**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**-------**



**BÁO CÁO PROJECT 3**

**MÔN HỌC: HỆ ĐIỀU HÀNH**

**Sinh viên thực hiện:** 20120443 - Nguyễn Tấn Chữ

20120444 - Nguyễn Chí Công

20120447 - Trịnh Quốc Cường

20120383 - Nguyễn Đức Tiến

20120446 - Nguyễn Đình Cường

**| Giáo viên hướng dẫn |**

Thầy Lê Viết Long

Thành Phố Hồ Chí Minh - 2022

MỤC LỤC

[Phần chia công việc 3](#_Toc119699079)

[I/ Tìm hiểu và cài đặt tổng quan 4](#_Toc119699080)

[1. Tổng quan về NachOS 4](#_Toc119699081)

[2. Cài đặt và biên dịch NachOS 4](#_Toc119699082)

[3. Quá trình biên dịch trên NachOS 4](#_Toc119699083)

[4. Cài đặt tổng quan 5](#_Toc119699084)

[II/ Cài đặt các System Calls 6](#_Toc119699085)

[1. Cài đặt system call CreateFile 6](#_Toc119699086)

[2. Cài đặt system call Open và Close 7](#_Toc119699087)

[3. Cài đặt system call Read và Write 7](#_Toc119699088)

[4. Cài đặt các system call 7](#_Toc119699089)

[Tài Liệu Tham Khảo 10](#_Toc119699090)

# Phần chia công việc

|  |  |  |  |
| --- | --- | --- | --- |
| **MSSV** | **Họ và tên** | **Công việc** | **Mức độ hoàn thành** |
| 20120443 | Nguyễn Tấn Chữ |  | 100% |
| 20120444 | Nguyễn Chí Công |  | 100% |
| 20120447 | Trịnh Quốc Cường |  | 100% |
| 20120383 | Nguyễn Đức Tiến | * Viết system call CreateFile * Viết system call Open và Close * Viết system call Write và Read | 100% |
| 20120446 | Nguyễn Đình Cường |  | 100% |

# I/ Tìm hiểu và cài đặt tổng quan

## Tổng quan về NachOS

* NachOS (Not Another Completely Heuristic Operating System) là phần mềm mã nguồn mở giả lập một máy tính ảo và một số thành phần cơ bản của hệ điều hành chạy trên máy tính ảo này nhằm giúp cho việc tìm hiểu và xây dựng các thành phần phức tạp của hệ điều hành.

## Cài đặt và biên dịch NachOS

* Sử dụng VNware Workstation để tạo máy ảo và cài đặt hệ điều hành Ubuntu 14.04 lên trên máy ảo.
* Sau đó tiến hành cài trình biên dịch gcc/g++ trong Ubuntu
* Tiến hành biên dịch NachOS ( mã nguồn NachOS đã được cấu hình với cross-compiler-MIPS)

## Quá trình biên dịch trên NachOS

* Trong thư mục ../nachos-3.4/code/test có file hợp ngữ start.s, khi biên dịch, file này sẽ biên dịch các file mã nguồn <tenfile X>.c thành file hợp ngữ có đuôi <tenfile X>.s
* Sau đó file <tenfile X>.s này sẽ liên kết với file start.s tạo thành 1 file <tenfile X>.coff ( bao gồm file <tenfile X>.o và start.o ), đây là dạng file thực thi trên Linux vơi kiến trúc MIPS.
* File <tenfile X>.coff sẽ được công cụ coff2noff ( được viết sẵn trong NachOS) chuyển thành file <tenfile X>.noff , đây chính là dạng file thực thi trên NachOS với kiến trúc MIPS.
* Quá trình biên dịch này có thể mô tả qua 1 ví dụ với file ‘haft.c’ sau đây:

Diagram, schematic

Description automatically generated

## Cài đặt tổng quan

1. Cài đặt trước

* Để sao chép được vùng nhớ từ User sang System và từ System sang User, tiến hành cài đặt lần lượt 2 hàm: Char\* User2System(int virtAddr, int limit) và int System2User(int virtAddr, int len, char\* buffer) trong file ../code/userprog/exception.cc
* Để tránh việc NachOS sẽ bị vòng lặp gọi thực hiện system call này mãi mãi thì cần tăng program counter trước khi system call trả kết quả về. Tiến hành cài đặt hàm tăng Program counter trong file ../code/userprog/exception.cc

