TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN ĐHQG-HCM

KHOA CÔNG NGHỆ THÔNG TIN

PROJECT 1 – EXCEPTION VÀ SYSTEM CALL

Môn Học: Hệ điều hành

Giảng viên:

Nguyễn Văn Giang

Lê Quốc Hòa

Sinh viên thực hiện:

Lê Ngọc Tường	20127383
Nguyễn Tấn Phát	20127588
Huỳnh Vĩ Khang	20127663

1. Thông tin nhóm

STT	MSSV	HỌ VÀ TÊN	ĐÁNH GIÁ
1	20127383	Lê Ngọc Tường	Tốt
2	20127588	Nguyễn Tấn Phát	Tốt
3	20127663	Huỳnh Vĩ Khang	Tốt

2. Mức độ hoàn thiện:

⇨ **100 %**

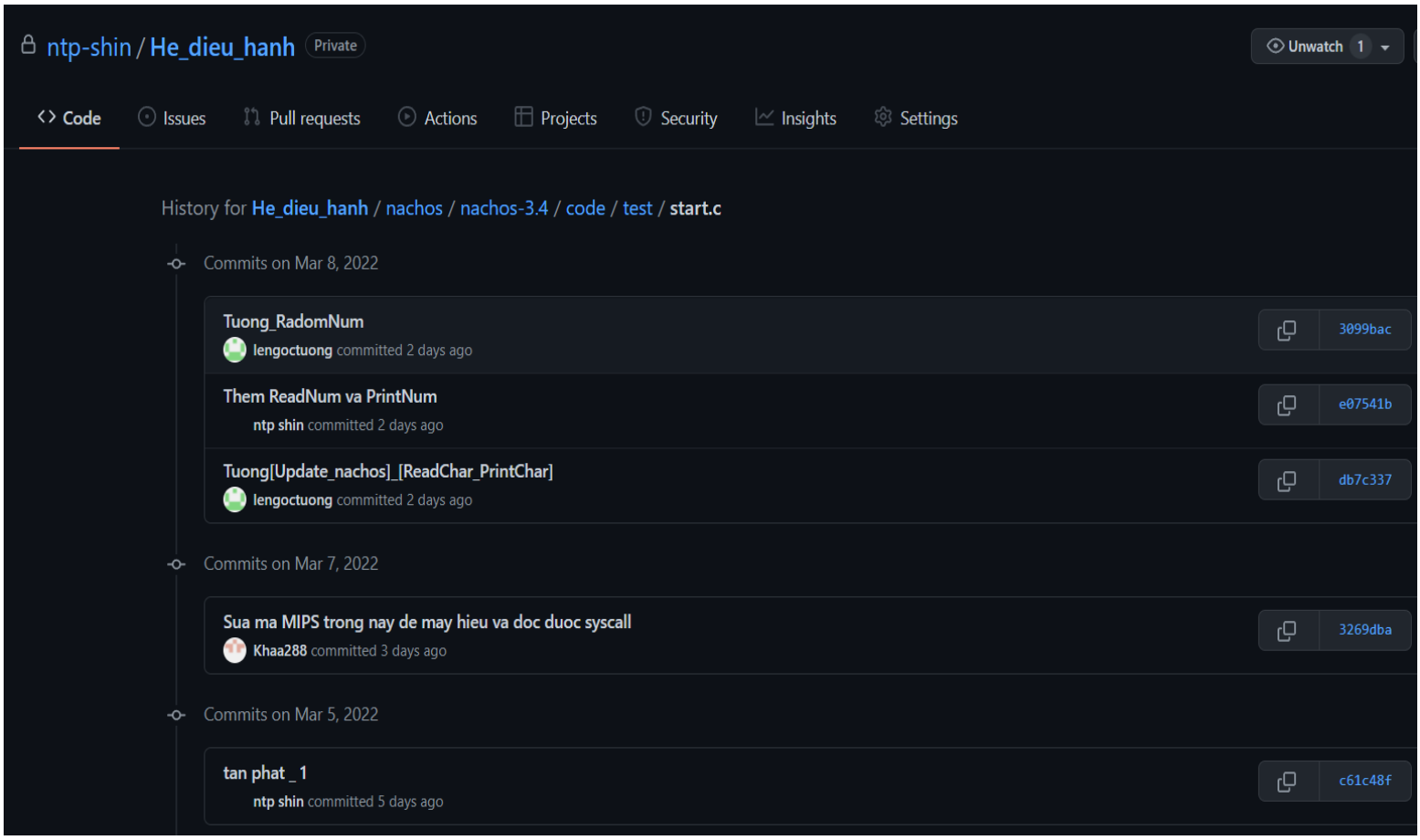
3. Môi trường làm việc:

