trống sẽ huỷ việc tạo addrspace và giải phóng tài nguyên.
- Sử dụng gPhysPageBitMap->Find() để lấy trang vật lý mới cho từng trang ảo, thay vì sử dụng i như trước.
- Giải phóng tài nguyên khi Userprogram kết thúc.

3.2.1.3. Trong folder ‘Threads’: sửa file **system.h** và **system.cc**

- Tạo biến toàn cục ở **system.h**, cấp phát và huỷ ở **system.cc**

```
// students' applied change

extern SynchConsole *synchcons;      // the console synchronizer
extern Semaphore *addrLock;          // semaphore
extern BitMap *gPhysPageBitMap;      // quan ly frame
extern PTable *pTab;                 // quan ly tien trinh
extern STable *semTab;                // quan ly semaphore
```

- Thêm các class tương ứng:

- o Nhóm class quản lý tiến trình

- **Class PTable:** Lớp PTable được sinh ra để quản lý toàn bộ tiến trình trong trong Nachos. Trong cài đặt của nhóm, chỉ có 1 thể hiện duy nhất của lớp PTable được sử dụng toàn cục trong toàn bộ Nachos, đây chính là PTable* pTab được khai báo và khởi tạo toàn cục ở file system.h và system.cc. Lớp PTable được khai báo và cài đặt lần lượt ở PTable.h và PTable.cc
- **Class PCB:** Tương ứng với mỗi tiến trình mà ta muốn chạy đa chương khi được nạp vào PTable, sẽ có 1 PCB (Process control block) tương ứng để quản lý 1 tiến trình đấy. PCB trong đồ án lần này được thể hiện trong chính PCB* bm[MAXPROCESS] của chính pTab mà ta đã đề cập. Lớp PCB được khai báo và cài đặt lần lượt ở PCB.h và PCB.cc

```

10 class PCB {
11     private:
12         Semaphore *joinsem;    // Semaphore cho qua trinh join
13         Semaphore *exitsem;    // Semaphore cho qua trinh exit
14         Semaphore *multex;    // Semaphore cho qua trinh truy xuat doc quyen
15         int exitcode;
16         int pid;
17         int numwait;           // So tien trinh da join
18         char FileName[32];     // Ten tien trinh
19         Thread *thread;        // Tien trinh cua chuong trinh
20     public:
21         int parentID;          // ID cua tien trinh cha
22         char boolBG;           // Kiem tra tien trinh nen
23
24         PCB(int id);           // Constructor
25         ~PCB();                // Destructor
26
27         int Exec(char *filename, int pid); // Nap chuong trinh co ten luu trong
                                           // bien file name va processID se la pid
28
29         int GetID()
30         {return pid;} // Tra ve ProcessID cua tien trinh goi thuc hien
31         int GetNumWait(); // Tra ve so luong tien trinh cho
32
33         void JoinWait(); // Tien trinh cha doi tien trinh con ket thuc
34         void ExitWait(); // Tien trinh con ket thuc
35         void JoinRelease(); // Bao cho tien trinh cha tiep tục thực thi
36         void ExitRelease(); // Cho phép tien trinh con ket thuc
37         void IncNumWait(); // Tang so tien trinh cho
38         void DecNumWait(); // Giam so tien trinh cho
39         void SetExitCode(int ec) // Dat exitcode cua tien trinh
40         {exitcode = ec;}
41         int GetExitCode() // Tra ve exitcode
42         {return exitcode;}
43
44         void SetFileName(char* filename); // Dat ten tien trinh
45         char* GetFileName(); // Tra ve ten tien trinh
46     };

```

```

12 class PTable {
13     private:
14         BitMap *bm;
15         PCB *pcb[MAXPROCESS];
16         int psize;
17         Semaphore *bmsem; // DUNG de ngan chan truong hop nap 2
                           // tien trinh cung mot luc
18     public:
19         PTable(int size);
20         ~PTable();
21         int ExecUpdate(char* name); //SC_Exec
22         int ExitUpdate(int); //SC_Exit
23         int JoinUpdate(int id); //SC_Join
24         int GetFreeSlot();
25         bool IsExist(int pid);
26         void Remove(int pid);
27
28         char* GetFileName(int id);
29     };
30
31 #endif
32