1. Cài đặt system calls:

* Trong file ../code/userprog/syscall.h, tiến hành khai báo các system call sẽ cài đặt.
* Để NachOS có thể call những system call đã khai báo, cần phải thêm 1 số mã lệnh MIPS cho từng system call vào 2 file ../code/test/start.c và ../code/test/start.s. 1 đoạn lệnh ví dụ về system call ReadInt:

Graphical user interface

Description automatically generated with low confidence

* Các system call còn lại đều có cùng format như đoạn lệnh trên. Tiến hành cài đặt lần lượt vào 2 file ở trên

# II/ Cài đặt các System Calls

## Cài đặt system call CreateFile

* CreateFile system call sẽ sử dụng Nachos FileSystem Object để tạo một file rỗng. Chuỗi filename đang ở trong user space, có nghĩa là buffer mà con trỏ trong user space trỏ tới phải được chuyển từ vùng nhớ user space tới vùng nhớ system space. System call CreateFile trả về 0 nếu thành công và -1 nếu có lỗi.
* Cài đặt CreateFile:

+ Bước 1: Ta sẽ đọc địa chỉ của tên file được lưu trong thanh ghi r4.

+ Bước 2: Ta thực hiện chép giá trị ở thanh ghi r4 từ vùng nhớ user space tới vùng nhớ system space bằng hàm User2System(). Giá trị được chép là tên file mình sẽ tạo.

+ Bước 3: Sau đó ta sẽ kiểm tra tên file vừa được chép xem tên file có NULL không, file có được tạo ra với cái tên đó không. Nếu như kiểm tra đúng thì sẽ báo lỗi và trả về -1 vào thanh ghi r2. Ngước lại thì thành công và trả về 0 vào thanh ghi r2.

## Cài đặt system call Open và Close

* User program có thể mở 2 loại file, file chỉ đọc và file đọc và ghi. Mỗi tiến trình sẽ được cấp một bảng mô tả file với kích thước cố định. Đồ án này, kích thước của bảng mô tả file là có thể lưu được đặc tả của 10 files. Trong đó, 2 phần tử đầu, ô 0 và ô 1 để dành cho console input và console output.
* System call mở file phải làm nhiệm vụ chuyển đổi địa chỉ buffer trong user space khi cần thiết và viết hàm xử lý phù hợp trong kernel. System call Open sẽ trả về id của file (OpenFileID = một số nguyên), hoặc là -1 nếu bị lỗi.
* Mở file có thể bị lỗi như trường hợp là không tồn tại tên file hay không đủ ô nhớ trong bảng mô tả file. Tham số type = 0 cho mở file đọc và ghi, = 1 cho file chỉ đọc. Nếu tham số truyền bị sai thì system call phải báo lỗi. System call sẽ trả về -1 nếu bị lỗi và 0 nếu thành công.
* Quy ước tham số type:

+ type = 0: file đọc và ghi

+ type = 1: file chỉ đọc

+ type = 2: stdin

+ type = 3: stdout

* Cài đặt Open:

+ Bước 1: Ta đọc địa chỉ của tên file trong thanh ghi r4, đọc giá trị cho tham số type trong thanh ghi r5.

+ Bước 2: Kiểm tra tham số type có nằm trong đoạn [0, 3] không. Sau đó kiểm tra index của file có nằm trong bảng mô tả file không [0, 9].

+ Bước 3: Nếu thỏa 2 điều kiện trên, thì ta sẽ chép giá trị ở thanh ghi r4 bằng hàm User2System.

+ Bước 4: Ta sẽ kiểm tra file đang mở ở type nào. Nếu type = 0 hoặc type = 1. Thi ta sẽ ghi giá trị = index – 1 vào thành ghi r2. Nếu type = 2 thì ta sẽ ghi giá trị 0 vào thanh ghi r2. Nếu type = 3 thì ta sẽ ghi giá trị 1 vào thanh ghi r2. Thanh ghi r2 sẽ lưu lại id của file.

+ Trường hợp mở file không tồn tại hoặc 2 điều kiện ban đầu không thỏa thì trả về -1 cho thanh ghi r2 để báo lỗi.

