HỆ ĐIỀU HÀNH

PROJECT 02

Tìm hiểu system calls - các thao tác với files

1. Quy định chung

Đồ án được làm theo nhóm: mỗi nhóm tối đa **3** sinh viên, tối thiểu **2** sinh viên.

- Các bài làm giống nhau sẽ đều bị điểm 0 toàn bộ phần thực hành (dù có điểm các bài tập, đồ án thực hành khác).

- Môi trường: máy ảo Linux (khuyến khích dùng Ubuntu 18.04), Nachos 4.0

2. Cách thức nộp bài

Nộp bài trực tiếp trên Website môn học, không chấp nhận nộp bài qua email hay hình thức khác.

Tên file: **MSSV1_MSSV2_MSSV3.zip** (Với MSSV1 < MSSV2 < MSSV3)

Ví dụ: Nhóm gồm 3 sinh viên: 2012001, 2012002 và 2012003 làm đề 1, tên file nộp: **2012001_2012002_2012003.zip**

Cấu trúc file nộp gồm:

1. **Report.pdf:** chứa báo cáo về bài làm
2. **Source**: thư mục chứa source code của Nachos (NachOS-4.0/code/)

Nộp chương trình: khi bạn hoàn thành, xóa các object file (.o) và các file thực thi bạn đã tạo.

Lưu ý: Cần thực hiện đúng các yêu cầu trên, nếu không, bài làm sẽ không được chấm.

3. Hình thức chấm bài:

Mỗi nhóm cần phải demo đồ án của nhóm mình và trả lời câu hỏi vấn đáp từ Giảng viên.
Thời gian vấn đáp: thông báo sau

4. Thang điểm chi tiết

Phần	Câu	Ghi chú	Điểm
1	1	Create	**0,75đ**
	2	Open	**0,75đ**
	3	Close	**0,75đ**
	4	Read	**0,75đ**
	5	Write	**0,75đ**
	6	Seek	**0,75đ**
	7	Remove	**0,75đ**
	8	Sao chép vùng nhớ kernel-user	**0,75đ**
2	1	createfile	**0,5đ**
	2	cat	**0,5đ**
	3	copy	**0,5đ**
	4	delete	**0,5đ**
	5	concatenate	**0,5đ**
		Không để user làm sụp hệ điều hành	**0,5đ**
		Báo cáo	**1đ**

5. Nội dung:

Phần 1. Cài đặt system call thao tác với file

1. Cài đặt system call **int Create(char *name)**. Create system call sẽ sử dụng Nachos FileSystem Object để tạo một file rỗng. Chú ý rằng 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 Create trả về 0 nếu thành công và -1 nếu có lỗi

2. Cài đặt system call **OpenFileID Open(char *name, int type)** và **int Close(OpenFileID id)** (mở và đóng file). Chương trình người dùng có thể mở 2 kiểu file, file chỉ có thể đọc (read only), và file có thể đọc và viết (read and write). Bạn xây dựng một mảng các file descriptor table với kích thước là 20 file descriptors. 2 phần tử đầu tiên 0 và 1 sẽ không được sử dụng, cho mục đích console input và console output tương ứng. System call sẽ chịu trách nhiệm chuyển đổi buffer của user space khi cần thiết và cấp pháp tương ứng dưới kernel. Sử dụng **file system** objects được cung cấp trong **filesys** folder

   Hàm system call "**Open**" sẽ trả về file descriptor id (OpenFileID là một số nguyên integer), hoặc -1 nếu lỗi. Mở file không thành công có thể có nhiều lý do như: mở một file không tồn tại hoặc nếu không còn chỗ trống trong file descriptor table. **type** với 0 là cho phép đọc viết, và 1 là chỉ cho phép đọc. Nếu type sử dụng ngoài hai giá trị (0, 1), thì system call báo lỗi. Hàm system call "**Close**" sẽ truyền vào OpenFileID và sẽ trả về -1 nếu lỗi và 0 nếu thành công

3. Cài đặt system call **int Read(char *buffer, int size, OpenFileID id)** và int **Write(char *buffer, int size, OpenFileID id)**. 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.

   - Trong trường hợp console là read và write console, sử dụng SynchConsoleInput class và SynchConsoleOutput class cho việc read và write, bạn phải đảm bảo trả đúng dữ liệu cho người dùng. Read và write vào console sẽ trả đúng số lượng ký tự được read hoặc write, không phải là charcount. Trong trường hợp read hoặc write bị lỗi, trả về -1.

4. Cài đặt system call int Seek(int position, OpenFileID id). Seek sẽ phải chuyển con trỏ tới vị trí thích hợp. position lưu vị trí cần chuyển tới, nếu pos = -1 thì di chuyển đến cuối file.

Trả về vị trí thực sự trong file nếu thành công và -1 nếu bị lỗi. Gọi Seek trên console phải báo lỗi.

5. Cài đặt system call int Remove(char *name). Remove system call sẽ sử dụng Nachos FileSystem Object để xóa file. Chú ý: cần kiểm tra file có đang mở hay không trước khi xóa.

Phần 2. Viết chương trình người dùng

1. Viết chương trình **createfile** để kiểm tra system call Create. Bạn sẽ dùng tên file cố định, hoặc cho người dùng nhập vào từ console từ ReadString

   Vd: Tạo file hello.txt

   ```
   ./nachos -x ../test/createfile hello.txt
   ```

2. Viết chương trình **cat** , yêu cầu nhập filename, rồi hiển thị nội dung của file đó

   Vd: Hiển thị nội dung file hello.txt

   ```
   ./nachos -x ../test/cat hello.txt
   ```

3. Viết chương trình **copy**, yêu cầu nhập tên file nguồn và file đích và thực hiện copy

   Vd: Copy file a.txt thành file b.txt

   ```
   ./nachos -x ../test/copy a.txt b.txt
   ```

4. Viết chương trình **delete** để kiểm tra system call Remove

   Vd: xóa file hello.c

   ```
   ./nachos -x ../test/delete hello.c
   ```

5. Viết chương trình **concatenate** để nối nội dung của 2 file, yêu cầu nhập tên file nguồn 1 và file nguồn 2

   ```
   ./nachos -x ../test/concatenate a.txt b.txt
   ```

Chú ý: không để cho user có thể làm sụp HĐH, system call nên xử lý càng nhiều trường hợp càng tốt

Ví dụ các ngoại lệ bắt buộc phải xử lý:

1. Nhập sai tên file.
2. Nhập đường dẫn không tồn tại

…

Phần 3. Viết báo cáo

Bao gồm các comments bên trong chương trình, ngắn nhưng đầy đủ, và mô tả bạn đã thiết kế và cài đặt như thế nào, tại sao làm như vậy. Vui lòng không copy mã chương trình của các bạn vào phần báo cáo. Chỉ cần giải thích các SystemCall và cách cài đặt (ý tưởng).