ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM

KHOA CÔNG NGHỆ THÔNG TIN

---o0o---

BÁO CÁO

PROJECT 3:

HỆ ĐIỀU HÀNH

Thành viên nhóm	**MSSV**
Đỗ Phan Tuấn Đạt	22127057
Phạm Thành Đạt	22127064
Đỗ Đình Hải	22127095
Lê Hồ Phi Hoàng	22127123
Trần Nguyễn Minh Hoàng	22127131

Lớp: 22CLC02

Môn học: Hệ điều hành

Học kỳ: 2

Năm học: 2023-2024

I. Mục lục

II. Môi trường làm việc

1. Môi trường làm việc:

Visual Studio Code trên Ubuntu

2. Ngôn ngữ:

C; C++

III. Cài đặt các class mới

1. Lớp PTable

- Được khai báo trong file ptable.h và cài đặt trong file ptable.cc
- Chức năng: dung để quản lý các tiến trình được chạy trong hệ điều hành.
- Lớp PTable bao gồm một mảng 1D có độ dài tối đa 10, thuộc kiểu con trỏ của đối tượng của lớp PCB. Mảng này dùng để lưu con trỏ đến các block tiến trình đang chạy.
- Hàm construct dùng để khởi tạo tiến trình đầu tiên, hàm destruct sẽ huỷ tất cả các tiến trình.
- Các hàm ExecUpdate, JoinUpdate, ExitUpdate dùng để xử lý logic cho các system call Exec, Join và Exit tương ứng.
- Hàm GetFreeSlot tìm ô trống trong bảng/mảng 1 chiều để lưu tiến trình mới.
- Hàm IsExists dùng để kiểm tra có tồn tại processID tham số không, có nghĩa là kiểm tra tiến trình được đưa vào có tồn tại không.
- Hàm Remove xoá process ID khỏi bảng/mảng khi tiến trình kết thúc.
- Hàm GetFileName trả về tên tiến trình.

2. Lớp PCB

- Tượng trưng cho PCB (Process Control Block): Lưu thông tin để quản lý process.
- Một số thuộc tính quan trọng:
    - int pid: Định danh của tiến trình để phân biệt các tiến trình.

- Thread* thread: Lưu tiến trình được nạp.
- int parentID: id của tiến trìn cha.
- FileName: Lưu tên của tiến trình.
- 3 thuộc tính Semaphore: Để quản lý quá trình Join, Exit và nạp chương trình.

- Phương thức:
  - Hàm constructor khởi tạo các thuộc tính được đề cập ở trên, cùng với đó là giá trị của Semaphores. Ta gán giá trị ban đầu của joinSem và exitSem là 0, còn với mutex thì là 1. Vì joinSem và exitSem sẽ được sử dụng trong các thao tác chờ (JoinWait, ExitWait) và thao tác chạy (JoinRelease, ExitRelease) cho mục đích synchronization, còn mutex được dùng để đảm bảo mutual exclusion.
  - Hàm Exec: Hàm dùng để nạp tiến trình mới:
    - Tạo 1 tiến trình mới.
    - Gán các tham số pid, processID, parentID cho thread vừa tạo.
    - Gọi hàm Fork để chạy thread mới (được setup thông qua hàm MyStartProcess) cùng lúc với thread hiện tại.
  - Hàm IncNumWait và DecNumWait quản lý số processes đang trong hàng đợi.
  - Hàm SetFileName và GetFileName được sử dụng để gán và lấy tên file từ 1 process cụ thể.

IV. Cài đặt xử lý các system call mới

1. SC_Exec

- Trước khi cài đặt system call Exec thì thêm đoạn code cài đặt Exec(char* name, int pid) ở lớp PCB và ExecUpdate(char* name) ở lớp Ptable
- Quá trình xử lý của system call Exec:
  - Đọc địa chỉ tên chương trình "name" từ thanh ghi r4.
  - Tên chương trình lúc này đang ở trong user space. Gọi hàm User2System đã được khai báo trong lớp machine để chuyển vùng nhớ user space tới vùng nhớ system space. Nếu bị lỗi thì báo "Không mở được file" và gán -1 vào thanh ghi 2.

