**Hệ Điều Hành**

**Báo cáo đồ án 1**

Exceptions và các system calls đơn giản

**Thông tin các thành viên tham gia**

1612143 – Trần Phan Phú Duy

1612243 – Nguyễn Thị Thanh Hương

1612XXX – Nguyễn Khắc Nguyên Khang

C:\Users\tdqua_000\Dropbox\SS-Slides\DeCuong-CDIO\Template CDIO v4.2\Templates\Hinh anh\LogoTruong.png

Bộ môn Công nghệ phần mềm

Khoa Công nghệ thông tin

Đại học Khoa học tự nhiên TP HCM

**MỤC LỤC**

1. **Phân chia công việc – Đánh giá mức độ hoàn thành**

|  |  |  |
| --- | --- | --- |
| **Tên thành viên** | **Công việc đảm nhiệm** | **Đánh giá mức độ hoàn thàn** |
| Trần Phan Phú Duy |  |  |
| Nguyễn Thị Thanh Hương | Syscall: SC\_PrintString  Syscall: SC\_Seek  Chương trình: cat |  |
| Nguyễn Khắc Nguyên Khang |  |  |

1. **Cài đặt Tổng quan**
   1. **Biên dịch và cài đặt Nachos** <Khang viết>
   2. **Các bước cài đặt SystemCall**

Theo hướng dẫn của đồ án, chúng ta sẽ hình thành các bước cài đặt thông qua việc cài đặt SystemCall: SC\_Create

* **Bước 1**: ./code/userprog/syscall.h

#define SC\_Create 4 // define syscall để dùng trong switch-case

int Create(char\* name); // Khai báo prototype của hàm

* **Bước 2:** ./code/test/start.c và /code/test/start.s thêm dòng

.globl Create

.ent Create

Create:

addiu $2, $0, SC\_Create

syscall

j $31

.end Create

* **Bước 3**: ./code/userprog/exception.cc sửa điều kiện if thành switch.. case (chỉ sửa một lần duy nhất, lần sau cứ theo format sẵn mà làm cho từng case system call)
* **Bước 4:** Viết chương trình ở mức người dùng để kiểm tra file .c ./code/test

Sử dụng hàm như đã khai báo prototype ở /code/userprog/syscall.h

* **Bước 5:** ./code/test/Makefile

Thêm tên chương trình (tên file) vào dòng all

all: halt shell matmult sort (tên file chương trình ở ./test)

Thêm đoạn sau phía sau malmult

<Tên file>.o: <Tên file>.c

$(CC) $(CFLAGS) -c <Tên file>..c

<Tên file>.: <Tên file>.o start.o

$(LD) $(LDFLAGS) start.o <Tên file>.o -o <Tên file>.coff

../bin/coff2noff <Tên file>.coff <Tên file>

* **Bước 6**: Biên dịch lại nachos, cd tới ./nachos/code chạy lên “gmake all”.
* **Bước 7**: Chạy thử chương trình:

**./userprog/nachos -rs 1023 -x ./test/ <Tên chương trình>**

* 1. **Các bước thêm một lớp mới vào Nachos**<Duy viết>
  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

1. **Cài đặt các System call và Exception**
2. **Cài đặt lại các exception**

Viết lại file exception.cc để xử ly’ tất cả các exceptions được liệt kê trong machine/machine.h.

Mở file exception.cc (code/userprop)viết lại các case này theo các ExceptionType Mỗi exception ta thêm lệnh interrupt->Halt() để tắt hệ điều hành.

1. **Cài đặt hàm IncreasePC():**

Công dụng: Làm tăng Programming Counter, nạp lệnh tiếp theo để thực hiện. Tránh bị loop.

Nguyên lý hoạt động: Đầu tiên lưu giá trị của PC hiện tại cho PC trước, nạp giá trị kế cho PC hiện tại, nạp giá trị kế tiếp nữa cho PC kế.

