**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**ĐỒ ÁN 3**

**NACHOS – ĐỒNG BỘ HÓA**

25/12/2018

**Giáo viên hướng dẫn:** Phạm Tuấn Sơn

**Nhóm thực hiện:** Nhóm 11

MỤC LỤC

[1. THÔNG TIN THÀNH VIÊN 3](#_Toc533538104)

[2. CÀI ĐẶT SYSTEM CALL Exec VÀ ĐA CHƯƠNG 4](#_Toc533538105)

[2.1. Cài đặt trước 4](#_Toc533538106)

[2.2. Cài đặt các syscall 6](#_Toc533538107)

[2.2.1. Syscall Exec 6](#_Toc533538108)

[2.2.2. Syscall Join 6](#_Toc533538109)

[2.2.3. Syscall Exit 6](#_Toc533538110)

[2.2.4. Syscall CreateSemaphore 6](#_Toc533538111)

[2.2.5. Syscall Up 7](#_Toc533538112)

[2.2.6. Syscall Down 7](#_Toc533538113)

[2.3. Cài đặt đa chương và đồng bộ hóa 7](#_Toc533538114)

[3. DEMO 8](#_Toc533538115)

[3.1. Chương trình shell.c 8](#_Toc533538116)

[3.2. Chương trình ứng dụng semaphore 9](#_Toc533538117)

[TÀI LIỆU THAM KHẢO 10](#_Toc533538118)

# THÔNG TIN THÀNH VIÊN

|  |  |  |  |
| --- | --- | --- | --- |
| **MSSV** | **Họ tên** | **Email** | **Đóng góp** |
| 1612579 | Nguyễn Văn Tài | vantaisetotaba@gmail.com | 25% |
| 1612582 | Phạm Đỗ An Tâm | charatsu98@gmail.com | 25% |
| 1612858 | Huỳnh Minh Huấn | minhhuanhuynh289@gmail.com | 25% |
| 1612772 | Nguyễn Hữu Tứ | nguyenhuutuschool@gmail.com | 25% |

# CÀI ĐẶT SYSTEM CALL Exec VÀ ĐA CHƯƠNG

## Cài đặt trước

* Cài đặt thêm lớp PCB, PTable, STable, Sem.
* Cài đặt trước các syscall hỗ trợ khác (đã làm trong đồ án 1).
* Trong file disk.h sửa lại giá trị sau: SectorSize = 512
* Trong file machine.h sửa lại các giá trị sau:
* NumPhysPages = 128; PageSize = SectorSize
* Khai báo và khởi tạo thêm các biến toàn cục trong system.h và system.cc:
* PTable\* pTab;
* Semaphore \*addrLock;
* STable \* semTab;
* BitMap\* gPhysPageBitMap;
* Cài đặt và khai báo hàm StartProcess\_2(int pID) trong file pcb.cc và pcb.h
  + 1. **Cài đặt lớp PCB**

|  |  |  |
| --- | --- | --- |
|  | Tên | Ý nghĩa |
| Thuộc tính | Semaphore \*joinsem, \*exitsem, \*mutex; | Semaphore cho các quá trình join, exit, truy suất độc quyền. |
| Thread \*thread; | Lớp thread. |
| int pid; | Process ID. |
| int numWait | Số tiến trình đã join. |
| char ThreadName[255] | Tên tiến trình. |
| int exitcode; | Giá trị exitcode. |
| int parentID | ID của tiến trình cha. |
| int JoinStatus | Trạng thái có join với tiến trình nào không. |
| Phương thức | PCB(int id); | Constructor khởi tạo. |
| ~PCB(); | hủy các đối tượng đã tạo. |
| int Exec(char \*filename, int pID); | Nạp chương trình có tên filename và process ID là pID. |
| int GetID(); | Lấy giá trị pID. |
| int GetNumWait(); | Lấy giá trị NumWait. |
| void JoinWait(); |  |
| void ExitWait(); |  |
| void JoinRelease(); |  |
| void ExitRelease(); |  |
| void IncNumWait(); | Tăng numwait. |
| void DecNumWait(); | Giảm numwait. |
| void SetExitCode(int ec); | Set giá trị exitcode. |
| int GetExitCode(); | Get giá trị exitcode. |
| char\* GetNameThread(); | Lấy tên tiến trình. |
| void StartProcess\_2(int pID); | Hàm run process. |

