BÁO CÁO ĐỒ ÁN 1

EXCEPTIONS VÀ CÁC SYSCALL ĐƠN GIẢN

MSSV 1: 18120308…..Họ tên: Nguyễn Tấn Đạt

MSSV 2: 18120325…...Họ tên: Phạm Anh Đức

MSSV 3: 18120347…...Họ tên: Ngô Hải Hà

Bộ môn Công nghệ phần mềm

Khoa Công nghệ thông tin

Đại học Khoa học tự nhiên TP HCM

MỤC LỤC

1. Thông tin nhóm

STT	MSSV	Họ tên
1	18120308	Nguyễn Tấn Đạt
2	18120325	Phạm Anh Đức
3	18120347	Ngô Hải Hà

2. Nội dung đồ án

2.1. CÀI ĐẶT NACHOS TRÊN LINUX

Sử dụng phần mềm Vmware Player giả lập máy ảo có hệ điều hành Lubutu.

- Tải về file nén nachos.rar có sẵn trên Moodle và giải nén.
- Vào ./nachos/cross-compiler/decstation-ultrix/bin/ Thiết lập thuộc tính excutable thành anyone đối với các file trong thư mục (as, cc1, cpp0, gcc, ld).
- Trong thư mục /code vào Makefile sửa MAKE = gmake thành MAKE = make.
- Tải lệnh make bằng lệnh: sudo apt-get install make
- Cài đặt g++ 5.4.0
- Tiến đến thư mục nachos-3.4/code/, gõ make all để cài NachOS.
- Tiến hành chạy thử Halt: **./userprog/nachos –rs 1023 –x ./test/halt**

2.2. DEFAULT SETTING

2.2.1. Quy trình thực thi của một chương trình trên NachOS

Trước khi chương trình này có thể thực thi được, nó phải được biên dịch.

Quá trình biên dịch chương trình này gồm 3 giai đoạn như sau:

- Chương trình .c được biên dịch bởi cross-compiler gcc thành tập tin .s là mã hợp ngữ chạy trên kiến trúc MIPS
- Tập tin .s này sẽ được liên kết với tập tin start.s để tạo thành tập tin .coff, là định dạng thực thi trên hệ điều hành Linux cho kiến trúc máy MIPS.
- Tập tin .coff được chuyển thành tập tin .noff, là định dạng thực thi trên hệ điều hành Nachos cho kiến trúc máy MIPS, sử dụng tiện ích “coff2noff” được cung cấp sẵn trong thư mục /code/bin/

Sau khi biên dịch thành công, ta có tập tin (.noff) là chương trình thực thi trên

Nachos. Để thực hiện chương trình này vào thư mục code, gõ lệnh: % ./userprog/nachos –rs 1023 –x ./test/"tenchuongtrinh"

2.2.2. Các bước tạo và thực thi một System call

- Vào chỉ mục ./code/userprog/ ta mở file system.h và define 1 System call mới (vd #define SC_New 13), định nghĩa một prototype để xử lý cho syscall này (vd int New(int, ..)). Việc này để định nghĩa một giá trị cho System call để chương trình có thể thực thi được syscall này qua giá trị của nó.

#define SC_New 13

int New(int, ..){…}

- Vào thư muc ./code/test, ở cả file start.s và file start.c ta đều viết 1 đoạn assembly để khai báo ánh xạ giữa systam call đã định nghĩa và phương thức xử lý nó.

```
.globl New
    .ent New
New:
    addiu $2,$0,SC_New
    syscall
    j $31
    .end New
```

- Vào tập tin /code/userprog/exception.cc viết hàm xử lý cho việc bắt lấy system call đã định nghĩa ở trên.

Cần lưu ý rằng sau khi hoàn thành syscall ta phải tăng biến thanh ghi lên bằng lệnh: IncreaseProgramCounter(); để tránh lỗi lặp vô hạn.

2.2.3. Chức năng của hàm System2User và User2System

Hệ điều hành Nachos gồm 2 chế độ thao tác: User's mode và Kernel's mode. Và các giá trị vùng nhớ của 2 chế độ này không thể được chế độ khác sử dụng. Cho nên để có thể giao tiếp giữa 2 chế độ ta phải copy giá trị vùng nhớ từ mode này qua mode khác để sử dụng.

