Đại học Khoa học tự nhiên

Môn học: Hệ điều hành

Hệ thống tập tin

Thành viên nhóm 18:

Nguyễn Trần Minh Quang - 20127298

Lê Hoàng Long - 18127047

Lê Tiến Đạt - 21127569

Giảng viên:

Phạm Tuấn Sơn

Mục lục

1 Thay đổi cấu trúc hệ điều hành Nachos

1.1 Quản lý program counter

Mặc định hệ điều hành Nachos chưa cài đặt xử lý các exception, các syscall nên chưa có xử lý tăng **program counter** khi chương trình thực thi xong.

Nếu không tăng sau khi chương trình thực hiện xong thì sẽ dẫn đến hiện tượng loop trong hệ điều hành Nachos.

Qua khảo sát file *machine.h* ta thấy được có 3 thanh ghi quản lý **program counter** là:

Listing 1: Định nghĩa PC

```
#define PCReg       34    // Current program counter
#define NextPCReg   35    // Next program counter
#define PrevPCReg   36    // Previous program counter
```

Qua đó nhóm em đã bắt đầu định nghĩa **IncreasePC()** là một method trong lớp **Machine** ở *machine.h*

Listing 2: Định nghĩa IncreasePC()

```
class Machine {
    public:
        // [ ... ]
        void IncreasePC();

    private:
        // [ ... ]
};
```

1.1.1 Cài đặt IncreasePC()

Thực hiện tăng program counter qua các bước:

Bước 1: Chép giá trị từ thanh ghi *PCReg*, *NextPCReg* vào lần lượt các thanh ghi *PrevPCReg*, *PCReg*

Bước 2: tăng giá trị thanh ghi *NextPCReg* với một giá trị là **4** vì Nachos là một hệ thống 32-bit

1.2 Cài đặt sử dụng SynchConsole

Để hỗ trợ đọc và ghi ra console ta cần sử dụng lớp **SynchConsole**, để thuận tiện cho việc thuận tiện nhóm em đã định nghĩa biến toàn cục **synchConsole** ở *system.h*

Listing 3: Định nghĩa biến synchConsole

```
extern SynchConsole *synchConsole;
```

1.3 Giao tiếp bộ nhớ kernel space và user space

Bọn em định nghĩa 3 method mới vào lớp **Machine** đó là **BorrowMemory()**, **TransferMemory()** và **BorrowMemory()**

Hàm **BorrowMemory()** có nhiệm vụ chuyển bộ nhớ từ user space vào kernel space, sẽ nhận vào địa chỉ bộ nhớ ở user space và kích thước, sử dụng **Machine#ReadMem()** đọc một khoảng có kích thước được yêu cầu từ bộ nhớ ở user space và chép nó vô một mảng được allocate ở kernel space. Trả về một pointer nếu thành công, **NULL** nếu thất bại

Hàm **TransferMemory()** có nhiệm vụ chuyển bộ nhớ từ kernel space vào user space, sẽ nhận vào một pointer, kích thước và địa chỉ bộ nhớ ở user space, sử dụng **Machine#WriteMem()** sao chép thông tin từ kernel space vào địa chỉ với kích thước được yêu cầu. Trả về **TRUE** nếu thành công, và ngược lại **FALSE** nếu thất bại

Hàm **BorrowString()** nhiệm vụ và cách hoạt động giống **BorrowMemory()** nhưng chỉ cần nhận địa chỉ bộ nhớ ở user space và sẽ dừng khi gặp ký tự **NULL** hoặc ghi đạt đến giới hạn là **256** ký tự

1.4 Xử lý exception và hộ trợ gọi các syscall

Với hầu hết các exception, xử dụng macro **DEBUG** để in ra exception để người dùng biết được lỗi gì đã xảy

Với exception **NoException** không gì xảy ra thì chỉ tăng PC

Với exception **SyscallException**, gọi hàm **SyscallHandler()** mà nhóm em định nghĩa, hàm này sẽ xử lý và trả về một số nguyên tương ứng với tuỳ response mong muốn của các syscall, sau đó ghi vào thanh ghi số 2, rồi tăng program counter