```

- Nhóm class quản lý các semaphore:
 - **Class Sem:** Quản lý Semaphore
 - **Class STable :** Quản lý bảng semaphore với tối đa là 10 semaphore

3.2.2. Cài đặt system call Exec

- Mô tả cài đặt:
 - Input: tên chương trình con đã được biên dịch trong Nachos
 - Output: -1 nếu lỗi, SpaceID nếu tạo thành công
 - Chức năng: Nạp vào một chương trình thành một tiến trình con được quản lý bởi PCB và PTable
- Các tập tin có liên quan:
 - **threads/ptable.cc:** hàm ExecUpdate(char* file) thực hiện chạy độc quyền và thêm vào vị trí trống trên PCB, trả về pID và đánh dấu khi tạo thành công, trả về -1 khi thất bại.
 - **userprogs/exception.cc:** Khai báo system call ex_Exec
 - **userprogs/syscall.h:** Điều chỉnh khai báo cho system call ex_Exec và khai báo hàm SpaceID Exec(char* name).
- Các bước cài đặt:
 - *Bước 1:* Lấy các tham số địa chỉ buffer lưu string cần ghi từ thanh ghi số 4
 - *Bước 2:* Chuyển dữ liệu buffer từ User space và system space thông qua User2System()
 - *Bước 3:* Nếu bộ nhớ không đủ thì báo lỗi.
 - *Bước 4:* Mở file. Nếu file không tồn tại thì báo lỗi và trả về -1 (cho thanh ghi 2)
 - *Bước 5:* Gọi hàm pTab->ExecUpdate(name) và lưu id của tiến trình mới vào thanh ghi 2 (nếu chạy thành công)
 - *Bước 6:* Giải phóng bộ nhớ của name và tăng thanh ghi PC.

3.2.3. Cài đặt system call Join và Exit

3.2.3.1. Systemcall Join:

- Mô tả cài đặt:
 - Input: process id của tiến trình muốn join vào tiến trình cha

- Output: -1 nếu id không hợp lệ, hoặc tiến trình đang cố join vào tiến trình không phải tiến trình cha. Ngược lại, trả về exit code cho tiến trình nó đã join.
- Chức năng: Nạp vào một chương trình thành một tiến trình vào hàng chờ được quản lý bởi PCB và Ptable.
- Các tập tin có liên quan:
 - **threads/ptable.cc**: hàm **JoinUpdate(int id)** thực hiện kiểm tra process ID tồn tại trong pCB và thêm vào hàng đợi tới khi tiến trình con kết thúc.
 - **userprogs/exception.cc**: Khai báo system call **ex_Join**
 - **userprogs/syscall.h**: khai báo hàm **int Join(SpaceID id)**.
- Các bước cài đặt:
 - *Bước 1*: Nhận tham số từ thanh ghi 4
 - *Bước 2*: Gọi **pTab->JoinUpdate(id)** để join vào tiến trình cha. Chỉ khi tiến trình con này kết thúc thì tiến trình cha mới được tiếp tục thực thi.
 - *Bước 3*: Ghi kết quả vào thanh ghi 2

3.2.3.2. Syscall Exit

- Mô tả cài đặt:
 - Input: **exitCode** (exitCode trả về cho tiến trình mà nó đang chạy)
 - Output: Trả về process ID tiến trình đã thực hiện block trong đó hoặc trả về - 1 khi bị lỗi
 - Chức năng: Giải phóng một tiến trình con đang trong hàng chờ được quản lý bởi PCB và Ptable
- Các tập tin có liên quan:
 - **threads/ptable.cc**: hàm **int ExitUpdate(int ec)** thực hiện gán exit code và trả về process ID của tiến trình.
 - **userprogs/exception.cc**: Khai báo system call **ex_Exit**
 - **userprogs/syscall.h**: khai báo hàm **void Exit(int exitCode)**
- Các bước cài đặt:
 - *Bước 1*: Nhận tham số từ thanh ghi 4

- *Bước 2:* Gọi pTab->ExitUpdate(exitCode) để giải phóng tiến trình và xin tiến trình cha để kết thúc.
- *Bước 3:* Gọi currentThread->FreeSpace() để thu hồi bộ nhớ.
- *Bước 4:* Gọi currentThread->Finish() để báo scheduler kết thúc tiến trình.

