**Bộ Môn: Hệ Điều Hành**

**Báo cáo đồ án 1**

Exceptions và các system calls đơn giản

**Thông tin các thành viên:**

1612143 – Trần Phan Phú Duy

1612243 – Nguyễn Thị Thanh Hương

1612288 – Nguyễn Khắc Nguyên Khang

Bộ môn Công nghệ phần mềm

Khoa Công nghệ thông tin

Đại học Khoa học tự nhiên TP HCM

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

**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 thành** |
| Trần Phan Phú Duy | Syscall: SC\_Open  Syscall: SC\_Write  Chương trình: echo | **100%** |
| Nguyễn Thị Thanh Hương | Syscall: SC\_PrintString  Syscall: SC\_Seek  Chương trình: cat | **100%** |
| Nguyễn Khắc Nguyên Khang | Syscall: SC\_Close  Syscall: SC\_Read  Chương trình: copy | **100%** |

1. **Cài đặt Tổng quan**
   1. **Biên dịch và cài đặt Nachos** <Khang viết>, chỗ này Khang hình ảnh demo luôn nha
   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 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 lý 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**

**char\* User2System(int virtAdrr, int limit)**

* Input: Địa chỉ vùng nhớ User, giới hạn của buff
* Output: Chuỗi Buf
* Mục đích: Sao chép buff từ vùng nhớ User sang vùng nhớ System
* Cách thức cài đặt:
  1. Khởi tạo 1 chuỗi (kernelBuf) chứa kết quả trả về có độ dài là limit+1
  2. Khởi chạy vòng lặp từ 0 đến limit-1, đọc từ phần tử từ virtAdrr và gán vào kernelBuf. Nếu ký tự đọc ra là 0 thì dừng vòng lặp.
  3. Return kernelBuf.

**int System2User(int virtAdrr, int len, char\* buffer)**

* Input: Địa chỉ vùng nhớ User, giới hạn của chuỗi buffer, chuỗi buffer.
* Output: Số byte được sao chép
* Mục đích: Sao chép buffer từ vùng nhớ System sang vùng nhớ User.
* Cách thức cài đặt:
  1. Nếu len < 0 trả ra -1
  2. Nếu len = 0 trả ra 0
  3. Khởi chạy vòng lặp để đọc từng ký tự trong buffer và ghi vào vùng nhớ User
  4. Trả ra số ra byte được sao chép chính là biến chạy của vòng lặp.

1. **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)**

* Mô tả cài đặt: SC\_Open.
* Mục đích: Mở file với tham số truyền vào gồm tên file và kiểu mở file.
* Input: Địa chỉ tên file trên user space, kiểu mở file.
* Output: trả ra id của file nếu mở thành công, -1 nếu thất bại.
* Cách thức cài đặt:
  1. Đọc địa chỉ tên file từ thanh ghi r4, đọc kiểu mở file từ thanh ghi r5.
  2. Quy định type:
     + Type = 0: mở file ở dạng đọc và ghi.
     + Type = 1: mở file ở dạng chỉ đọc.
     + Type = 2: mở file ở stdin.
     + Type = 3: mở file ở stdout.
     + Ngooài 4 giá trị trên thì báo lỗi.
  3. Thực hiện chép giá trị ở r4 từ vùng nhớ User sang System bằng phương thức User2System. Giá trị chép được chính là tên file người dùng yêu cầu.
  4. Mỗi tiến trình Read/Write sẽ được cấp một bảng mô tả file có kích thước là 10 ô trống (0 ≤ index ≤ 9), trong đó vị trí index=0 dành cho kiểu file stdin, index=1 dành cho kiểu file stdout.
  5. Kiểm tra nếu trong bảng mô tả file còn 1 vị trí index trống thì thực hiện mở file với kiểu type truyền vào.
  6. Nếu type là 0 hoặc 1, tiến hành gán tại vị trí index đó chính là giá trị trả về của phương thức Open(filename, type) đã được định nghĩa trong filesys. Nếu giá trị khác NULL thì kết quả trả về là id của file ứng với index sẽ được ghi vào thanh ghi r2.
  7. Nếu type là 2, nghĩa là file được mở ở dạng stdin. Giá trị trả về là 0 (index trong bảng mô tả file) sẽ được ghi vào thanh ghi r2.
  8. Nếu type là 3, nghĩa là file được mở ở dạng stdout. Giá trị trả về là 1 (index trong bảng mô tả file) sẽ được ghi vào thanh ghi r2.
  9. Tất cả các trường hợp còn lại (không còn ô trống trong bảng mô tả file, mở file thất bại do tên file không tồn tại, type nằm ngoài các giá trị quy định) giá trị trả về -1 sẽ được ghi vào thanh ghi r2.