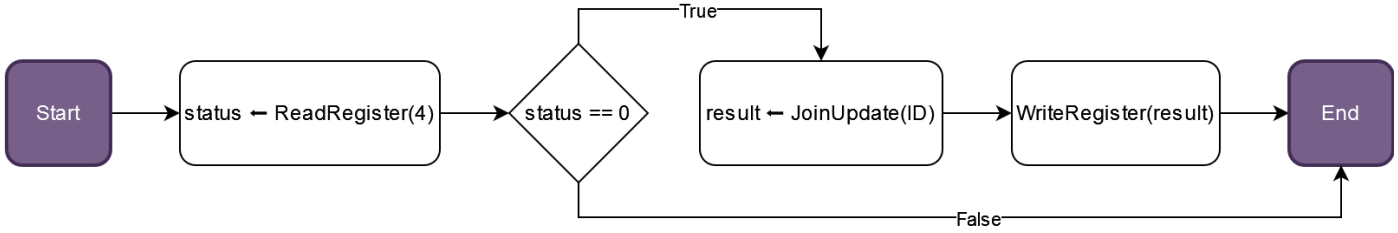
- Nếu không có lỗi thì gọi pTab-> ExecUpdate(name), trả về và lưu kết quả thực thi phương thức này vào thanh ghi r2.
- Tăng program counter

2. SC_Join

- Khai báo SC_Join trong ./userprog/syscall.h
- Cài đặt hàm JoinUpdate ở PTable
- Lưu đồ thuật toán:

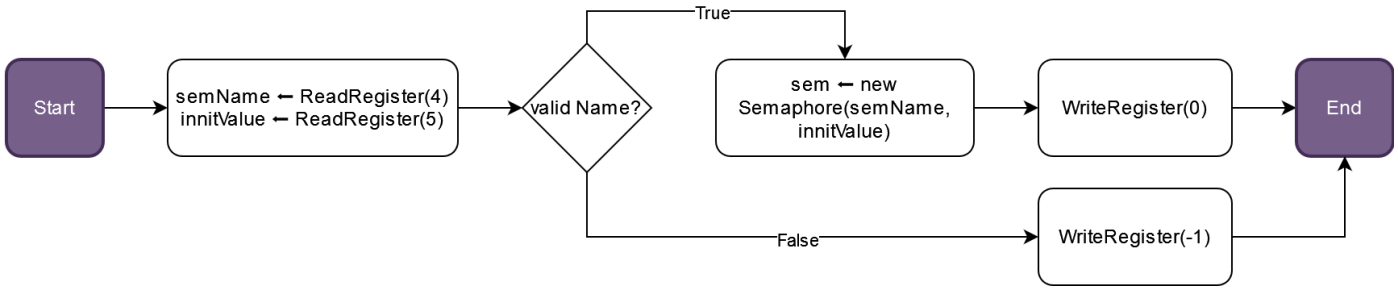
3. SC_Exit

- Khai báo SC_Exit trong ./userprog/syscall.h
- Cài đặt hàm ExitUpdate ở lớp PTable
- Lưu đồ thuật toán:



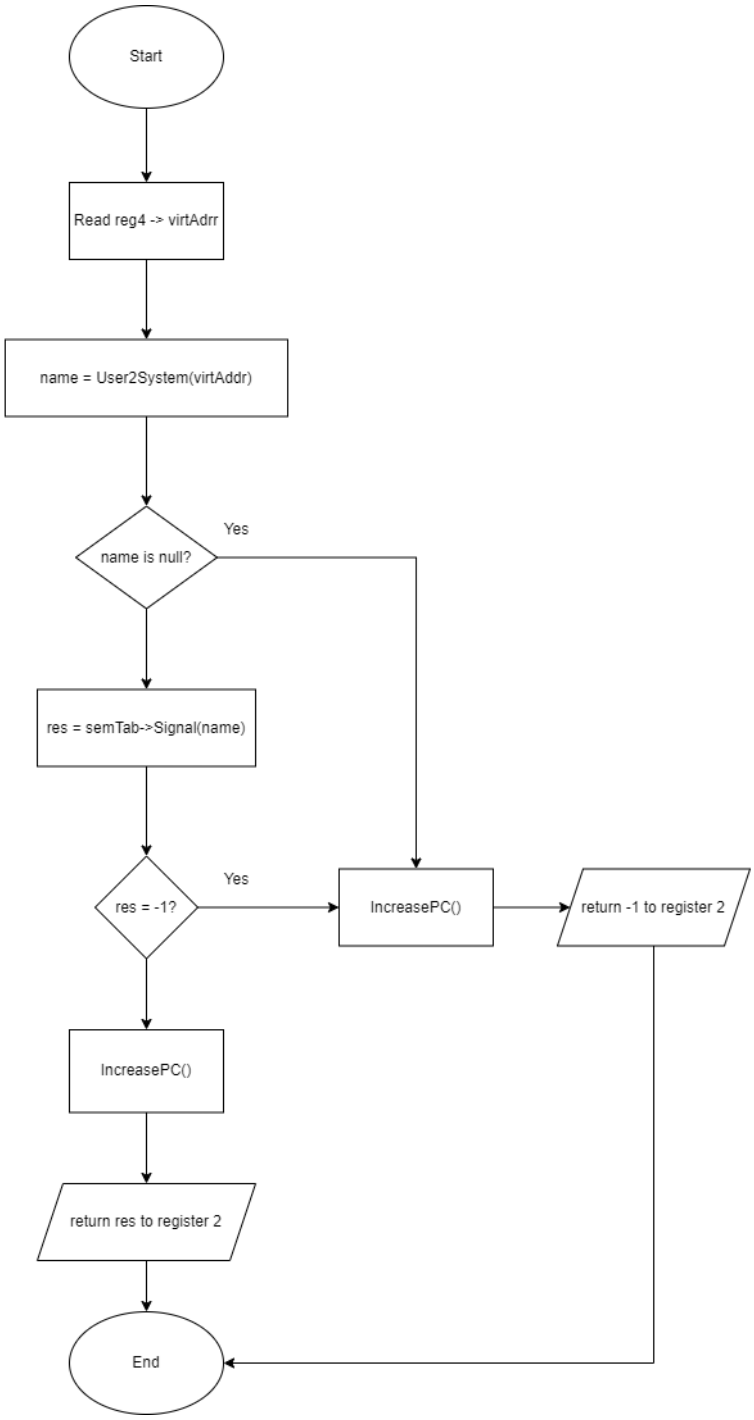
4. SC_CreateSemaphore

- Khai báo SC_CreateSemaphore trong ./userprog/syscall.h
- Lưu đồ thuật toán:



