**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TPHCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

Đồ án 2   
Báo cáo đồ án hệ điều hành

**Lâm Khả Hân -1512316**

**Phạm Thị Thanh Hoài -1512172**

**Đào Minh Toàn - 1512581**

Nội dung

[II. Quản lý hệ thống tập tin 2](#_Toc500102516)

[1. Cài đặt system call int CreateFile(char \*name) 2](#_Toc500102517)

[2. Cài đặt system call OpenFileID Open(char \*name, int type) và int Close(OpenFileID id) 2](#_Toc500102518)

[3. Cài đặt system call int Read(char \*buffer, int charcount, OpenFileID id) và int Write(char \*buffer, int charcount, OpenFileID id) 2](#_Toc500102519)

[4. Cài đặt system call int Seek (int pos, OpenFileID id) 2](#_Toc500102520)

[5. Cài đặt các chương trình để kiểm tra các system call 2](#_Toc500102521)

[III. Đa chương, đồng bộ hóa 2](#_Toc500102522)

[1. Thay đổi mã cho các exception 2](#_Toc500102523)

[2. Cài đặt system call SpaceID Exec(char\* name) 3](#_Toc500102524)

[3. Cài đặt system calls int Join(SpaceID id) và void Exit(int exitCode) 3](#_Toc500102525)

[4. Cài đặt system call int CreateSemaphore(char\* name, int semval) 3](#_Toc500102526)

[5. Cài đặt system call int Up(char\* name), và int Down(char\* name) 3](#_Toc500102527)

[6. Cài đặt chương trình shell 3](#_Toc500102528)

**(\*) Bài làm có tham khảo các nguồn trên internet như:**

<https://github.com/kuroyakumo95/nachoshotga>

<https://github.com/dangkhoasdc/nachos>

<https://github.com/srijanshetty/nachos>

và một số nguồn khác

1. **Thông tin thành viên**

Lâm Khả Hân MSSV: 1512316

Phạm Thị Thanh Hoài MSSV: 1512172

Đào Minh Toàn MSSV: 1512581

1. **Quản lý hệ thống tập tin**
2. **Cài đặt system call int CreateFile(char \*name)**

Chuyển filename từ vùng nhớ user space tới vùng nhớ system space l sau đó dùng hàm Create đã tạo trước đó để tạo file.

1. **Cài đặt system call OpenFileID Open(char \*name, int type) và int Close(OpenFileID id)**

Mỗi tiến trình sẽ được cấp một bảng mô tả file với kích thước cố định. Kích thước của bảng mô tả file ở đây là có thể lưu được đặc tả của 10 files

Dùng type = 0 (Đọc / ghi), 1 chỉ đọc, 2 dành cho console input, 3 dành cho 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.

Còn syscall Close thì sẽ dựa vào index id rồi xóa, nếu xóa ok thì trả về 0 còn lỗi thì -1.

1. **Cài đặt system call int Read(char \*buffer, int charcount, OpenFileID id) và int Write(char \*buffer, int charcount, OpenFileID id)**

Các system call đọc và ghi vào file với id cho trước.

* Read: Nếu id = 0 thì là ReadString, nếu id = 1 thì trả về -1.
* Write: Nếu id = 1 thì là ReadString, nếu id = 0 thì trả về -1.

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.

1. **Cài đặt system call int Seek (int pos, OpenFileID id)**

Seek sẽ phải chuyển con trỏ tới vị trí thích hợp. Trả về vị trí thực sự trong file nếu thành công và -1 nếu bị lỗi.

Thêm hàm OpenFile::GetCurrentPos() để lấy vị trí hiện tại của con trỏ và OpenFile::Seek(…) cho lớp chưa có. Hàm OpenFile::Seek(…) cũng được sửa lại giá trị trả về là int cho biết vị trí sau khi dịch.

1. **Cài đặt các chương trình để kiểm tra các system call**

Hầu như chỉ cần gọi các system call trong các chương trình này. Và sửa Halt() thành Exit(0) để tiện xử lý cho phần sau.

1. **Đa chương, đồng bộ hóa**
2. **Thay đổi mã cho các exception**

Đầu tiên sửa Halt() thành Exit(0) để không gây ra việc HĐH phải shut down.

Ở mỗi exception thay Halt bằng increasePC()

Tạo biến BitMap\* gPhysPageBitMap khai báo tại syscall.h và cài đặt tại syscall.cc và chỉ thị biên dịch trong Makefile.common.

Ngoài ra còn có 1 số biến khác được khởi tạo

1. **Cài đặt system call SpaceID Exec(char\* name)**

Chuyển name kernel từ user , sau đó gọi pTab->ExecUpdate để thêm lệnh đó vào.

1. **Cài đặt system calls** int Join(SpaceID id) **và** void Exit(int exitCode)

* Join: lấy dữ liệu đầu vào là id sau đó gọi PTable::JoinUpdate(…). Join sẽ đợi và  
  block dựa trên tham số “SpaceID id” . Join trả về exit code cho tiến trình nó đã đang  
  block trong đó, -1 nếu join bị lỗi.
* Exit: lấy dữ liệu đầu vào là id sau đó gọi PTable::JoinUpdate(…). Exit trả về exit code cho tiến trình nó đã join

1. **Cài đặt system call** int CreateSemaphore(char\* name, int semval)

CreateSemaphore nhận 2 giá trị đầu vào là tên và giá trị của nó nhằm khởi tạo Sem và được STable \* semTab xử lí

1. **Cài đặt system call** int Up(char\* name), **và** int Down(char\* name)

semTab sẽ tìm trong bảng lưu trữ của nó xem Sem nào có tên trùng với giá trị đầu vào hay không để gọi phương thức Semaphore::Wait() (cho Down) và Semaphore::Signal (cho Up).

1. **Cài đặt chương trình** shell

Shell nhận một lệnh tại một thời điểm và thực thi chương trình tương ứng. dùng vòng lặp while yêu cầu người dùng nhập vào tên chương trình cần chạy sau đó kiểm tra chương trình có tồn tại ở trên máy không, nếu không thì yêu cầu nhập lại.