**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)**

* Mô tả cài đặt: SC\_Open.
* Mục đích: ghi file với tham số là buffer, số ký tự cho phép là charcount và id của file.
* Input: Buffer, số ký tự cho phép, id file.
* Output: Số byte thực sự ghi được nếu thành công, -1 nếu thất bại.
* Cách thức cài đặt:
  1. Đọc địa chỉ buffer truyền vào từ thanh ghi r4, charcount từ thanh ghi r5, id file từ thanh ghi r6.
  2. Thực hiện chép giá trị ở r4 từ vùng nhớ User sang System bằng phương thức User2System. Giá trị chép được chính là buffer người dùng muốn ghi vào file.
  3. Kiểm tra id của file có nằm trong [0, 9] hay không. Nếu nằm ngoài thì trả kết quả là -1 về thanh ghi r2.
  4. Kiểm tra tại vị trí index=id trong bảng mô tả file có giá trị hay không. Nếu bằng NULL (nghĩa là file không tồn tại) thì cũng trả về kết quả -1 về thanh ghi r2.
  5. Nếu thỏa id của file và tại vị trí index=id trong bảng mô tả file có giá trị khác NULL thì ta tiếp tục kiểm tra Type lúc người dùng mở file. Nếu type = 1 (file chỉ đọc) hoặc type = 2 (file ở dạng stdin) thì trả về kết quả -1 về thanh ghi r2.
  6. Nếu thỏa hết những điều kiện bên trên. Tiến hành lấy vị trí hiện tại của con trỏ file (oldPos) bằng phương thức GetCurrentPos() trong lớp FileSystem.
  7. Nếu type = 0 (file ở dạng đọc và ghi). Sử dụng phương thức Write(buf, size) trong lớp FileSystem để ghi file. Giá trị trả về nếu lớn hơn 0 thì ta lưu con trỏ file ở vị trí mới (newPos) bằng vị trí hiện tại thông qua phương thức GetCurrentPos(). Kết quả số byte thực sự ghi được là (newPos – oldPos) được ghi ở thanh ghi r2.
  8. Nếu type = 3 (file ở dạng stdout). Ta duyệt từng ký tự trong buff cho đến khi gặp kí tự ‘\n’ thì dừng. Dùng phương thức Write(char, int) trong lớp SynchConsole để ghi file ra màn hình console. Số byte thực sự đọc được là biến chạy vòng while trừ 1 được ghi vào thanh ghi r2.

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**

**Mục đích:** Tạo ra một file mới từ tên file cố định hoặc do người dùng nhập tư console

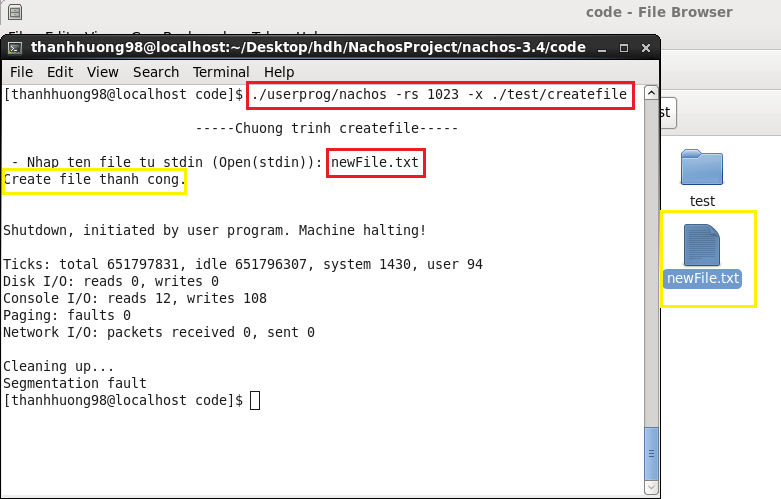
**Cách thức hoạt động**:

1. Gọi system call Open để mở file stdin với type quy ước bằng 2

(stdin: Đọc từ console)

1. Kiểm tra hợp lệ: Nếu mở file thành công thì gọi system call Read đọc tên file vừa nhập từ stdin.Và gọi system call CreateFile để tạo file với tham số truyền vào là tên file đọc được.
2. Đóng file stdin với systsem call Close.

**Hình ảnh demo:**

****

Chạy chương trình createfile và hiện thị kết quả

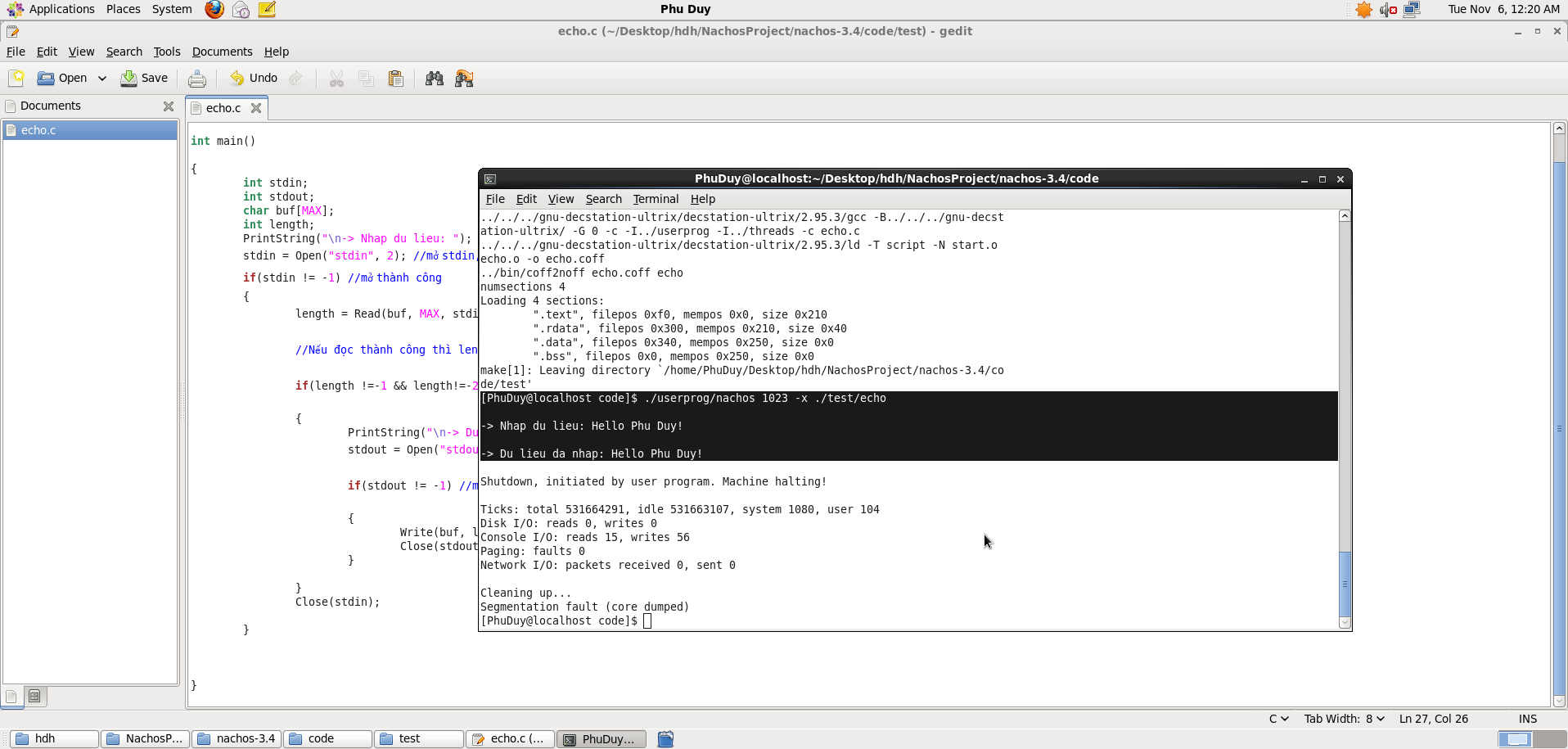
1. **Chương trình echo**

**Mục đích:** Nhập vào 1 chuỗi trên màn hình console, xuất lại chuỗi đó trên màn hình console.

**Cách thức cài đặt:**

1. Gọi hàm Open(…) truyền vào tham số type = 2 để mở file dạng console stdin
2. Sau khi mở file thành công thì gọi hàm Read để đọc dữ liệu người dùng nhập vào từ console, gán vào chuỗi buffer.
3. Tiến hành dùng hàm Open(…) truyền vào tham số type = 3 để mở file dạng console stdout