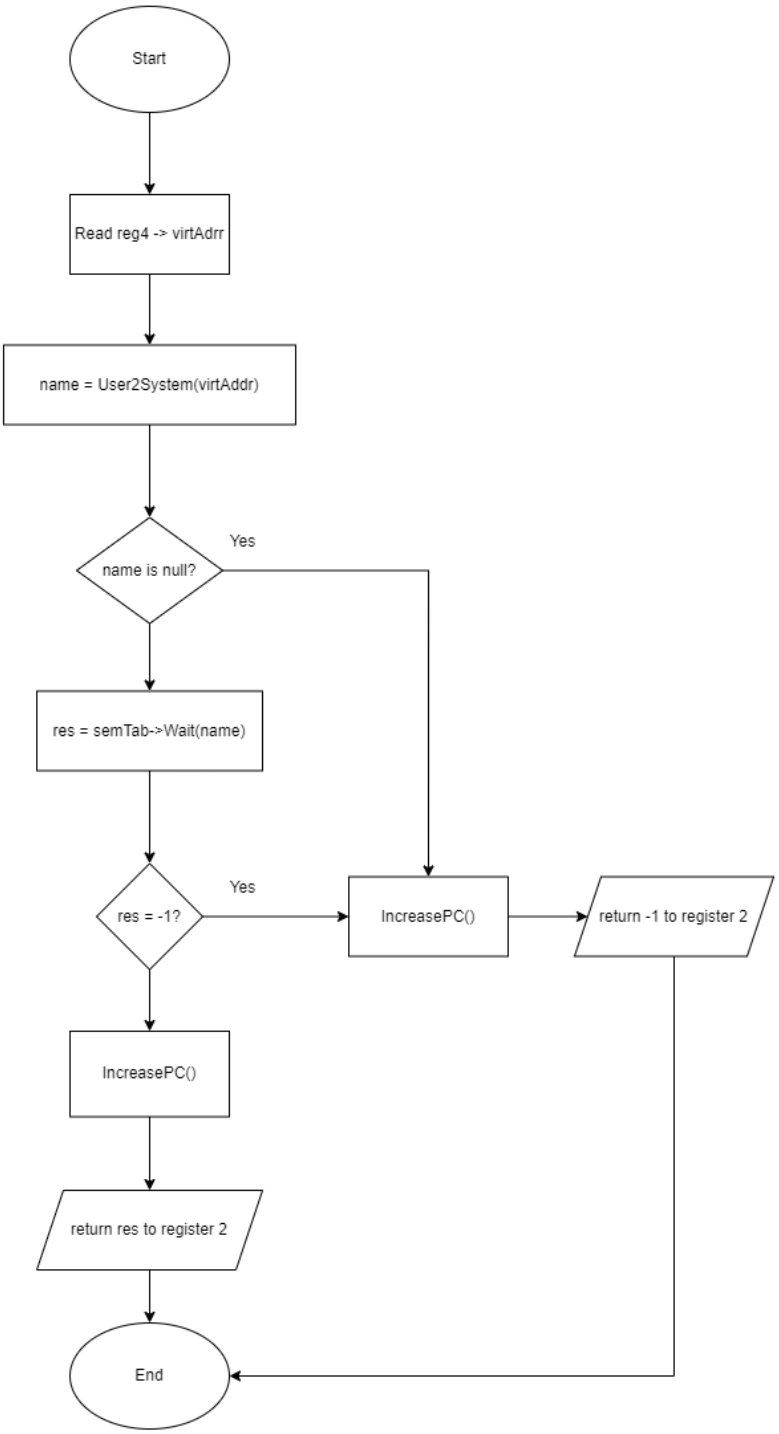
5. SC_Up

- System call sử dụng lớp Stable để giải phóng tiến trình đang chờ.
- Khai báo SC_Up ở ./userprog/syscall.h.
- Cài đặt hàm Signal(char* name, int pid) ở lớp Stable, hàm signal() ở lớp Sem.
- Lưu đồ thuật toán:



6. SC_Down

- System call sử dụng lớp Stable để giải phóng tiến trình đang chờ.
- Khi báo SC_Down ở ./userprog/syscall.h.
- Cài đặt hàm Wait(char* name, int pid) ở lớp Stable, hàm wait() ở lớp Sem.
- Lưu đồ thuật toán:



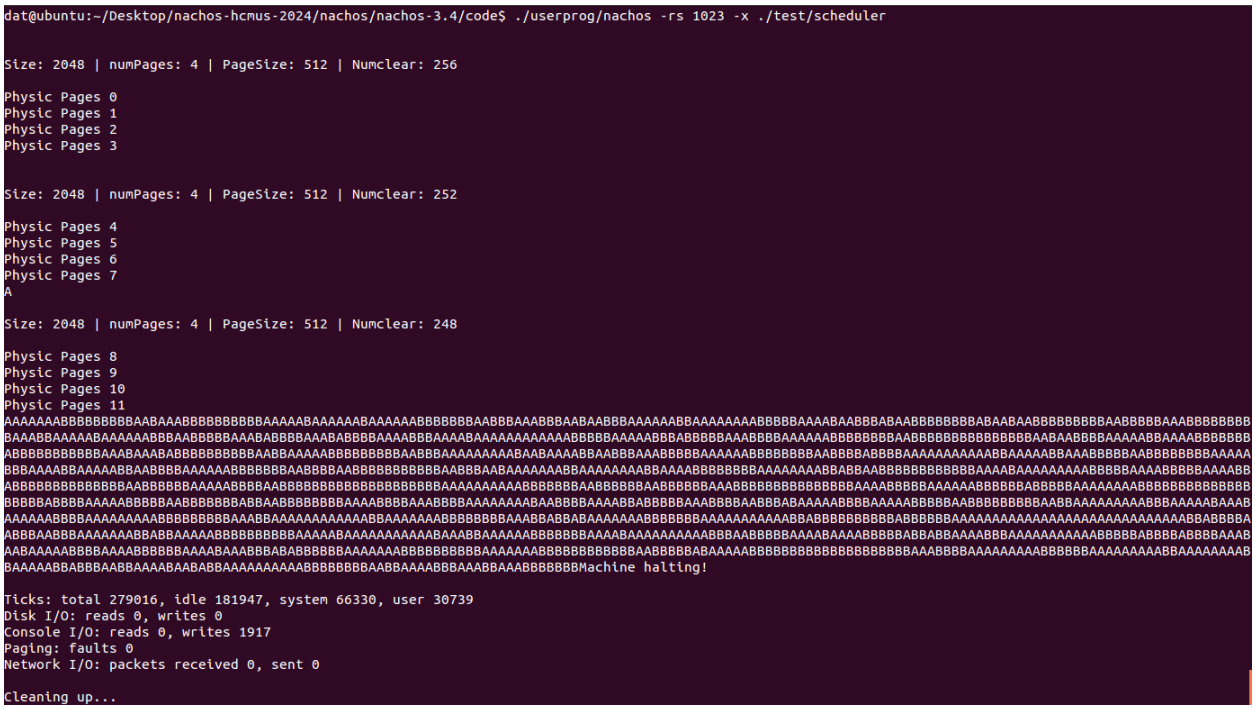
V. Chương trình minh họa scheduler

1. Phân tích mã nguồn

- Chương trình scheduler có chức năng chạy hai chương trình "Ping" (in ra 1000 chữ A) và "Pong" (in ra 1000 chữ B) song song với nhau.
- Chương trình bắt đầu bằng việc in ra thông điệp "Ping-Pong test starting...".
- Chạy chương trình "ping" và "pong": Sử dụng hàm Exec để chạy hai chương trình "ping" và "pong". Hai chương trình này được chạy bất đồng bộ (concurrently).
- Đợi kết thúc của các chương trình con: chương trình gọi hai lần hàm Join, mỗi lần với một PID khác nhau. Điều này đảm bảo rằng chương trình sẽ đợi cho đến khi cả hai chương trình "ping" và "pong" hoàn thành trước khi tiếp tục thực hiện công việc tiếp theo.
    - Hàm Join sử dụng JoinUpdate để cập nhật trạng thái đợi và trả về mã lỗi của chương trình đã kết thúc. Nếu không có lỗi xảy ra, chương trình sẽ tiếp tục thực thi.
- Hoàn thành: Sau khi cả hai chương trình "ping" và "pong" đã hoàn thành, chương trình scheduler kết thúc.

2. Demo sử dụng chương trình

- cd vào nachos/nachos-3.4/code
- Chạy dòng lệnh: ./userprog/nachos -rs 1023 -x ./test/scheduler
- Kết quả:



VI. Đóng góp

- Tổng công việc đã hoàn thành: 100% công việc được giao, chọn chương trình minh họa là Ping Pong.
- Bảng phân công nhệm vụ:

Thành viên	MSSV	Công việc	Tỉ lệ (%) hoàn thành	Tổng đóng góp cho nhóm
Trần Nguyễn Minh Hoàng	22127131	- Cài đặt lớp PCB. - Báo cáo phân tích phần tương ứng và phân tích chương trình minh họa Ping Pong.	100%	20%
Lê Hồ Phi Hoàng	22127123	- Cài đặt SC_Exec. - Viết báo cáo các phần tương ứng và demo chương trình minh họa Ping Pong.	100%	20%

Đỗ Đình Hải	22127095	- Cài đặt SC_Join và SC_Exit và lớp Ptable. - Viết báo cáo phân tích các phân tương ứng.	100%	20%
Phạm Thành Đạt	22127064	- Sửa lỗi, cập nhật lớp Ptable, cài đặt SC_Up và SC_Down và chương trình minh họa Ping Pong. - Viết báo cáo phân tích các phần tương ứng.	100%	20%
Đỗ Phan Tuấn Đạt	22127057	- Cài đặt SC_CreateSemaphore. - Viết báo cáo phân tích phần tương ứng. - Tổng hợp, thiết kế báo cáo.	100%	20%

VII. Tài liệu tham khảo

- Slide tài liệu Hệ điều hành của thây Lê Viết Long, ĐHKHTN-ĐHQG HCM
- Source code nachos 3.4 cung cấp bởi thầy Lê Viết Long
- Các hướng dẫn cho đồ án 3 cung cấp bởi thầy Lê Viết Long:
    - Cac Lop Trong Project 3.docx
    - Constructor_Cua_AddrSpace.pdf
    - HuongDan_Project3.pdf
    - Seminar_HDH_Buoi3.pptx