Chú ý: Viết hàm này trong exception.cc.

1. **Cài đặt 2 hàm hỗ trợ: User2System và System2User**<Duy viết>
2. **Cài đặt syscall CreateFile: int CreateFile(char \* name)**

* Mô tả cài đặt: SC\_CreateFile
* Mục đích: Tạo ra file rỗng với tham số là tên file
* Input: Địa chỉ chứa tên file ở User Space
* Output: 0: Thành công | -1: lỗi
* Cách thức hoạt động:

1. Đọc địa chỉ của tham số name từ thanh ghi r4
2. Sau đó thực hiện chép giá trị ở r4 từ vùng nhớ User sang System bằng hàm User2System(). Giá trị chép được thực sự chính là tên file.
3. Tiếp tục kiểm tra tên file có NULL không, file có được tạo ra với tên file đó không. Nếu thành công thì trả về 0, ngược lại thì trả về -1 vào thanh ghi r2.
4. **Cài đặt System Call:**

**OpenFileID Open(char \*name, int type)**Duy

**int Close(OpenFileID id)**Khang

1. **Cài đặt System Call:**

**int Read (char\* buffer, int charcount, OpenFileID id)**Khang

**int Write (char\* buffer, int charcount, OpenFileID id)**Duy

1. **Cài đặt System Call: int Seek (int pos, OpenFileID id )**

* Mô tả cài đặt: SC\_Seek
* Mục đích: Di chuyển con trỏ đến vị trí thích hợp trong file
* Input: Vị trí cần chuyển: int pos và id của file: OpenFileID id
* Output: Vị trí thực sự (Thành công) | -1 (lỗi)
* Cách thức hoạt động:

1. Đọc tham số pos từ thanh ghi r4 và id của file từ thanh ghi r5.
2. Sau đó tiến hành kiểm tra id của file truyền vào có nằm ngoài bảng mô tả file không (bảng mô tả file gồm 10 file), file cần di chuyển con trỏ có tồn tại không và kiểm tra người dùng có gọi Seek trên console không(Tức là thuộc 2 luồng: stdin(2) và stout(3)). Nếu vi phạm các điều kiện trên thì trả về -1 cho thanh ghi r2 ,ngược lại hợp lệ thì kiểm tra nếu pos = -1 thì gán pos bằng độ dài của file bằng phương thức Length() của lớp FileSystem (vì quy ước: pos = -1 thì chuyển đến cuois file).
3. Gọi phương thức Seek của lớp FileSystem với tham số truyền vào là pos để dịch chuyển con trỏ đến vị trí mong muốn và trả về vị trí dịch chuyển cho r2.
4. **Cài đặt System Call: void PrintString (char\*buffer)**

* Mô tả cài đặt: SC\_PrintString
* Mục đích: Xuat mot chuoi la tham so buffer truyen vao ra man hinh console
* Input: Buffer(char\*)
* Output: Chuỗi đọc được từ buffer in ra console
* Cách thức hoạt động:

1. Đọc địa chỉ của tham số buffer từ thanh ghi r4
2. Sau đó thực hiện copy chuỗi từ vùng nhớ User Space sang System Space với bộ đệm buffer dai 255 ki tu bằng cách sử dụng hàm User2System(). Đồng thời, đếm độ dài thực sự của chuỗi đó.
3. Sử dụng đối tượng gSynchConsole của lớp SysnchConsole gọi đến phương thức Write của chính lớp đó để in chuỗi buffer ra console.
4. **Viết các chương trình Test**
5. **Chương trình createfile để kiểm tra System Call CreateFile**
6. **Chương trình echo**
7. **Chương trình cat**
8. **Chương trình copy**
9. **Demo thực thi chương trình**
10. **Demo biên dịch Nachos**
11. **Demo chương trình createfile**
12. **Demo chương trình echo**
13. **Demo chương trình cat**
14. **Demo chương trình copy**