ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

HỌC PHẦN HỆ ĐIỀU HÀNH
(CSC10007 – 21CLC01)

BÁO CÁO ĐỒ ÁN

LẬP TRÌNH NACHOS: ĐA CHƯƠNG

GIẢNG VIÊN LÝ THUYẾT PHẠM TUẤN SƠN

GIẢNG VIÊN THỰC HÀNH LÊ VIẾT LONG

SINH VIÊN THỰC HIỆN 21127291 – NGUYỄN QUỲNH HƯƠNG
21127507 – ĐINH CÔNG HUY HOÀNG
21127681 – LÊ MỸ KHÁNH QUỲNH

MỤC LỤC

I. THÔNG TIN ĐỒ ÁN & THÀNH VIÊN

STT	MSSV	Họ và tên	Đóng góp
1	21127291	Nguyễn Quỳnh Hương	100%
2	21127507	Đinh Công Huy Hoàng	100%
3	21127681	Lê Mỹ Khánh Quỳnh	100%

II. CHI TIẾT ĐỒ ÁN

1) Cài đặt system call *Exec*

System call **Exec** có prototype ***SpaceID Exec(char* name, int priority).***

Từ thanh ghi $4, đọc vị trí bắt đầu chuỗi name (tham số đầu vào) bằng biến virtAddr. Chuyển chuỗi sang user space và gán vào biến char* name được khởi tạo. Sử dụng biến int pr để lưu giá trị priority (tham số đầu vào) được đọc từ thanh ghi $5. Thực hiện kiểm tra các điều kiện của các dữ liệu đọc được. Trong trường hợp dữ liệu không thỏa mãn điều kiện, trả về giá trị -1 cho thanh ghi $2.

Nếu dữ liệu thỏa mãn, thực hiện khởi tạo một tiến trình mới: Thread* mythread, nếu hệ thống không còn đủ bộ nhớ để tạo tiến trình thì báo lỗi, và trả về giá trị -1 cho thanh ghi $2.

Tạo biến pid, thực hiện tìm ô còn trống trong bảng trang pageTable bằng phương thức **Find** của lớp *BitMap* để lưu tiến trình. pageTable được khởi tạo với dạng dữ liệu BitMap. Giá trị trống tìm được từ pageTable tương ứng được gán cho pid của tiến trình. Sau đó thực hiện lưu tên tiến trình vào bảng fileNameTable, với index tương ứng là pid của tiến trình vừa được cấp.

Set giá trị priority đọc được cho tiến trình vừa được khởi tạo bằng phương thức **setPriority** (được cài đặt thêm trong thư mục *threads,* file *thread.h*).

Gọi lệnh **Fork** của lớp *Thread* để thực thi chương trình. Sau khi fork, tiến trình hiện tại thực hiện **Yield** để nhường CPU cho tiến trình tiếp theo dựa theo độ ưu tiên. Cuối cùng, trả về giá trị pid cho thanh ghi $2.

2) Cài đặt system call *PrintChar*

System call **PrintChar** có prototype ***void PrintChar(char c)***.

Đọc thanh ghi $4 để lấy ra ký tự cần xuất ra. Sử dụng phương thức **Write** trong lớp *SynchConsole* để thực hiện xuất ký tự vừa đọc được ra màn hình.

3) Cài đặt đa tiến trình

Dưới đây là các tập tin và thư mục đã được chỉnh sửa để hỗ trợ tính năng xử lí đa tiến trình của NachOS:

- Thư mục *threads*:
  + *system.h*: Bổ sung các biến toàn cục sau:
    1. extern BitMap *gPhysPageBitMap: quản lý các frame vật lý trên RAM
    2. extern BitMap *pageTable: quản lý các trang, cụ thể là cấp PID và giới hạn số trang cho các tiến trình
    3. extern char** fileNameTable: quản lý tên các tiến trình

  + *system.cc*: Cấp phát bộ nhớ cho các biến toàn cục vừa khai báo ở trên

- Thư mục *userprog*:
  + *addrspace.cc*: Chỉnh sửa phương thức khởi tạo **AddrSpace** và phương thức **~AddrSpace**:
    1. ***AddrSpace(OpenFile *executable)***: tương tự với hàm lúc đầu, hàm thực hiện tìm và tính toán kích thước của tiến trình được nạp vào, sau đó tìm những trang còn trống trên RAM bằng phương thức **NumClear** trong lớp *BitMap* để tìm, nếu không đủ sẽ trả về số trang là 0 và trở về. Nếu thành công, thực hiện tìm khung trang vật lí còn trống để đánh dấu vào và thực hiện copy các gói dữ liệu vào từng trang.
    2. ***~AddrSpace()***: thực hiện hủy đi các trang đã được cấp.