3.2.4. Cài đặt system call CreateSemaphore

- Mô tả cài đặt:

- Input:
 - name: tên của semaphore cần tạo
 - init: giá trị khởi tạo của semaphore
- Output: Trả về chỉ số của semaphore, nếu semaphore đã tồn tại hoặc đã hết ô trống thì trả về -1
- Chức năng: Tạo semaphore với tên cho trước và giá trị value cho trước.

- Các tập tin có liên quan:

- **threads/stable.cc:** hàm **Create(char *name, int init)** thực hiện gán exit code và trả về process ID của tiến trình.
- **threads/stable.h:** hàm **Sem(char*, int)** thực hiện khởi tạo semaphore với tên được cung cấp
- **userprogs/exception.cc:** Khai báo system call ex_CreateSemaphore
- **userprogs/syscall.h:** khai báo hàm int CreateSemaphore(char* name, int semval).

- Các bước cài đặt:

- *Bước 1:* Nhận tham số name và semVal từ thanh ghi 4 và 5. (Chuyển name từ User space sang System space)
- *Bước 2:* Kiểm tra, nếu bộ nhớ đầy thì báo lỗi, gán -1 vào thanh ghi 2.
- *Bước 3:* Gọi semTab->Create(name, semVal) để khởi tạo tiến trình
 - Nếu khởi tạo thất bại, báo lỗi và gán -1 vào thanh ghi 2.
- *Bước 4:* Giải phóng name, ghi kết quả trả về vào thanh ghi 2.

3.2.5. Cài đặt system call Wait và Signal:

3.2.5.1. Systemcall Wait

- Mô tả cài đặt:

- Input: name: tên của tiến trình muốn đưa vào trạng thái chờ
- Output: trả về 0 nếu thành công -1 nếu thất bại.
- Chức năng: Đưa tiến trình với tên tương ứng vào trạng thái chờ
- Các tập tin có liên quan:
 - **threads/stable.cc**: hàm **Wait(Char*)** thực hiện tìm tên của semaphore tương ứng và đưa tiến trình của semaphore đó vào trạng thái chờ.
 - **threads/stable.h**: hàm wait() thực hiện cho đối tượng vào trạng thái chờ.
 - **userprogs/exception.cc**: Khai báo system call ex_Wait
 - **userprogs/syscall.h**: khai báo hàm int Wait(char* name).
- Các bước cài đặt:
 - *Bước 1*: Nhận tham số từ thanh ghi 4. (Chuyển name từ User space sang System space)
 - *Bước 2*: Kiểm tra, nếu bộ nhớ đầy thì báo lỗi, gán -1 vào thanh ghi 2.
 - *Bước 3*: Gọi semTab->Wait(name) để đưa tiến trình vào trạng thái chờ
 - Nếu tiến trình không tồn tại, báo lỗi và gán -1 vào thanh ghi 2.
 - *Bước 4*: Giải phóng name, ghi kết quả trả về vào thanh ghi 2.