- Vmware Workstation 12 Player.
- Github
- CentOS (Linux)

4. Bảng phân công công việc:

Hiểu mã NachOS	Đã hoàn thành	Cả nhóm
Hiểu thiết kế	Đã hoàn thành	Cả nhóm
Xý lý Exception	Đã hoàn thành	Cả nhóm
Tăng giá trị PC	Đã hoàn thành	Cả nhóm
ReadNum	Đã hoàn thành	Tấn Phát
PrintNum	Đã hoàn thành	Tấn Phát
ReadChar	Đã hoàn thành	Ngọc Tường
PrintChar	Đã hoàn thành	Ngọc Tường
RandomNum	Đã hoàn thành	Ngọc Tường
ReadString	Đã hoàn thành	Vĩ Khang
PrintString	Đã hoàn thành	Vĩ Khang

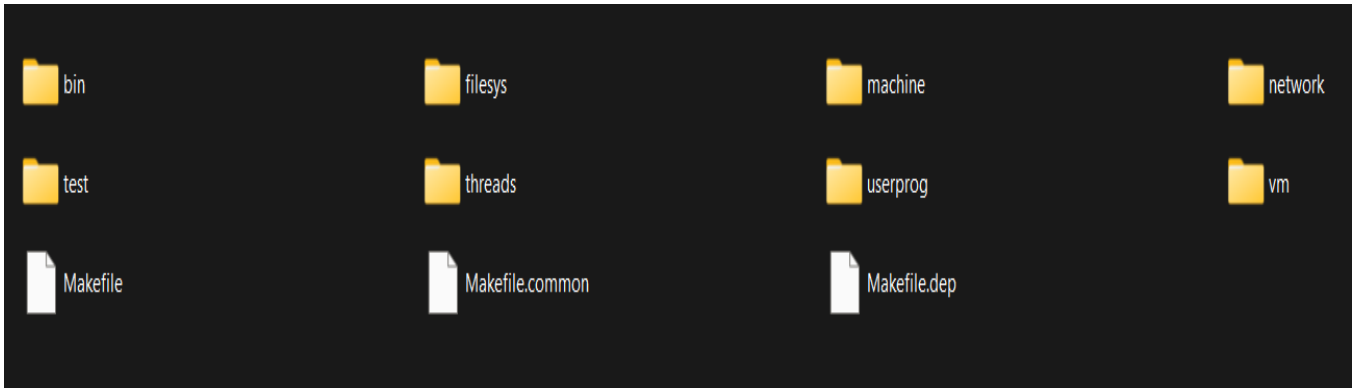
Help	Đã hoàn thành	Vĩ Khang
Ascii	Đã hoàn thành	Ngọc Tường
Sort	Đã hoàn thành	Ngọc Tường
Phân công, lên lịch họp nhóm, kiểm tra tiến độ (ở github)		Tấn Phát
Tổng hợp và viết báo cáo		Tấn Phát