* Cài đặt Close:

+ Bước 1: Ta đọc địa chỉ tên file trong thanh ghi r4

+ Bước 2: Sau đó ta kiểm tra file cần đóng có tồn tại không bằng openf[id] được cài đặt trong lớp FileSystem. Ngoài ra, ta sẽ kiểm tra id đó có nằm trong bảng mô tả file không.

+ Bước 3: Nếu 2 điều kiện đều thỏa thì ta sẽ xóa dữ liệu trong openf[id] và gán lại bằng NULL. Gán thanh ghi r2 bằng 0.

+ Bước 4: Nếu có 1 trong 2 không thỏa và báo lỗi thì ta sẽ gán giá trị -1 vào thanh ghi r2 và báo lỗi.

## Cài đặt system call Read và Write

* Các system call đọc và ghi vào file với id cho trước. Bạn cần phải chuyển vùng nhớ giữa user space và system space, và cần phải phân biệt giữa Console IO (OpenFileID 0, 1) và File.
* Lệnh Read và Write sẽ làm việc như sau: Phần console read và write, bạn sẽ sử dụng lớp SynchConsole. 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, tuy nhiên 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)). Phần đọc, ghi vào file, bạn sẽ sử dụng các lớp được cung cấp trong file system. Sử dụng các hàm mặc định có sẳn của filesystem và thông số trả về cũng phải giống như việc trả về trong synchconsole. Cả read và write trả số kí tự đọc, ghi thật sự. Cả Read và Write trả về -1 nếu bị lỗi và -2 nếu cuối file. Cả Read và Write sử dụng dữ liệu binary.
* Cài đặt Read:

+ Bước 1: Ta đọc tham số buffer từ thanh ghi r4, tham số charcount từ thanh ghi r5 và id từ thanh ghi r6.

+ Bước 2: Ta kiểm tra id có nằm trong bảng mô tả file không. File cần đọc có tồn tại không và file đó có type khác 3 ko.

+ Bước 3: Trường hợp vi phạm 1 trong 3 điều kiện: Chương trình báo lỗi và trả về -1 cho thanh ghi r2.

+ Bước 4: Trường hợp hợp lệ thì ta lấy vị trí con trỏ trong file bằng phương thức GetCurrentPos() trong lớp FileSystem gọi là OldPos. Sau đó chép giá trị từ thanh ghi r4 từ phía User sang System bằng User2System. Giá trị chép là buffer (chuỗi ký tự).

+ Bước 5: Ta xét trường hợp type = 2. Nếu thỏa ta sẽ gọi phương thức Read của gSynchConsole để đọc buffer có độ dài charcount. Sau đó trả số byte đọc được cho thanh ghi r2. Tiếp theo chép buffer từ System sang User bằng System2User().

+ Bước 6: Ta xét trường hợp type = 1 và type = 0. Nếu thỏa ta sẽ lấy vị trí con trỏ hiện tại bằng GetCurrentPos() gọi là NewPos, trả về số byte đọc được cho thanh ghi r2 bằng công thức: NewPos -OldPos và cũng chép buffer từ phía System sang User bằng System2User().

+ Bước 7: Nếu file cần đọc rỗng thì ta sẽ trả về -2 cho thanh ghi r2.

* Cài đặt Write:

+ Bước 1: Ta đọc tham số buffer từ thanh ghi r4, tham số charcount từ thanh ghi r5 và id từ thanh ghi r6.

+ Bước 2: Ta kiểm tra id có nằm trong bảng mô tả file không. File cần đọc có tồn tại không và file đó có type khác 2 và type khác 1 ko.

+ Bước 3: Trường hợp vi phạm 1 trong 3 điều kiện: Chương trình báo lỗi và trả về -1 cho thanh ghi r2.

+ Bước 4: Trường hợp hợp lệ thì ta lấy vị trí con trỏ trong file bằng phương thức GetCurrentPos() trong lớp FileSystem gọi là OldPos. Sau đó chép giá trị từ thanh ghi r4 từ phía User sang System bằng User2System. Giá trị chép là buffer (chuỗi ký tự).

+ Bước 5: Ta xét trường hợp type = 0, thì ta lấy vị trí con trỏ hiện tại bằng GetCurrentPos() gọi là NewPos, trả về số byte ghi được cho thanh ghi r2 bằng công thức: NewPos -OldPos.