3.2.5.2. Systemcall Signal:

- Mô tả cài đặt:
 - Input: name: tên của tiến trình
 - Output: trả về -1 nếu thất bại.
 - Chức năng: Đưa tiến trình chờ vào trạng thái hoạt động
- Các tập tin có liên quan:
 - **threads/stable.cc**: Thủ tục int Signal(Char*) thực hiện tìm tên của semaphore tương ứng và đưa tiến trình của semaphore đó vào trạng thái hoạt động .
 - **threads/stable.h**: Thủ tục signal() thực hiện cho đối tượng hoạt động khi đang trong trạng thái chờ.
 - **userprogs/exception.cc**: Khai báo system call ex_Signal
 - **userprogs/syscall.h**: khai báo hàm int Signal(char* name)
- Các bước cài đặt:

- *Bước 1:* Nhận tham số từ thanh ghi 4. (Chuyển name từ User space sang System space)
- *Bước 2:* Kiểm tra, nếu bộ nhớ đầy thì báo lỗi, gán -1 vào thanh ghi 2.
- *Bước 3:* Gọi semTab->Signal(name) để đưa tiến trình vào trạng thái chờ
 - Nếu tiến trình không tồn tại, báo lỗi và gán -1 vào thanh ghi 2.
- *Bước 4:* Giải phóng name, ghi kết quả trả về vào thanh ghi 2.

3.2.6. Cài đặt system call Seek

Đây là systemcall mà nhóm cài đặt thêm

- Mô tả cài đặt:
 - Input:
 - Pos: vị trí cần chuyển đến trong file
 - Id: id của file cần chuyển
 - Output: trả về pos nếu thành công, -1 nếu thất bại.
 - Chức năng: Dịch chuyển con trỏ đọc file sang vị trí mới
- Các bước cài đặt:
 - *Bước 1:* Nhận tham số từ thanh ghi 4,5 tương ứng với pos và id.
 - *Bước 2:* Kiểm tra tính hợp lệ của id, file có tồn tại không, và có phải file đang trong console không. Nếu lỗi, trả về -1
 - *Bước 3:* Kiểm tra vị trí mới có hợp lệ hay không. Nếu lỗi, trả về -1
 - *Bước 4:* Nếu hợp lệ, gọi hàm Seek(pos) của filesystem để di chuyển con trỏ. Trả về pos cho thanh ghi 2.

3.3. Chương trình minh họa

Xây dựng ứng dụng “Thống kê sử dụng máy nóng lạnh”: Bài toán yêu cầu đồng bộ cho hai vòi nước phục vụ cho các sinh viên, mỗi vòi nước chỉ phục vụ một sinh viên tại một thời điểm.

