HỆ ĐIỀU HÀNH

PROJECT 01
Exceptions và các system calls đơn giản

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
- Link cài đặt: https://www.fit.hcmus.edu.vn/~ntquan/os/setup_nachos.html

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		Hiểu mã NachOS	**1đ**
2		Hiểu thiết kế	**1đ**
3	1	Xử lý exceptions	**0,5đ**
	2	Tăng giá trị PC	**0,5đ**
	3	ReadNum	**0,75đ**
	4	PrintNum	**0,75đ**
	5	ReadChar	**0,5đ**
	6	PrintChar	**0,5đ**
	7	RandomNum	**0,5đ**
	8	ReadString	**0,75đ**
	9	PrintString	**0,75đ**
	10	help	**0,5đ**
	11	ascii	**0,5đ**
	12	sort	**0,5đ**
		Không để user làm sụp hệ điều hành	**0,5**
4		Báo cáo	**0,5đ**

5. Nội dung:

Phần 1. Hiểu mã chương trình nachos

Việc đầu tiên là đọc để hiểu nachos. Mã hiện nay ở mức 1 người dùng bằng chương trình C tại một thời điểm. Các bạn có thể dùng chương trình **halt** để thử nghiệm nachos, chương trình halt sẽ yêu cầu hệ điều hành tắt máy (máy ảo giả lập chạy nachos). Chạy chương trình

"**./build.linux/nachos -x ./test/halt**". Dò tìm khi chương trình người dùng nạp, chạy, và gọi một system call.

Các tập tin trong đồ án này:

- **progtest.cc** kiểm tra các thủ tục để chạy chương trình người dùng
- **syscall.h** system call interface: các thủ tục ở kernel mà chương trình người dùng có thể gọi
- **exception.cc** xử lý system call và các exception khác ở mức user, ví dụ như lỗi trang, trong phần mã chúng tôi cung cấp, chỉ có 'halt' system call được viết
- **bitmap.*** các hàm xử lý cho lớp bitmap (hữu ích cho việc lưu vết các ô nhớ vật lý) **filesys.h**
- **openfile.h** định nghĩa các hàm trong hệ thống file nachos. Trong đồ án này chúng ta sử dụng lời gọi thao tác với file trực tiếp từ Linux, trong đồ án khác chúng ta sẽ triển khai hệ thống file trên ổ đĩa giả lập. (nếu kịp thời gian)
- **translate.*** Phiên bản nachos chúng tôi gửi các bạn, chúng tôi giả sử mỗi địa chỉ ảo là cũng giống hệt như địa chỉ vật lý, điều này giới hạn chúng ta chỉ chạy 1 chương trình tại một thời điểm. Các bạn có thể viết lại phần này để cho phép nhiều chương trình chạy
- cùng lúc trong đồ án sau.
- **machine.*** mô phỏng các thành phần của máy tính khi thực thi chương trình người dùng: bộ nhớ chính, thanh ghi, v.v.
- **mipssim.cc** mô phỏng tập lệnh của MIPS R2/3000 processor
- **console.*** mô phỏng thiết bị đầu cuối sử dụng UNIX files. Một thiết bị có đặc tính (i) đơn vị dữ liệu theo byte, (ii) đọc và ghi các bytes cùng một thời điểm, (iii) các bytes đến bất đồng bộ
- **synchconsole.*** nhóm hàm cho việc quản lý nhập xuất I/O theo dòng trong Nachos.
- **../test/*** Các chương trình C sẽ được biên dịch theo MIPS và chạy trong Nachos

Phần 2. Hiểu thiết kế

Để hiểu HĐH làm việc như thế nào, chúng ta cần phải hiểu và phân biệt được **kernel (system space)** và **user space**. Mỗi chương trình trong hệ thống phải có các thông tin cục bộ của nó, bao gồm program counters, registers, stack pointers, và file system handler. Mặc dù user program truy cập các thông tin cục bộ của nó, nhưng HĐH điều khiển các truy cập này, HĐH đảm bảo các yêu cầu từ user program tới kernel không làm cho HĐH sụp đổ. Việc chuyển quyền điều khiển từ user mode thành system mode được thực hiện thông qua system calls, software interrupt/trap. Trước khi gọi một lệnh trong hệ thống thì các tham số truyền vào cần được nạp vào các thanh ghi của CPU. Để chuyển một biến mang giá trị, tiến trình chỉ việc ghi giá trị vào thanh ghi. Để chuyển một biến tham chiếu, thì giá trị lưu trong thanh ghi được xem như là "user space pointer". Bởi vì user space pointer không có ý nghĩa đối với kernel, mà chúng ta cần là chuyển nội dung từ user space vào kernel sao cho ta có thể xử lý dữ liệu này. Khi trả thông tin từ system về user space, thì các giá trị phải đặt trong các thanh ghi của CPU.

Nachos cung cấp một CPU giả lập, thực tế CPU giả lập này giống hệt CPU thật (MIPS-32bit chip), nhưng chúng ta không thể chỉ thực thi chương trình như một tiến trình bình thường của UNIX, bởi vì chúng ta muốn kiểm soát có bao nhiêu lệnh được thực hiện trong một đơn vị thời

gian, không gian địa chỉ làm việc như thế nào, các interrupt và exception(system calls) được xử lý như thế nào.