  + *progtest.cc*: Thêm phương thức StartProcess_2:
    void StarProcess_2(int id): thực hiện truyền giá trị id vào và tra cứu dựa trên bảng fileNameTable đã tạo ở trên để dò ra tên tiến trình muốn tạo, sau đó thực hiện khởi tạo và chạy tiến trình như hàm **StartProcess**.

  + *syscall.h*: Định nghĩa các system call **Exec**, **Exit** và **PrintChar**, cũng như các phương thức tương ứng.

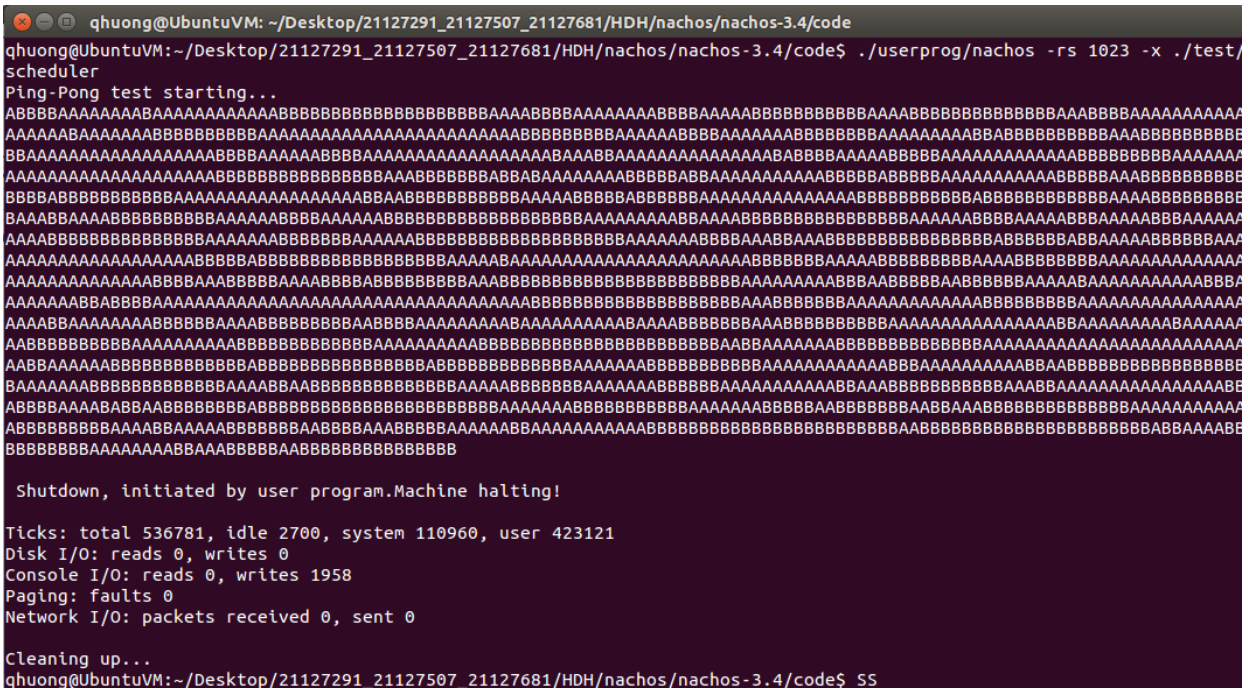
+ *exception.cc*: Cài đặt các system call SC_Exec, SC_Exit, SC_PrintChar.

- Thư mục *test*: Bổ sung các chương trình test: *ping.c*, *pong.c* và *scheduler.c*.
- Thay đổi sector size thành 512 trong *disk.h*, nhằm đảm bảo đủ số trang để lưu kích thước của chương trình.F