5. Nội dung tìm hiểu:

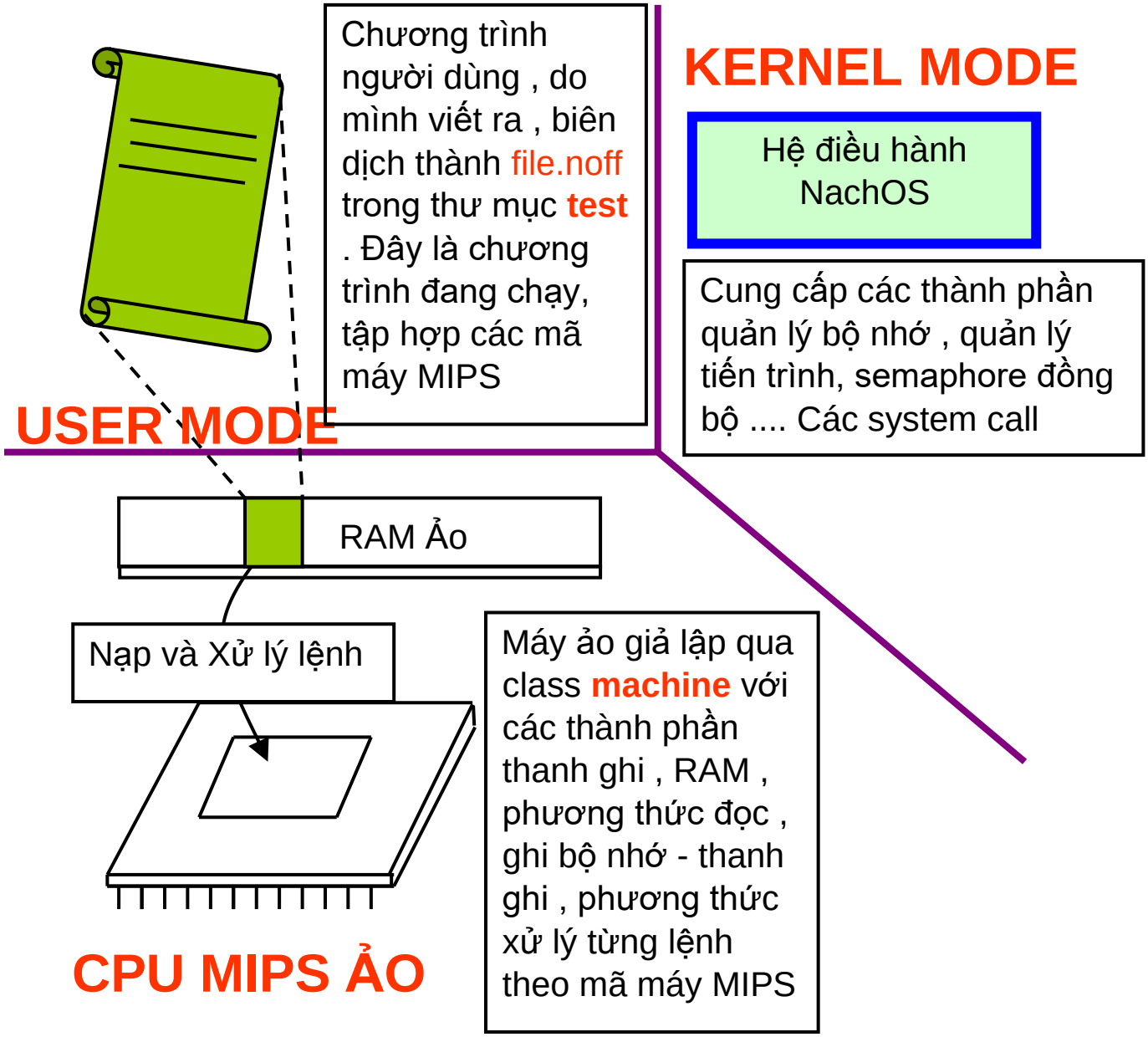
5.1 Hiểu mã chương trình NachOS:

NachOS là một phần mềm giả lập hệ điều hành chạy trên kiến trúc MIPS ở mức cơ bản, giao tiếp qua các câu lệnh chứ chưa có giao diện. NachOS sẽ các thư mục code, các mã nguồn dưới đây.



- Folder userprog: Gồm các file mã nguồn, dùng để xử lý system call. Trong đó có 2 file thường dùng khi viết system call.
  - Syscall.h: Khai báo các system call và các hàm trả về.
  - Exception.cc: Kiểm tra xem system call nào được gọi và sau đó tính toán, thực thi system call ấy.
- Folder threads: chứa các thư viện cần thiết hoặc thư viện muốn cài đặt thêm. Và trong file system.h và system.cc thực hiện khai báo, cấp phát, và xóa vùng nhớ cấp phát một biến toàn cục thuộc lớp “SynchConsole” để hỗ trợ việc nhập xuất với màn hình console.
- Folder test: Chứa các chương trình C/C++ sẽ được biên dịch theo MIPS và chạy trong NachOS.
  - Halt.c: Dùng để chạy thử, kiểm tra hàm Halt()
  - Start.c: Có thể coi là một thư viện giúp biên dịch chương trình. Halt.c => Halt.s => Halt.noff