Nachos cung cấp môi trường giả lập để chạy các chương trình C, xem Makefile trong thư mục test, chương trình biên dịch thông qua GCC/G++ tạo file object, sau đó chuyển sang định dạng đặc biệt của Nachos nhờ "coff2noff"

Phần 3. Exceptions và system calls

Cài đặt các xử lý cho **exceptions** và các **system calls** cơ bản cho việc nhập xuất (số nguyên, ký tự …). Chú ý các bạn không nên thay đổi mã chương trình trong thư mục **machine**, chỉ được thay đổi mã CT bên trong thư mục **userprog**

1. Viết lại file exception.cc để xử lý tất cả các exceptions được liệt kê trong *machine/machine.h*. Hầu hết các exception trong này là run-time errors, khi các exception này xảy ra thì user program không thể được phục hồi. Trường hợp đặc biệt duy nhất là *no exception* sẽ trả quyền điều khiển về HĐH, còn *syscall exceptions* sẽ được xử lý bởi các hàm chúng ta viết cho user system calls. Với tất cả exceptions khác, HĐH hiển thị ra một thông báo lỗi và Halt hệ thống.

2. Tất cả các system calls (ko phải Halt) sẽ yêu cầu Nachos tăng program counter trước khi system call trả kết quả về. Nếu không lập trình đúng phần này thì Nachos sẽ bị vòng lặp gọi thực hiện system call này mãi mãi. Cũng như các hệ thống khác, MIPS xử lý dựa trên giá trị của program counter, vì vậy bạn phải viết mã để tăng gia trị biến program counter, tìm đoạn mã này trong thư mục **machine.** Bạn phải copy mã này vào vị trí thích hợp trong phần xử lý các system call của bạn. Hiện tại, bạn phải dùng **Halt system call** tại cuối mỗi user program.

3. Cài đặt system call **int ReadNum()**. **ReadNum** system call sẽ sử dụng lớp SynchConsoleIn để đọc một số nguyên do người dùng nhập vào. Nếu giá trị người dùng nhập không phải là số nguyên thì trả về zero (0)

4. Cài đặt system call **void PrintNum(int number)**. **PrintNum** system call sẽ sử dụng lớp SynchConsoleOut để xuất một số nguyên ra màn hình

5. Cài đặt system call **char ReadChar()**. **ReadChar** system call sẽ sử dụng lớp SynchConsoleIn để đọc một ký tự do người dùng nhập vào.

6. Cài đặt system call **void PrintChar(char character)**. **PrintChar** system call sẽ sử dụng lớp SynchConsoleOut để xuất một ký tự ra màn hình

7. Cài đặt system call **int RandomNum()**. **RandomNum** system call sẽ trả về một số nguyên dương ngẫu nhiên.

8. Cài đặt đặt system call **void ReadString (char[] buffer, int length)**, Chú ý rằng thanh ghi chỉ chứa số, vì vậy khi truyền vào string sẽ là địa chỉ. Buffer là vùng nhớ thuộc userspace, khi người dùng nhập chuỗi thì nội dung được lưu trữ ở kernel space bạn cần viết một hàm tương ứng để chuyển dữ liệu từ kernelspcae qua userspace ("*kernel->machine->ReadMem()*")

9. Cài đặt system call **void PrintString (char[] buffer)**, **PrintString** system call dùng để in chuỗi ký tự trong buffer ra màn hình . Chú ý: tương tự như ReadString bạn cần phải có hàm để chuyển dữ liệu từ userspace qua kernelspace ("*kernel->synchConsoleOut->PutChar()*", không dùng *cout, printf* hoặc những hàm tương tự, để cho Nachos thực hiện*)*

Chú ý:

Chuỗi sẽ kết thúc bởi "\0".

Sau khi xử lý một system call bạn cần tăng PC lên, nếu không sẽ xảy ra hiện tượng loop (Chi tiết coi trong file HDTH: **[3]**).

10. Viết chương trình **help**, CT help dùng để in ra các dòng giới thiệu cơ bản về nhóm và mô tả vắn tắt về chương trình **sort** và **ascii**. (thật ra: chỉ việc gọi system call PrintString(char[])).

11. Viết chương trình **ascii** để in ra bảng mã ascii (bắc buộc phải in các kí tự đọc được, các mã ký tự không đọc được không cần phải in ra).

12. Viết chương trình **sort** với yêu cầu sau:
    - Cho phép người dùng nhập vào một mảng **n** số nguyên với **n** là số do người dùng nhập vào (n <=100)
    - Sử dụng thuật toán bubble sort để sắp xếp mảng trên theo chiều tăng dần hoặc giảm dần tuỳ vào người dùng lựa chọn.

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 số nguyên quá lớn, vượt quá phạm vi của số nguyên trong C.
2. Nhập ký tự thay vì nhập số nguyên

…

Phần 4. 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).