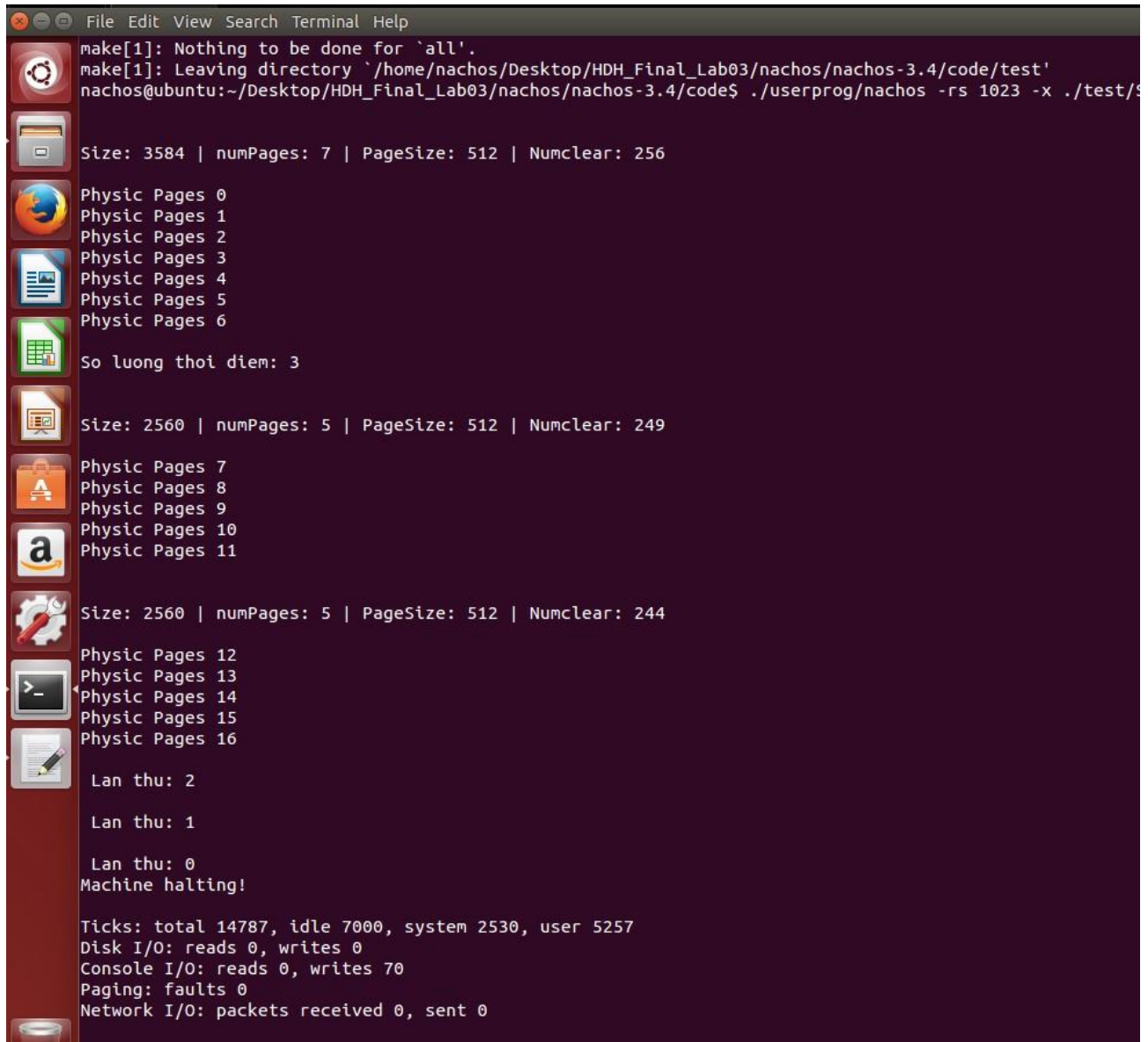
Thực hiện việc đồng bộ, lập lịch đáp ứng yêu cầu bài toán để vòi nước rót nước cho các sinh viên.

- Mỗi vòi nước tại một thời điểm chỉ rót nước cho một sinh viên và có thể sử dụng 2 vòi nước.

- Ta tạo ba tiến trình main (tiến trình chính), tiến trình sinhvien và voinuoc.
- Kết quả vòi nước nào thực hiện rót nước được ghi ở file output.txt.