5.2 Hiểu phần thiết kế

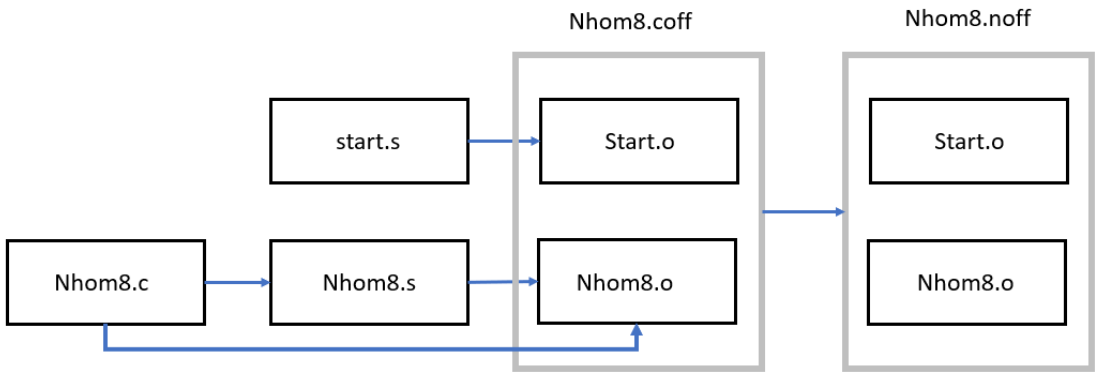


- Hệ thống NachOS sẽ gồm 3 phần chính như sau:
  - Chương trình ứng dụng(code/test): đây sẽ là nơi lập trình cũng như chạy thử các system call.
  - Cỗ máy ảo tức các tệp cấu thành nên hệ thống NachOS ví dụ như: filesys, Machine
  - Hệ điều hành NachOS, trong hệ thống này sẽ gồm 2 phần chính là USER MODE và KERNEL MODE

- Đầu tiên thì cỗ máy ảo trong NachOS đều nằm trong thư mục (code/machine) và chủ yếu nó chỉ nằm trong file machine.h
- Hệ điều hành NachOS cấu tạo từ các file mã nguồn dùng để xử lí system call và chúng được đặt trong các thư mục như threads,userprog,.. nhưng đa phần là ở userprog.
- Hệ thống này cũng chia làm 2 phần bao gồm:
    - **USER SPACE**: vùng nhớ của những chương trình chạy trên NachOS/MIPS. Khi chúng ta lập trình trên Linux, nhập dữ liệu, tất cả đều được thao tác ở userspace tức ở chương trình người dùng và chỉ duy nhất người dùng hiểu.
    - **SYSTEM SPACE**: Để cho hệ điều hành hiểu được người dùng đang làm gì thì đây chính là vùng nhớ của hệ điều hành Nachos, tất cả các lệnh trên userspace sẽ được chuyển xuống vùng nhớ hệ thống này và xử lý.

- Những file chúng ta lập trình trong thư mục code/test lúc biên dịch sẽ tạo thành file .noff đây chính là tập tin chứa toàn mã máy để hệ điều hành hiểu chúng ta đang làm những gì và sau đây là quá trình thực thi.
    1. Trong thư mục code/test sẽ có sẵn 1 file hợp ngữ start.s (đây được coi như là 1 file thư viện.
    2. Khi biên dịch thì file .c mà ta đã lập trình trước đó sẽ được biên dịch tạo thành 1 file.s tương tự như nó (vd: Nhom8.c -> Nhom8.s)
    3. File.s này sẽ được link với với start.s sẽ tạo thành 1 file tổng hợp chính là file.coff tuy nhiên đây mới chỉ là file thực thi cho Linux.
    4. Sau đó thì phần mềm coff2noff sẽ tạo thêm 1 file.noff từ file .coff ở trên, lúc này đây chính là file thực thi cho NachOS

Sơ đồ khi biên dịch ra 1 file mã nguồn vd (Nhom8.c)



5.3 Exception và system call

- **Exception.cc**: thêm các trường hợp exceptions được liệt kê ở machine/machine.h. Khi không có exception thì return. Với các exception thuộc run time error -> in ra thông tin lỗi và sau đó tắt máy. Với các syscall exception tùy loại syscall sẽ có các cách xử lí khác nhau.
- **IncreasePC()**: Tăng giá trị PC – tăng giá trị thanh ghi lên 4byte nhằm mục đích thực hiện lệnh tiếp theo.