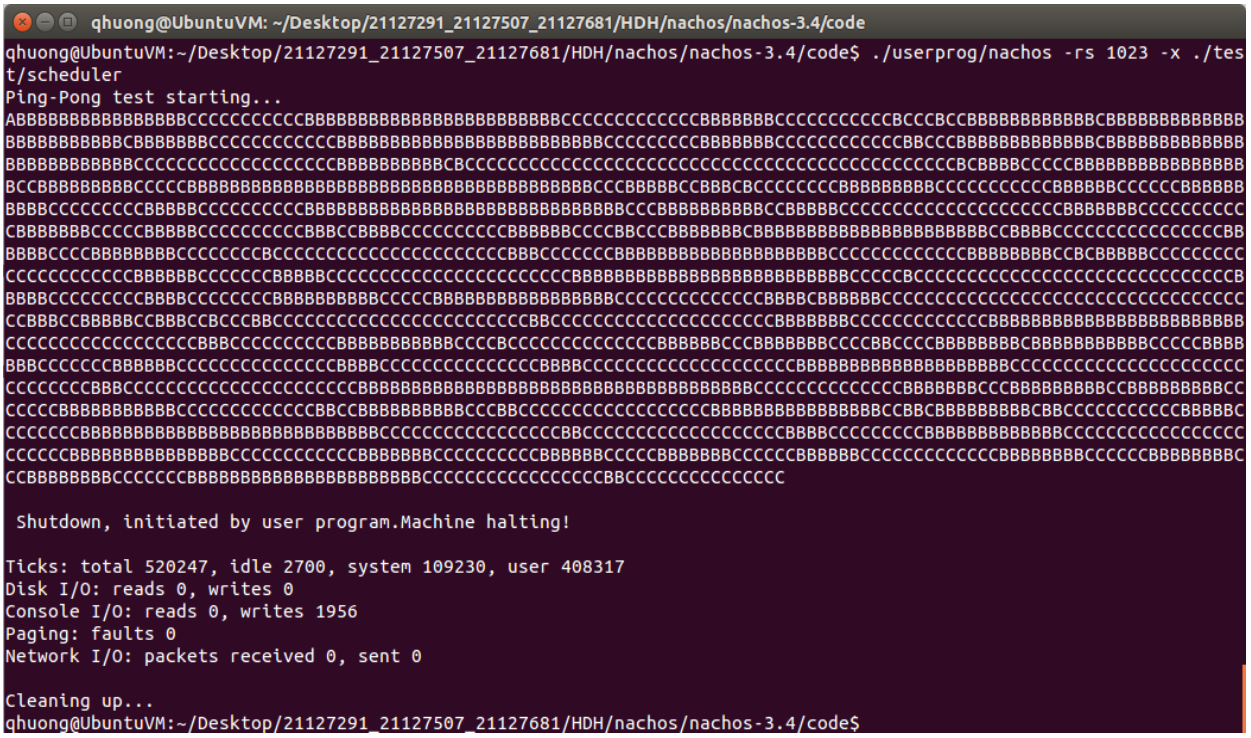
4) Mở rộng: thêm tham số độ ưu tiên cho system call Exec

- Cài đặt lại hàm **ReadyToRun**: thay đổi thứ tự khi thêm thread vào readyList, sử dụng phương thức **SortedInsert** ở lớp *List* để có thể thêm các thread theo thứ tự độ ưu tiên ta cung cấp.
- Cài đặt lại hàm **FindNextToRun**: ban đầu hàm này chỉ hoạt động theo cơ chế first-in-first-out (FIFO). Thực hiện sort lại liên tục readyList nhằm luôn lấy ra được thread có độ ưu tiên cao nhất.
- Thêm biến priority trong lớp *Thread* để lưu giá trị của độ ưu tiên của từng thread, phục vụ cho quá trình sort các thread trong list.
- Nhằm thấy rõ sự khác biệt sau khi cài đặt system call **Exec** có độ ưu tiên tiến trình, nhóm đã viết thêm một chương trình *pang.c* trong thư mục *test*. *pang.c* xuất ra màn hình 1000 kí tự "C". Trong *scheduler.c*, tiến trình ping được gán độ ưu tiên là 2, còn pong và pang được gán độ ưu tiên là 1.

III. ẢNH DEMO ĐỒ ÁN



Ảnh 1. Kết quả khi chạy chương trình scheduler. System call Exec lúc này chưa có độ ưu tiên.



Ảnh 2. Kết quả khi chạy chương trình scheduler. System call Exec lúc này đã có độ ưu tiên.

IV. NGUỒN THAM KHẢO

- Video và slide bài giảng của giảng viên – học phần CSC10007, lớp 21CLC01 (Phạm Tuấn Sơn, Lê Viết Long – Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Thành phố Hồ Chí Minh).

- Nachos-Operating-System GitHub Repository (Ming Qin, Xinyu Qian, Evan Carey, Kevin Caasi – University of California San Diego): **LINK**

- nachos GitHub Repository (Nguyễn Thị Ngân Khánh, Võ Đăng Khoa, Tạ Ngọc Duy Khoa, Đặng Minh Quân – cựu sinh viên K16, Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Thành phố Hồ Chí Minh): **LINK**