* + 1. **Cài đặt lớp PTable**

|  |  |  |
| --- | --- | --- |
|  | Tên | Ý nghĩa |
| Thuộc tính | BitMap \*bm; | Danh sách bitmap. |
| PCB \*pcb[MAXPROCESS]; | Đối tượng semaphore |
| int psize; | Số lượng pcb; |
| Semaphore \*bmsem; | Dung de ngan chan truong hop nap 2 tien trinh cung luc |
| Phương thức | PTable(int size); | Constructor khởi tạo với giá trị size |
| ~PTable(); | hủy các đối tượng đã tạo |
| int ExecUpdate(char\* filename); | Update tiến trình cần thực thi. |
| int ExitUpdate(int ec); | Update tiến trình cần thoát. |
| int JoinUpdate(int pID); | Update tiến trình cần join. |
| int GetFreeSlot(); | Tìm slot trống để khởi tạo pcb. |
| bool IsExist(int pID); | Kiểm tra xem có tồn tại process ID này chưa. |
| void Remove(int pID); | Xóa một process ID ra khỏi mảng quản lý, khi tiến trình này kết thúc. |
| char\* GetName(int pID); | Lấy tên tiến trình. |

* + 1. **Cài đặt lớp STable**

|  |  |  |
| --- | --- | --- |
|  | Tên | Ý nghĩa |
| Thuộc tính | BitMap \*bm; | quản lý slot trống. |
| Sem\* semTab[MAX\_SEMAPHORE]; | quản lý tối đa 10 đối tượng Sem. |
| Phương thức | STable(); | Constructor khởi tạo. |
| ~STable(); | hủy các đối tượng đã tạo. |
| int Create(char \* name, int init); | Kiểm tra Semaphore “name” chưa tồn tại thì tạo. |
| int Wait(char \* name); | Nếu tồn tại Semaphore “name” thì gọi this->P() để thực thi. Ngược lại báo lỗi. |
| int Signal(char \* name); | Nếu tồn tại Semaphore “name” thì gọi this->V() để thực thi, ngược lại báo lỗi |
| int FindFreeSlot(); | Tìm slot trống. |

* + 1. **Cài đặt lớp Sem (./userprog/sem.h)**

Dùng để quản lý Semaphore

|  |  |  |
| --- | --- | --- |
|  | Tên | Ý nghĩa |
| Thuộc tính | Char name[50] | Tên semaphore |
| Semaphore \*sem | Đối tượng semaphore |
| Phương thức | Sem(char\* na, int i); | Constructor khởi tạo với tên và giá trị semaphore |
| ~Sem(); | hủy các đối tượng đã tạo |
| void wait(); | thực hiện thao tác chờ |
| void signal(); | thực hiện thao tác giải phóng Semaphore |
| char\* GetName(); | Trả về tên của Semaphore |