Riêng syscall **Halt** thì sẽ **Halt** cả hệ điều hành Nachos luôn không cần gọi **SyscallHandler()** hay tăng program counter

Và vì đồ án yêu cầu tạo thêm 2 syscall **Seek** và **Delete** nhóm em đã định nghĩa thêm ở *syscall.h*, *start.s* và *start.c*

1.5 Sửa đổi hệ thống filesystem

1.5.1 Quy định về open mode của file

Để thuận tiện nhóm của em đã quy định **open mode** của file theo giá trị **1** hoặc **0** lần lượt tương ứng **cho phép** hoặc **không cho phép** tại các vị trí được quy định trước đó

Vị trí 0: Cho phép đọc
Vị trí 1: Cho phép ghi
Vị trí 2: Sử dụng console

Như vậy file có quyền đọc ghi sẽ có open mode là **0x1 | 0x2 = 0x3**

1.5.2 Sửa đổi lớp OpenFile

Khảo sát định nghĩa lớp **OpenFile** nhóm em thấy rằng hiện tại lớp này được sử dụng để giao tiếp với hệ thống file riêng của Nachos và một lớp khác đè lên nếu có define *FS_STUB* để giao tiếp trực tiếp với hệ thống Linux

Tuy nhiên trong đồ án này cần giao tiếp trực tiếp với hệ thống Linux và các định nghĩa sẵn kia không đủ nên đã thay đổi lại lớp này

Nhận vào **tên** và **open mode** với **open mode** mặc định sẽ là **0x1 | 0x2 = 0x3**

Ngoài ra bọn em còn chiếm dụng 2 tên là **//stdin** và **//stdout**, khi khởi tạo bằng 2 tên này **open mode** sẽ lần lượt là **0x1 | 0x4 = 0x5** và **0x2 | 0x4 = 0x6**

Các hàm Read và Write sẽ kiểm tra quyền theo **open mode** và nếu được quyền sử dụng console sẽ gọi hàm đọc ghi console tương ứng, nếu không sẽ gọi hàm đọc ghi file trên Linux tương ứng

Hàm Unlink sẽ gọi hàm Unlink tương ứng trên Linux, không cho phép sử dụng nếu là console

1.5.3 Sửa đổi lớp FileSystem

Nhóm em đã thống nhất lại lớp FileSystem và thêm một số method private mới

Listing 4: Thay đổi lớp FileSystem

```
class FileSystem {
    public:
        // [ ... ]

    private:
        OpenFile **table;
        int capacity;
        bool GrowTable();
};
```

Mảng **table** sử dụng để lưu các file đã được mở

Biến **capacity** cho biết sức chứa của **table**

Hàm **GrowTable()** sẽ được gọi khi **table** không còn sức chứa, nó sẽ tạo một mảng mới có sức chứa lớn hơn, sao chép thông tin từ **table** qua, xoá đi **table** cũ và trỏ tới mảng mới. Hoạt động khá giống lớp **Vector** của C++ chỉ khác cách tăng sức chứa.

Ngoài ra lúc construct nhóm em sẽ mở 2 files **//stdin** và **//stdout** để 2 **OpenFileId** có giá trị 0, 1 sẽ lần lượt tương ứng với **ConsoleInput** và **ConsoleOutput**

2 Cài đặt các syscall về hệ thống tập tin

Nhóm em đã tách xử lý các syscall liên quan tới hệ thống tập tin ở riêng file *syscall_fs.cc*

