**HỆ ĐIỀU HÀNH**

**BÁO CÁO**

**SYSTEM CALL**

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

Bộ môn Hệ điều hành

Khoa Công nghệ thông tin

Đại học Khoa học tự nhiên TP HCM

**MỤC LỤC**

[1 Thông tin nhóm 2](#_Toc496814284)

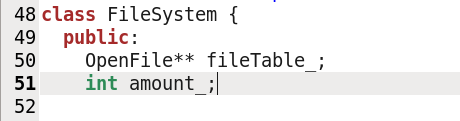
[2 Cài đặt các System Calls 3](#_Toc496814285)

[3 Các demo 7](#_Toc496814286)

# Thông tin nhóm

|  |  |  |  |
| --- | --- | --- | --- |
| **MSSV** | **Họ Tên** | **Email** | **Điện thoại** |
| 1512003 | Lê Tuấn Anh | [ltanh035@gmail.com](mailto:ltanh035@gmail.com) | 0163 819 7063 |
| 1512004 | Lê Việt Anh | [levietanh.k15.it@gmail.com](mailto:levietanh.k15.it@gmail.com) | 0168 406 4269 |
| 1512029 | Trần Quốc Bảo | [tranquocbao3897@gmail.com](mailto:tranquocbao3897@gmail.com) | 093 420 9840 |