## Cài đặt các syscall

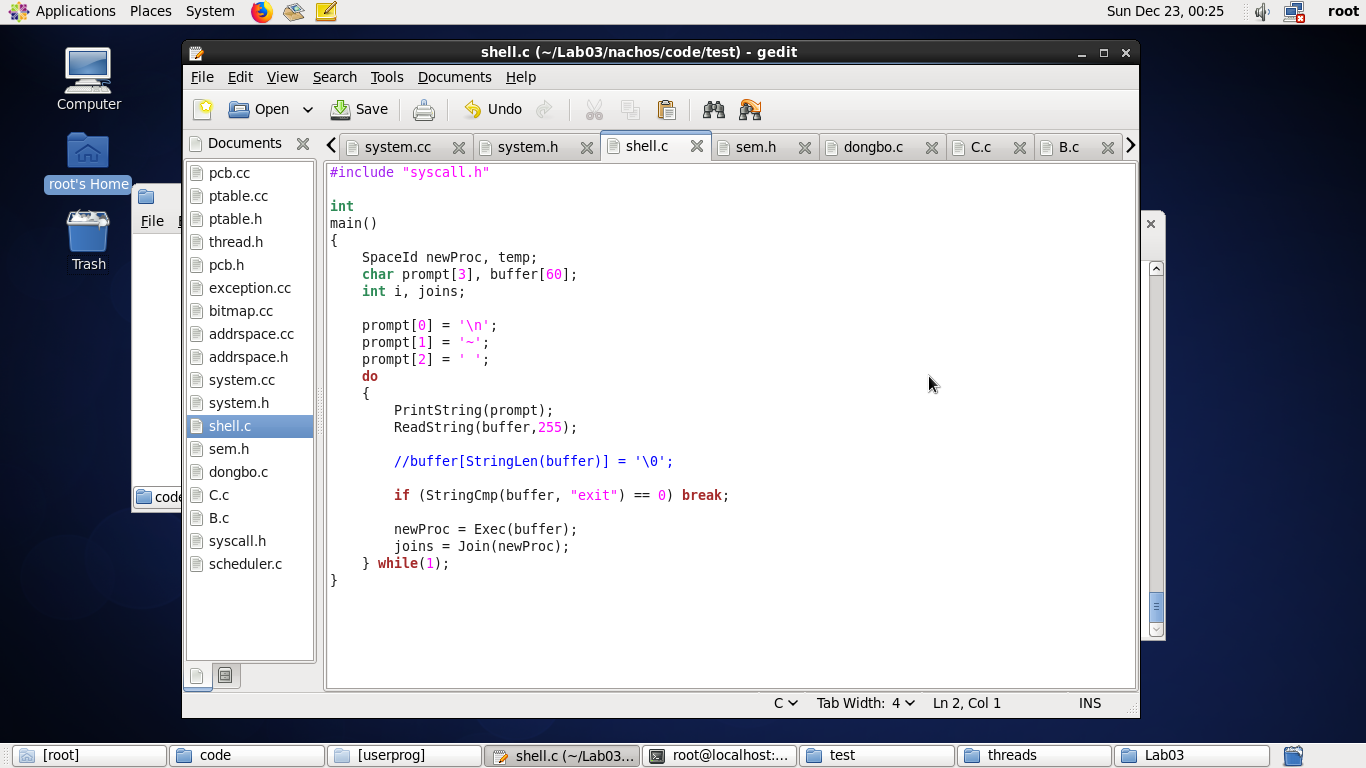
* + 1. **Syscall Exec**
* Input: tên file chương trình (filename)
* Output: pID của chương trình
* Các bước thực hiện:
* Đọc tên file (filename) từ thanh ghi r4
* Kiểm tra tên file có rỗng không? Nếu có trả về -1
* Kiểm tra tên file có trùng với tên của chương trình chính hay thread đang chạy hay không? Nếu có trả về -1.
* Kiếm tra xem file tồn tại hay không? Nếu không tồn tại thì trả về -1.
* Gọi thực thi pTab->ExecUpdate(filename).
* Trả về kết quả nhận được từ hàm ExecUpdate (là giá trị process ID).
  + 1. **Syscall Join**
* Input: process ID trả về từ Syscall Exec
* Output: Exit code của tiến trình
* Các bước thực hiện:
* Đọc ID từ thanh ghi r4 lưu vào id
* Gọi phương thức JoinUpdate(id) của lớp Ptable và lưu kết quả vào result
* Trả về result thông qua thanh ghi r2
  + 1. **Syscall Exit**
* Input: exitStatus
* Output: Trả về exitcode của tiến trình
* Các bước thực hiện:
* Đọc exitstatus từ thanh ghi r4, lưu vào exitStatus
* Gọi phương thức ExitUpdate(exitStatus) của lớp Ptable, lưu kế quả trả về trong result
* Trả về result thông qua thanh ghi r2
  + 1. **Syscall CreateSemaphore**