Cụ thể:

System2User: copy vùng nhớ trong thanh ghi từ System Space vào thanh ghi User Space.

User2System: copy vùng nhớ từ trong thanh ghi User Space vào thanh ghi System Space.

2.2.4. Lớp SynchConsole

Lớp này được khởi tạo qua biến toàn cục gSynchConsole(bạn phải khai báo biến này).

Bạn sẽ sử dụng các hàm mặc định của SynchConsole để đọc và ghi. Lưu ý bạn phải chịu trách nhiệm trả về đúng giá trị cho user.

Đọc và ghi với Console sẽ trả về số bytes đọc và ghi thật sự, chứ không phải số bytes được yêu cầu.

Trong trường hợp đọc hay ghi vào console bị lỗi thì trả về -1. Nếu đang đọc từ console và chạm tới cuối file thì trả về -2.

Đọc và ghi vào console sẽ sử dụng dữ liệu ASCII để cho input và output, (ASCII dùng kết thúc chuỗi là NULL (\0)).

2.3: CÀI ĐẶT SYSTEM CALL VÀ EXCEPTIONS

2.3.1. Viết lại exception.cc xử lý các exceptions trong machine.h

Vào trong /code/machine/machine.h, ta kiếm các exceptions trong enum ExceptionType gồm 9 exception, sau đó vào /code/userprog/exception.cc thêm các trường hợp exceptions đã thấy vào đoạn switch case, ngoại trừ No Exception và SyscallException, các exception khác đều chỉ xuất ra lỗi và gọi hàm Halt().

2.3.2. Viết lại cấu trúc điều khiển của chương trình để nhận các Nachos system calls

2.3.3. Cài đặt hàm increaseProgramCounter()

Trong tập tin /code/machine/mipssim.cc ta tìm được đoạn code để tăng Program Couter.

Từ đoạn code này t viết thành hàm increaseProgramCounter() trong file exception.cc bằng cách dùng hàm machine->WriteRegister(thanhghi) và machine->ReadRegister(thanhghi) để gán giá trị của PrevPCReg = giá trị của PCReg, giá trị của PCReg = NextPCReg, giá trị của NextPCReg = giá trị NextPCReg + 4. Vậy là ta đã xong quá trình tăng Program Counter.

2.3.4. Cài đặt system call int CreateFile(char* name)

Ta thực hiện define một giá trị cho hàm và khai báo tên syscall của nó trong /code/userprog/syscall.h:

```
#define SC_CreateFile      4

int CreateFile(char *name);
```

Tại start.s và start.c ta cũng tiến hành cài đặt đoạn code đoạn assembly để khai báo ánh xạ giữa systam call đã định nghĩa và phương thức xử lý nó.

Và ta cũng bắt exception nó trong /code/userprog/exception.cc:

- Nếu độ dài tên file = 0 xuất lỗi “File name is invalid”, trả về thất bại cho chương trình người dùng và thoát ra

- Nếu filename = NULL thì xuất lỗi “Not enough memory in system, trả về thất bại cho chương trình người dùng và thoát ra

- Nếu gọi hàm tạo file: fileSystem->Create(filename,0) trả về 0 thì xuất lỗi “Create ‘filename’ file fail”, trả về thất bại cho chương trình người dùng và thoát ra

- Còn lại thì thông báo thành công, trả về thành công cho chương trình người dùng, tăng PC và thoát ra

2.3.5. Cài đặt system call OpenFileId Open(char* name, int type) và int Close(OpenFileId id)

OpenFileId Open(char* name, int type):

Ta thực hiện define một giá trị cho hàm và khai báo tên syscall của nó trong /code/userprog/syscall.h:

#define SC_Open 5

OpenFileId Open(char* name, int type)

Tại start.s và start.c ta cũng tiến hành cài đặt đoạn code đoạn assembly để khai báo ánh xạ giữa systam call đã định nghĩa và phương thức xử lý nó.