+ Bước 6: Ta xét trường hợp type = 3, ta sẽ gọi phương thức Write của gSynchConsole để ghi từng ký tự trong buffer và kết thúc bằng ký tự xuống dòng “\n”, trả về số byte ghi được vào thành ghi r2.

## Cài đặt các system call

1. Cài đặt system call **int ReadInt()**
   * Bước 1: lấy buffer chứa chuỗi số nguyên đọc được, để lấy được buffer này, sử dụng hàm **Read(buffer, 256)** từ class **SynchConsole** để nạp chuỗi số vào buffer đồng thời trả ra độ dài của chuỗi số
   * Bước 2: Tiến hành lặp qua tất cả phần tử trong buffer, nếu có bất kỳ phần tử nào không thuộc khoảng ‘0’ đến ‘9’ thì trả về 0, tăng programcounter, giải phóng buffer, ngược lại thì tiếp tục lặp
   * Bước 3: Chuyển buffer thành số nguyên, rồi ghi vào thanh ghi số 2 với hàm **WriteRegister** từ class **Machine**. ( vì theo mô tả của NachOS thì thanh ghi số 2 trả ra giá trị trả về của system call ).
   * Bước 4: Giải phóng **buffer** và tăng programcounter
2. Cài đặt system call **char PrintInt(int number)**

* Bước 1: lấy số nguyên từ thanh ghi số 4 ( vì theo mô tả của **NachOS** thì thanh ghi số 4 chính là tham số thứ 1 của system call).
* Bước 2: xử lý trường hợp: **INT\_MIN** và **0**, in ra màn hình
* Bước 3: lưu lại dấu của số và chuyển số thành số dương.
* Bước 4: đếm số chữ số của số nguyên, sau đó khai báo 1 mảng char để lưu chuỗi số
* Bước 5: lặp để chuyển từng ký tự số vào trong mảng
* Bước 6: nếu số là số âm thì gán phần tử đầu tiên của mảng là “-“
* Bước 7: dung hàm **Write** trong class **SynchConsole** để in ra màn hình
* Bước 8: giải phóng mảng và tăng program counter

1. Cài đặt system call **char** **ReadChar**()

* Bước 1: đọc ký tự từ buffer, lấy ký tự bằng hàm **Read(buffer, 1)** từ class **SynchConsole** để nạp chuỗi số vào buffer đồng thời trả ra độ dài của chuỗi số.
* Bước 2: nếu độ dài chuỗi số là -1 tức là chuỗi rỗng tiến hành in ra màn hinh là chuỗi rỗng, kết thúc.
* Bước 3: nếu có 1 kí tự thì tiến hành ghi vào thanh ghi số 2
* Bước 4: xóa buffer và tăng program counter

1. Cài đặt system call **void** **PrintChar(char character)**

* Bước 1: đọc giá trị từ thanh ghi số 4.
* Bước 2: in ra giá trị bằng hàm **Write** từ class **SynchConsole**.
* Bước 3: tăng program counter.

1. Cài đặt System call **void** **ReadString( char[] buffer, int length)**

* Bước 1: lấy địa chỉ chuỗi kí tự từ thanh ghi số 4, và lấy độ dài ‘length’ của chuỗi từ thanh ghi số 5
* Bước 2: chuẩn bị sẵn 1 mảng buffer có độ dài bằng độ dài của chuỗi , đọc chuỗi mà User nhập vào từ bàn phím bằng hàm **Read(buffer, length)** từ class **SynchConsole**.
* Bước 3: dùng hàm **System2User** chuyển chuỗi từ vùng nhớ System sang User
* Bước 4: giải phóng buffer và tăng program counter.

1. Cài đặt system call **void PrintString (char[] buffer)**

* Bước 1: lấy địa chỉ chuỗi kí tự từ thanh ghi số 4 gán vào virtAd
* Bước 2: chuyển chuỗi từ vùng nhớ User sang vùng nhớ System bằng hàm **User2System( virtAd, 255)** và gán vào mảng buffer đã chuẩn bị sẵn
* Bước 3: đếm độ dài của chuỗi
* Bước 4: in chuỗi ra màn hình bằng hàm **Write** từ class **SynchConsole**
* Bước 5: giải phóng buffer và tăng program counter.

# Tài Liệu Tham Khảo

1. Biên dịch và cài đặt NachOS – file được cung cấp bởi giáo viên hướng dẫn.