* Input: tên semaphore, giá trị semaphore.
* Output: kết quả thực hiện
* Các bước thực hiện:
* Đọc địa chỉ “name” từ thanh ghi r4
* Đọc giá trị semaphore “semval” từ thanh ghi r5.
* Gọi thực hiện hàm semTab->Create(name,semval) để tạo Semaphore, nếu có lỗi thì báo lỗi. Lưu kết quả thực hiện vào thanh ghi r2.
  + 1. **Syscall Up**
* Input: tên “name”,
* Output: kết quả trả về khi gọi phương thức Signal(name) của STable.
* Các bước thực hiện:
* Đọc địa chỉ “name” từ thanh ghi r4.
* Tên địa chỉ “name” lúc này đang ở trong user space.
* Gọi hàm User2System đã được khai báo trong lớp machine để chuyển vùng nhớ user space tới vùng nhớ system space.
* Gọi thực thi semTab->Signal(name)
* Lưu kết quả thực hiện vào thanh ghi r2.
  + 1. **Syscall Down**
* Input: tên “name”.
* Output: kết quả trả về khi gọi phương thức Wait(name) của STable.
* Các bước thực hiện:
* Đọc địa chỉ “name” từ thanh ghi r4.
* Tên địa chỉ “name” lúc này đang ở trong user space.
* Gọi hàm User2System đã được khai báo trong lớp machine để chuyển vùng nhớ user space tới vùng nhớ system space.
* Gọi thực thi semTab->Wait(name).
* Lưu kết quả thực hiện vào thanh ghi r2.