Và ta cũng bắt exception nó trong /code/userprog/exception.cc:

- Nếu fileSystem->index > 9 thì trả về thất bại cho chương trình người dùng và thoát ra bởi vì kích thước bảng mô tả file chỉ có thể lưu đặc tả của 10 files.
- Nếu type = 2 thì xuất ra "Stdin mod" và trả về 0 cho chương trình người dùng rồi thoát ra, đây là console input.
- Nếu type = 3 thì xuất ra "Stdout mode" và trả về 1 cho chương trình người dùng rồi thoát ra, đây là console output
- Nếu dùng lệnh mở file: fileSystem->Open(buffer, type) == NULL thì xuất ra "Can not open file", trả về thất bại(-1) cho chương trình người dùng rồi thoát, vì xảy ra lỗi trong quá trình mở file.
- Còn lại xuất thông báo mở file thành công, trả về cho người dùng index, tăng index xong thoát ra
- Tăng PC.

int Close(OpenFileId id):

Ta thực hiện define một giá trị cho hàm và khai báo tên syscall của nó trong /code/userprog/syscall.h:

#define SC_Close 8

int Close(OpenFileId id);

Tại start.s và start.c ta cũng tiến hành cài đặt đoạn code đoạn assembly để khai báo

ánh xạ giữa systam call đã định nghĩa và phương thức xử lý nó.

Và ta cũng bắt exception nó trong /code/userprog/exception.cc:

- Nếu fileSystem->index < fileId truyền vào thì báo lỗi "Close file failed", trả về cho chương trình người dùng -1 (thất bại), tăng PC và thoát ra.
- Ngược lại thông báo đóng file thành công trả về cho chương trình người dùng 0(thành công).
- Tăng PC và thoát.

2.3.6. Cài đặt system call int Read(char* buffer, int charcount, OpenFileId id) và int Write(char *buffer, int charcount, OpenFileId id)

int Read(char* buffer, int charcount, OpenFileId id):

Ta thực hiện define một giá trị cho hàm và khai báo tên syscall của nó trong /code/userprog/syscall.h:

#define SC_Read 6

int Read(char *buffer, int charcount, OpenFileId id);

Tại start.s và start.c ta cũng tiến hành cài đặt đoạn code đoạn assembly để khai báo ánh xạ giữa systam call đã định nghĩa và phương thức xử lý nó.

Và ta cũng bắt exception nó trong /code/userprog/exception.cc:

- Nếu (id < 0 hoặc id > 9) thì báo lỗi "Can not read file", trả về cho chương trình người dùng -1(thất bại), tăng PC, thoát ra bởi vì kích thước bảng mô tả chỉ có thể lưu giữ 10 file.
- Nếu fileSystem->openfile[id] == NULL làm tương tự TH trên, khi openfile[id] rỗng.
- Nếu fileSystem->openfile[id]->type = 2 thì đây là console in nên ta sử dụng hàm gSynchConsole->Read(buffer, charcount), dùng System2User sau đó

trả về cho chương trình người dùng size đã đọc thực sự, tăng PC và thoát.

- Nếu fileSystem->openfile[id]->type = 3 thì đây là console out nên không thể đọc, xuất ra lỗi "Can not read file stdout", trả về cho chương trình người dùng -1, tăng PC và thoát.
- Còn lại thì bắt đầu đọc file, nếu (fileSystem->openfile[id]->Read(buffer, charcount)) > 0, thì trả về số byte thực sự đã đọc cho chương trình người dùng. Ngược lại khi ở cuối file thì trả về -2.
- Tăng PC

int Write(char *buffer, int charcount, OpenFileId id)

Ta thực hiện define một giá trị cho hàm và khai báo tên syscall của nó trong /code/userprog/syscall.h:

```
#define SC_Write        7

int Write(char *buffer, int charcount, OpenFileId id);
```

Tại start.s và start.c ta cũng tiến hành cài đặt đoạn code đoạn assembly để khai báo ánh xạ giữa systam call đã định nghĩa và phương thức xử lý nó.

Và ta cũng bắt exception nó trong /code/userprog/exception.cc:

- Nếu (id < 0 hoặc id > 9) thì báo lỗi "Can not read file", trả về cho chương trình người dùng -1(thất bại), tăng PC, thoát ra bởi vì kích thước bảng mô tả chỉ có thể lưu giữ 10 file.
- Nếu fileSystem->openfile[id] == NULL làm tương tự TH trên, khi openfile[id] rỗng.
- Nếu (fileSystem->openfile[id]->type = 1 thì đây là file chỉ đọc nên xuất lỗi "Can not write file only read", trả về -1 cho chương trình người dùng, tăng PC và thoát.
- Nếu fileSystem->openfile[id]->type = 2 thì đây là file console in do đó xuất lỗi "Can not write file stdin" ", trả về -1 cho chương trình người dùng, tăng PC và thoát.