# Cài đặt các System Calls

* + - 1. **System call SC\_CreateFile:** int CreateFile(char \*filename)
* Sử dụng hàm tạo file của hệ thống (FileSystem).
* Trả về: 0 nếu thành công, hoặc -1 nếu có lỗi xảy ra. *(Các lỗi được liệt kê bên dưới)*
* Xử lý các exception:
* Hệ thống không đủ bộ nhớ để xử lý mảng kí tự tên file.
* Không thể tạo file do không có quyền hoặc các lỗi khác (nếu có).
  + - 1. **System call SC\_Open:** OpenFileId Open(char \*name, int type)
* Xây dựng một cấu trúc để lưu các file đang được mở và có thể trả về ID của chúng. Ở đây dùng con trỏ cấp hai và đặt nó trong lớp FileSystem, bởi vì nếu đặt nó trong file exception.cc thì mỗi lần gọi syscall thì file exception.cc sẽ bị xóa và tạo lại -> làm mất bảng mô tả file.
* Để an toàn bảng mô tả file được đặt trong trường private của lớp FileSystem. Xây dựng thêm các hàm Getter – Setter cho bảng mô tả file này.
* Ở đây dùng con trỏ cấp 2 cho dễ cấp phát và thu hồi vùng nhớ - dễ dàng quản lý các con trỏ cấp 1 là OpenFile\*. Ở đây để các biến trong public cho dễ quản lý. Có viết thêm hàm *int EmptyEntryInTable()* để tìm các vị trí trống trong bảng file. Nếu trả về -1 thì bảng file đã đầy, cho nên không thể mở thêm file nữa. (Bỏ qua 2 ô đầu tiên 0, 1 trong bảng filde để dành cho Console In và Console Out)
* Đồng thời phải lưu lại được loại file đang mở là chỉ đọc hay đọc và ghi. Biến trạng thái này lưu trong đối tượng OpenFile là hợp lý nhất.
* Sử dụng hàm Open của hệ thống thêm dữ liệu trường xử cho type, đồng thời cũng cập nhật bảng file và trả ra vị trí của OpenFile\* đã mở trong bảng file. Thuộc tính type (int) được đặt trong scope private của lớp đối tượng OpenFile cho nên có xây dựng thêm các hàm Getter và Setter.
* Cập nhật các hàm của lớp đối tượng FileSystem và OpenFile cho phù hợp với bảng file và thuộc tính type.
* Trả về: 0 nếu thành công, hoặc -1 nếu có lỗi xảy ra. *(Các lỗi được liệt kê bên dưới)*
* Xử lý các exception:
* Bảng file đã đầy.
* Hệ điều hành không còn đủ bộ nhớ để nạp tên file.
* Lỗi không thể mở file. (Trả về từ FileSystem)
  + - 1. **System call SC\_Close:** int CloseFile(OpenFileId id)
* **Lưu ý:** Vì nếu sửa void Close sẽ bị lỗi và ảnh hưởng đến nhiều file khác nên ta đặt lại tên là CloseFile để có thể trả về int.
* *Về cơ bản thì CloseFile không thể đóng được OpenFileID là 0, 1 vì chúng là console. Cho nên người dùng gọi lệnh đóng hai OpenFileID đặc biệt này chương trình sẽ báo lỗi.*
* Trả về: 0 nếu thành công, hoặc -1 nếu có lỗi xảy ra. *(Các lỗi được liệt kê bên dưới)*
* Xử lý các exception:
* Người dùng truyền sai OpenFileID hoặc không hợp lệ.
* Người dùng cố tình gọi (❖) ở trên.
  + - 1. **System call SC\_Write:** int Write(char \*buffer, int charcount, OpenFileId id)
* Nếu là file thông thường có OpenFileID từ 2-9 thì ta dùng hàm Write của lớp đối tượng OpenFile để ghi. Chỉ cần truyền đúng tham số.
* Nếu OpenFileID là ConsoleOutput thì ta dùng hàm Write của lớp SynchConsole với với biến toàn cục gSynchConsole (biến này được tạo cùng cách với biến toàn cục machine hay là fileSystem).
* Trả về: số byte thực sự được đọc, hoặc -1 nếu có lỗi xảy ra. *(Các lỗi được liệt kê bên dưới)*
* Xử lý các exception:
* Người dùng truyền sai OpenFileID hoặc không hợp lệ. Bao gồm có truyền OpenFileID của ConsoleInput.
* Lỗi cấp phát chuỗi chứa kết quả đọc được.
  + - 1. **System call SC\_Read:** int Read(char \*buffer, int charcount, OpenFileId id)
* Nếu là file thông thường có OpenFileID từ 2-9 thì ta dùng hàm Read của lớp đối tượng OpenFile để đọc. Chỉ cần truyền đúng tham số.
* Nếu OpenFileID là ConsoleInput thì ta dùng hàm Read của lớp SynchConsole với với biến toàn cục gSynchConsole (biến này được tạo cùng cách với biến toàn cục machine hay là fileSystem). **Lưu ý:** Thêm kí tự NULL vào cuối chuỗi đọc được.
* Trả về: số byte thực sự được đọc, hoặc -1 nếu có lỗi xảy ra, hoặc -2 nếu đã là cuối file. *(Các lỗi được liệt kê bên dưới)*
* Xử lý các exception:
* Người dùng truyền sai OpenFileID hoặc không hợp lệ. Bao gồm có truyền OpenFileID của ConsoleOutput.
* Đã đến cuối file. Không còn dữ liệu để đọc tiếp.
* Lỗi cấp phát chuỗi nhận chuỗi từ user.
  + - 1. **System call SC\_Seek:** int Seek(int pos, OpenFileId id)
* Sử dụng hàm Seek của lớp đối tượng OpenFile. Nó được gọi thông qua con trỏ được quản lý trong bảng file nằm trong lớp đối tượng FileSystem.
* Trả về: vị trí đã seek thành công, hoặc -1 nếu có lỗi xảy ra. *(Các lỗi được liệt kê bên dưới)*
* Xử lý các exception:
* Người dùng truyền sai OpenFileID hoặc không hợp lệ.
* Người dùng yêu cầu seek trên console.
* Người dùng truyền vị trí pos cần seek không đúng. (Nhỏ hơn 0 hoặc vượt quá độ dài của file)

1. **Các system calls hỗ trợ khác:**

* SC\_Print – void print(char \*str): Xuất chuỗi str ra màn hình console.
* SC\_ScanChar – char ScanChar(): đọc một kí tự từ console.
* SC\_ScanLine – int ScanLine(char \*buf, int limit): đọc một dòng từ console.

# Các demo

Link youtube: <https://youtu.be/HZtmIbFNwzc>