- **ReadNum**() Tạo một mảng một chiều để lưu tất cả các kí tự được nhầm vào từ console. Kiểm tra kí tự đầu tiên có dấu '-' hay không, nếu có thì đó là số âm – được tính vào trường hợp riêng. Sau đó kiểm tra xem các kí tự còn lại trong mảng đó có hợp lệ hay không.
  - Nếu có kí tự '.' thì vẫn được tính là số
  - Nhưng ngoài ra có thêm các kí tự khác từ '0' đến '9' thì không hợp lệ.
  - Nếu hợp lệ thì đổi thành số nguyên rồi trả về.
- **PrintNum**() Đọc số từ thanh ghi R4 vào. Nếu bằng 0 thì viết lên con console "0" và kết thúc. Kiểm tra xem đó là số âm hay dương, nếu âm thì tính vào trường hợp riêng. Đếm số kí tự của số, lưu các kí tự vào mảng. Cuối cùng là in ra console.

- **ReadChar**() tạo char* buffer để lưu những gì đọc được từ Console, ta có một biến numBytes để biết số kí tự đọc được từ Console, biến này giúp xử lí các trường hợp từ console nhập nhiều hơn một kí tự hoặc nhập kí tự rỗng. Nếu gặp các trường hợp trên thì in ra thông báo và trả về thanh ghi số 2 bằng 0 (không thành công) ngược lại trả về thanh ghi số 2 mã ASCII của kí tự được nhập.
- **PrintChar**() đọc từ thanh ghi số 4 kí tự người dùng muốn in và dùng biến gSynchConsole in nó ra trên Console.

- **RandomNum**() viết thêm một syscall SC_SRandomNum để gọi srand(time(NULL)), syscall SC_RandomNum chỉ việc gọi hàm rand() và ghi giá trị nhận được vào thanh ghi số 2.

- **ReadString**() Dữ liệu được đọc từ UserSpace(địa chỉ, độ dài chuỗi) sẽ được chuyển đến vùng nhớ kernel thông qua hàm User2System để đọc dữ liệu, tuy nhiên để đọc được chuỗi này cần phải thông qua thư viện Synchcon bằng cách gọi hàm Read() để đọc chuỗi. Cuối cùng là đưa dữ liệu trở về vùng nhớ user bằng System2User.
- **PrintString**() Copy dữ liệu từ vùng nhớ user xuống kernel để máy đọc và hiểu được sau đó chỉ việc gọi hàm Write() từ thư viện Synchcon để in chuỗi ra màn hình.

- **Help**: viết một file help.c, dùng syscall PrintString để in ra các thông tin, sửa mã Makefile để compile help.c.
- **ASCII**: viết một file ascii.c, dùng syscall PrintNum, PrintChar và vòng lặp for để in ra các bảng mã ASCII từ 32 đến 126, sửa mã Makefile để compile ascii.c.
- **Bubble sort**: viết một file bubble_sort.c, dùng syscall ReadNum để đọc mảng, dùng thuật toán bubble sort để sắp xếp và dùng syscall PrintNum để in ra mảng sau khi sắp xếp, sửa mã Makefile để compile bubble_sort.c.

6. Tài liệu tham khảo:

[1] Bien dich va Cai dat Nachos.pdf

[2] Giao tiep giua HDH Nachos va chuong trinh nguoi dung.pdf

[3] Cach Viet Mot SystemCall.pdf

[4] Cach Them 1 Lop Vao Nachos.pdf

[5] https://mystudyhcmus.files.wordpress.com/2017/09/nachos_canban.doc

[6] https://github.com/nguyenthanhchungfit/Nachos-Programing-HCMUS