3.3.1. Demo chương trình:

3.3.1.1. Chương trình thực thi



```

make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory `/home/nachos/Desktop/HDH_Final_Lab03/nachos/nachos-3.4/code/test'
nachos@ubuntu:~/Desktop/HDH_Final_Lab03/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/s

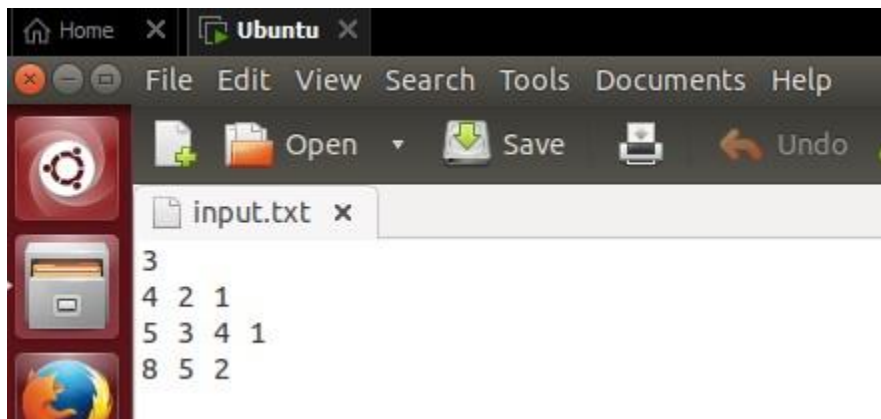
Size: 3584 | numPages: 7 | PageSize: 512 | Numclear: 256
Physic Pages 0
Physic Pages 1
Physic Pages 2
Physic Pages 3
Physic Pages 4
Physic Pages 5
Physic Pages 6
So luong thoi diem: 3

Size: 2560 | numPages: 5 | PageSize: 512 | Numclear: 249
Physic Pages 7
Physic Pages 8
Physic Pages 9
Physic Pages 10
Physic Pages 11

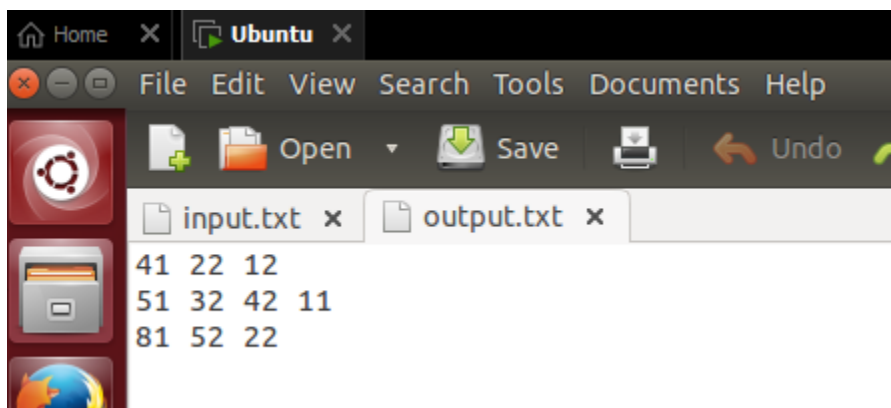
Size: 2560 | numPages: 5 | PageSize: 512 | Numclear: 244
Physic Pages 12
Physic Pages 13
Physic Pages 14
Physic Pages 15
Physic Pages 16
Lan thu: 2
Lan thu: 1
Lan thu: 0
Machine halting!

Ticks: total 14787, idle 7000, system 2530, user 5257
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 70
Paging: faults 0
Network I/O: packets received 0, sent 0
  
```

3.3.1.2. Input file



3.3.1.3. Output file



3.3.2. Hướng dẫn sử dụng chương trình:

- Bước 1: Compile file Nachos bằng cách chuyển đến thư mục “./nachos/nachos-3.4/code/” sau đó chạy lệnh: make
- Bước 2: Chuẩn bị file input ở thư mục code, vẫn ở vị trí thư mục ./nachos/nachos-3.4/code/, ta chạy lệnh: ./userprog/nachos -rs 1023 -x ./test/main.
- Bước 3: Xem kết quả chương trình.

4. TÀI LIỆU THAM KHẢO

[1] Trần Trung Dũng, Phạm Tuấn Sơn (2022). Giáo trình Hệ điều hành. NXB Khoa học và Kỹ thuật, Hồ Chí Minh.

[2] Tài liệu giảng viên cung cấp trong quá trình thực hiện đồ án. Bao gồm:

- [1] Bien dịch và Cài đặt nachos.
- [2] Giao tiếp giữa HDH Nachos và chương trình người dùng.
- [3] Cách Viết Một SystemCall.
- [4] Cách Thêm 1 Lớp Vào Nachos.
- [5] Các Lớp Trong Project 3.docx
- [6] “Constructor_Cua_AddrSpace.pdf”.
- [7] L.V.Long, “HuongDan_Project3.pdf”.
- [8] L.V.Long, “schandle.rar”.
- [9] L.V.Long, “Seminar_HDH_Buoi3.pptx”.
- [10] “Project 3 - Đa chương và đồng bộ hóaFile”

[3] General Nachos Documentation, <https://homes.cs.washington.edu/~tom/nachos/>, truy cập ngày 30/11/2023.

[4] Overview of the Nachos Operating System ,
https://student.cs.uwaterloo.ca/~cs350/common/os_overview.html, truy cập ngày 30/11/2023.