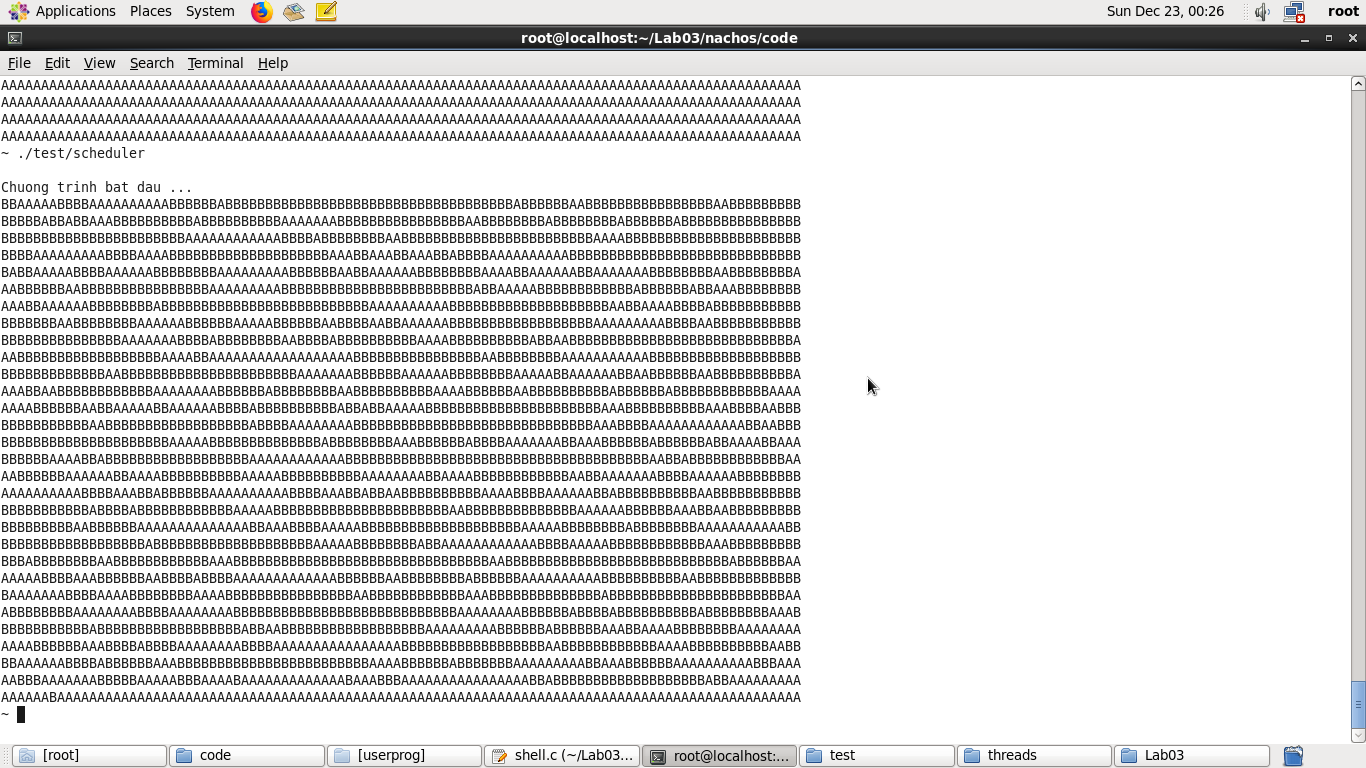
## Cài đặt đa chương và đồng bộ hóa

* Định nghĩa thêm hàm tạo của AddrSpace: AddrSpace(char\* filename)
* Trong đoạn code xây dựng page table của process ta cấp phát vị trí page vật lý như sau: pageTable[i].physicalPage = gPhysPageBitMap->Find()
* Chỉnh sửa lại đoạn code copy chương trình vào bộ nhớ chính

# DEMO

## Chương trình shell.c

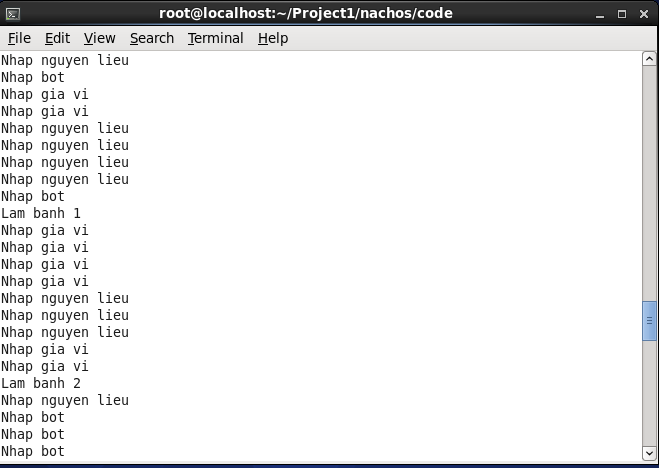
Chương trính shell

Kết quả chạy chương trình shell gọi lệnh “./test/scheduler”

* 1. **Chương trình ứng dụng semaphore**

Chương trình **San\_Xuat\_Banh:**

* Exec 4 tiến trình con:
* Nhap\_Bot: nhập bột cho nhà máy sản xuất, một lần nhập được 1 bịch bột
* Nhap\_Nguyen\_lieu: nhập nguyên liệu cho việc làm bánh, mỗi lần nhập được 1 bịch nguyên liệu
* Nhap\_Gia\_Vi: nhập gia vị cho quá trình làm bánh, mỗi lần nhập một bịch gia vị
* Lam\_Banh: thực hiện làm bánh sau khi có ít nhất 1 bịch bột, 1 bịch nguyên liệu, 1 bịch gia vị, mỗi lần làm được 1 cái bánh
* Hiện tại giới hạn số bánh làm ra là 20 cái
* Kết quả:



# TÀI LIỆU THAM KHẢO

* Tài liệu hướng dẫn của thầy Phạm Tuấn Sơn
* Tham khảo Source của anh Nguyễn Thành Chung:

https://github.com/nguyenthanhchungfit/Nachos-Programing-HCMUS