- Nếu fileSyetem->openfile[id]->type = 0 thì ta thực hiện lệnh viết kèm kiểm tra (fileSystem->openfile[id]->Write(buffer, charcount)) > 0, nếu đúng thì trả về cho chương trình người dùng số byte đã viết, tăng PC và thoát ra.
- Nếu fileSystem->openfile[id]->type = 3 thì đây là file console out nên ta sẽ dùng lệnh gSynchConsole->Write() để viết, sau đó tăng PC và thoát.

2.3.7. Cài đặt system call int Seek(int pos, OpenFileId id)

Ta thực hiện define một giá trị cho hàm và khai báo tên syscall của nó trong /code/userprog/syscall.h:

#define SC_Seek 11

int Seek(int pos, OpenFileId id);

Tại start.s và start.c ta cũng tiến hành cài đặt đoạn code đoạn assembly để khai báo ánh xạ giữa systam call đã định nghĩa và phương thức xử lý nó.

Và ta cũng bắt exception nó trong /code/userprog/exception.cc:

- Nếu (id < 0 hoặc id > 9) thì báo lỗi "Can not read file", trả về cho chương trình người dùng -1(thất bại), tăng PC, thoát ra bởi vì kích thước bảng mô tả chỉ có thể lưu giữ 10 file.
- Nếu fileSystem->openfile[id] == NULL làm tương tự TH trên, khi openfile[id] rỗng.
- Nếu id = 0 hoặc id = 1 thì đây là file console do đó báo lỗi "Cannot seek on file console", trả về -1(thất bại) cho chương trình người dùng, tăng PC và thoát.
- Nếu pos = -1 ta chuyển con trỏ tới cuối file
- Nếu pos > fileSystem->openfile[id]->Length() hoặc pos < 0 thì đây là phạm vi lỗi không nằm trong file nên báo lỗi "Can not seek file of this fileId 2", trả về -1(thất bại) cho chương trình người dùng. Ngược lại dùng hàm fileSystem->openfile[id]->Seek(pos) và trả về vị trí thực của pos, sau đó tăng PC và thoát.

2.3.8. Viết chương trình createfile kiểm tra system call Createfile

Trong chương trình này ta kiểm tra xem có cài được file hay không bằng cách kiểm tra file được tạo nếu tồn tại là đúng hoặc ngược lại.

2.3.9. Viết chương trình echo

Ở chương trình này ta sẽ cài đặt trước một syscall PrintString() hàm này sẽ in chuỗi String trên ra màn hình console và syscall ScanString() giúp đọc dữ liệu từ console mà người dùng nhập vào bằng Lớp SynchConsole tối đa MaxFileLength kí tự.

2.3.10. Viết chương trình cat

Trong chương trình này ta sẽ gọi lại một syscall đã được cài là Open(char* name, int type) để mở file trên.

Sau đó dùng syscall Read để đọc nội dung trong file và in ra màn hình console bằng syscall PrintString

2.3.11. Viết chương trình copy

Tại chương trình copy ta sẽ dùng các syscall Open, Read, CreateFile và Write để tạo, đọc và ghi nội dung file từ file gốc qua file đích.

Đầu tiên ta yêu cầu người dùng nhập file nguồn, file đích.

Nếu mở file nguồn không được thì thông báo không mở được file nguồn.

Nếu mở được file nguồn, ta sẽ tạo file đích và copy từng kí tự từ file nguồn vào file đích bằng system call Read và Write.

2.4: DEMO

https://youtu.be/DWuSaYUScpo

2.5: TÀI LIỆU THAM KHẢO

Ebook:

http://read.pudn.com/downloads161/ebook/733633/Nachos_CanBan/Nachos_CanBan.pdf

Youtube: Nguyễn Thành Chung

https://www.youtube.com/playlist?list=PLRgTVtca98hUgCN2_2vzsAAXPiTFbvHpO

Github:

https://github.com/ngankhanh98/nachos