4. Dùng hàm Write ghi chuỗi buffer vừa đọc đuợc vào file stdout vừa mở

**Hình ảnh demo:**

****

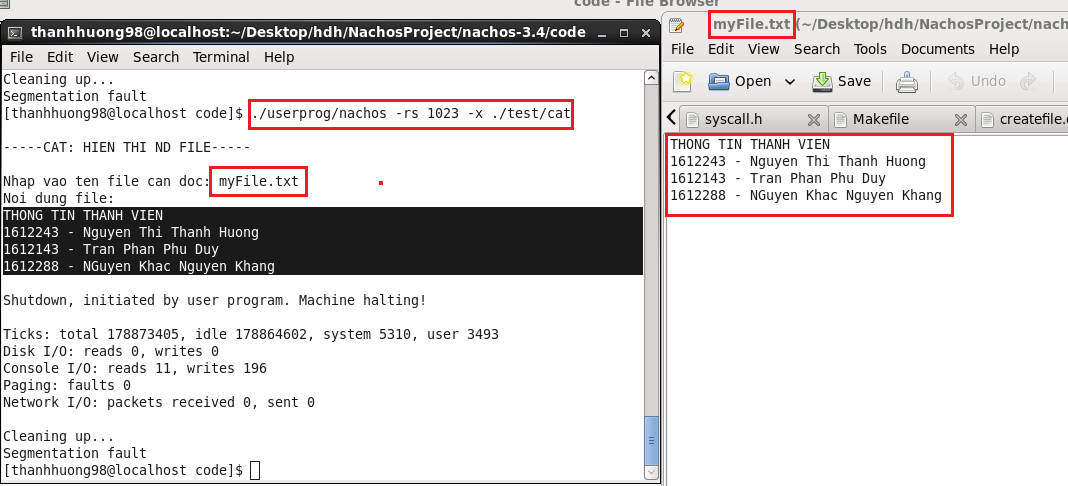
1. **Chương trình cat**

**Mục đích:** Nhập vô tên file, rồi hiện thị nội dung của file đó.

**Cách thức hoạt động:**

1. Gọi system call Open để mở file stdin với type quy ước bằng 2
2. Sau đó gọi system call Open để mở file đó với type bằng 1
3. Kiểm tra hợp lệ: Nếu mở file thành công thì gọi system call Seek để dịch chuyển con trỏ về cuối file lấy kích thước thực sự của file và di chuyển lại con trỏ ra đầu file. Tiến hành đọc từng kí tự trong file bằng system call Read và in từng kí tự đó ra màn hình bằng system call PrintChar cho đến hết kích thước của file.
4. Cuối cùng là đóng file với system call Close.

**Hình ảnh demo:**



Chạy chương trình cat và hiện thị kết quả

1. **Chương trình copy**