2.1 Syscall Create

Mục đích của syscall này là để tạo ra một file trống trên hệ điều hành Nachos

Bước 1: Lấy địa chỉ tên mong muốn ở thanh ghi số 4

Bước 2: Dùng hàm **Machine#BorrowString()** để lấy tên mong muốn

Bước 3: Gọi **FileSystem#Create()** tương ứng

Bước 4: Trả về response đã lấy được từ bước trên

2.2 Syscall Open

Mục đích của syscall này là để mở file

Bước 1: Lấy địa chỉ tên mong muốn ở thanh ghi số 4

Bước 2: Lấy open mode ở thanh ghi số 5

Bước 3: Dùng hàm **Machine#BorrowString()** để lấy tên mong muốn

Bước 4: Gọi **FileSystem#Open()** tương ứng

Bước 5: Trả về **OpenFileId** đã lấy được từ bước trên hoặc -1 nếu trả về lỗi

2.3 Syscall Close

Mục đích của syscall này là để đóng file

Bước 1: Lấy **OpenFileId** ở thanh ghi số 4

Bước 2: Gọi **FileSystem#Close()** tương ứng

Bước 3: Trả về 0 nếu thành công, -1 nếu bước trên trả về lỗi

2.4 Syscall Read

Mục đích của syscall này là để đọc file

Bước 1: Lấy địa chỉ buffer để đọc vào ở thanh ghi số 4

Bước 2: Lấy kích thước ở thanh ghi số 5

Bước 3: Lấy **OpenFileId** ở thanh ghi số 6

Bước 4: Lấy **OpenFile** tương ứng qua **FileSystem#Get()**

Bước 5: Trả về -1 nếu không lấy được **OpenFile**

Bước 6: Tạo một buffer ở kernel space để đọc vào

Bước 7: Trả về -1 nếu không tạo được buffer

Bước 8: Sử dụng **OpenFile#Read** để đọc vào buffer mới tạo

Bước 9: Sử dụng **Machine#TransferMemory** để chép thông tin từ buffer vào địa chỉ buffer ở user space lấy được ở **bước 1**

Bước 10: Trả về số bytes đã đọc được từ bước trên hoặc -1 nếu trả về lỗi

2.5 Syscall Write

Mục đích của syscall này là để ghi file

Bước 1: Lấy địa chỉ buffer để ghi vào ở thanh ghi số 4

Bước 2: Lấy kích thước ở thanh ghi số 5
Bước 3: Lấy **OpenFileId** ở thanh ghi số 6
Bước 4: Lấy **OpenFile** tương ứng qua **FileSystem#Get()**
Bước 5: Trả về -1 nếu không lấy được **OpenFile**
Bước 6: Nếu kích thước buffer lấy được ở **bước 2** có giá trị âm, thì lấy dữ liệu buffer bằng cách sử dụng **Machine#BorrowString** và cập nhật kích thước, còn nếu ngược lại thì sử dụng **Machine#BorrowMemory**
Bước 7: Trả về -1 nếu không chép được dữ liệu từ buffer
Bước 8: Sử dụng **OpenFile#Write** để ghi từ buffer mới sao chép được
Bước 9: Trả về số bytes đã ghi được từ bước trên hoặc -1 nếu trả về lỗi

2.6 Syscall Delete

Mục đích của syscall này là để xoá file ra khỏi hệ điều hành Nachos
Bước 1: Lấy địa chỉ tên mong muốn ở thanh ghi số 4
Bước 2: Dùng hàm **Machine#BorrowString()** để lấy tên mong muốn
Bước 3: Gọi **FileSystem#Remove()** tương ứng
Bước 4: Trả về 0 nếu thành công hoặc -1 nếu xảy ra lỗi ở bước trên

2.7 Syscall Seek

Mục đích của syscall này là để di chuyển con trỏ của file tới vị trí mong muốn
Bước 1: Lấy vị trí con trỏ mong muốn ở thanh ghi số 4
Bước 2: Lấy **OpenFileId** ở thanh ghi số 5
Bước 3: Lấy **OpenFile** tương ứng qua **FileSystem#Get()**
Bước 4: Trả về -1 nếu không lấy được **OpenFile**
Bước 5: Gọi **OpenFile#Seek()** tương ứng
Bước 3: Trả về 0 nếu thành công, -1 nếu bước trên trả về lỗi