Báo Cáo Đồ Án Hệ Điều Hành

Project 03 – System Call Nachos

19127102 – Võ Hoàng Gia Bảo

19127406 – Ngô Huy Hoàng

19127457 – Nguyễn Tuấn Kiệt

Giáo viên hướng dẫn : Lê Viết Long

Logo

Description automatically generated

Bộ môn Hệ Điều Hành

Khoa Công nghệ thông tin

Đại học Khoa học tự nhiên TP. HCM

I. Mục lục:

1. Bảng phân công:

2. Mức độ hoàn thiện và các vấn đề:

3. Các thông số chỉnh sửa cần phải thêm vào:

4. Mô tả chi tiết:

4.1. Syscall Exec:

4.2. Syscall Join:

4.3. Exit:

5. Hướng dẫn sử dụng:

6. Tài liệu tham khảo:

1. Bảng Phân công:

|  |  |  |
| --- | --- | --- |
| MSSV | Họ và tên | Phân công |
| 19127102 | Võ Hoàng Gia Bảo | -Thiết kế hàm và lập trình |
| 19127406 | Ngô Huy Hoàng | -Lập trình và hỗ trợ báo cáo |
| 19127457 | Nguyễn Tuấn Kiệt | -Tìm hiểu hàm và viết báo cáo |

2. Mức độ hoàn thiện và các vấn đề:

|  |  |  |
| --- | --- | --- |
|  | Mức độ hoàn thiện (%) | Vấn đề chưa hoàn thiện |
| Thiết kế lớp PTABLE | 100% |  |
| Thiết kế lớp PCB | 100% |  |
| Thiết kế lớp BITMAP | 100% |  |
| Thiết kế lớp THREAD | 100% |  |

3. Các thông số chỉnh sửa cần phải thêm vào:

**\* machine.h:**

#define NumPhysPages ~~32~~ 🡪 128

**\* disk.h:**

#define SectorSize ~~128~~ 🡪 512

**\* addrspace.cc:**

-Thay pageTable[i].phisycalPage = i 🡪Hàm 1 trang trống và đánh dấu đã sử dụng: Unsigned int FindAnEmptyPageAndMark()

-Viết hàm tính số trang còn trống: int CountEmptyPage()

-Thêm đoạn code:

if (numPages > CountEmptyPage()) {

printf("\nAddrSpace:Load: not enough memory for new process..!");

numPages = 0;

delete executable;

addrLock->V();

return ;

}

4. Mô tả chi tiết:

4.1. Syscall Exec:

**\*Cài đặt Exec(char\* name, int pid) ở lớp PCB: ( pcb.cc)**

+Gọi mutex🡪P(); //để giúp tránh tình trạng nạp 2 tiến trình cùng 1 lúc.

+Kiểm tra thread đã khởi tạo thành công chưa, nếu chưa thì báo lỗi là không đủ bộ nhớ

+Đặt processID của thread này là pID

+Đặt parrentID của thread này là processID của thread gọi thực thi Exec

+Gọi thực thi Fork(StartProcess\_2,pID) => Ta cast thread thành kiểu int, sau đó khi xử ký hàm StartProcess ta cast Thread về đúng kiểu của nó.

+Trả về id.

**\*Hàm StartProcess\_2:**

+Hàm sẽ nhận tham số là id của tiểu trình cần thực thi. Và lấy tên của tiểu trình

+Cấp phát cho file một vùng chứa lệnh và dữ liệu trên ram . Nếu cấp phát không thành công in ra “ Loi:không đủ bộ nhớ để cấp phát “

+Nếu cấp phát thành công ta chuyển file thực thi vào vùng cấp phát trên Ram

+Lưu lại trạng thái của vùng cấp trên Ram bằng các hàm IntRegisters() và RestoreState()

+Cuối cùng ta gọi hàm Run() của machine. Sẽ thực thi chương trình đang nằm trên Ram

**\*Cài đặt ExecUpdate(char\* name) ở lớp Ptable: ( ptable.cc)**

+Gọi bmsem🡪P(); //để giúp tránh tình trạng nạp 2 tiến trình cùng 1 lúc.

+Kiểm tra sự tồn tại của chương trình “name” bằng cách gọi phương thức Open của lớp fileSystem.

+Kiểm tra chương trình có được gọi là chính nó không.

Tìm slot trống trong bảng Ptable.Nếu có slot trống thì khởi tạo một PCB mới với processID chính là index của slot này, parrentID là processID của currentThread.

+bm->Mark(ID) //Đánh dấu đã sử dụng.

+Gọi thực thi phương thức Exec của lớp PCB. Gọi bmsem->V().

+Trả về kết quả thực thi của PCB->Exec.

**\*Quá trình xử lý của system call Exec: (file exception.cc)**

+Đọc địa chỉ tên chương trình 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.



4.2 Syscall Join:

**\*Cài đặt JoinWait() ở lớp PCB:**

+Gọi joinsem🡪P() để tiến trình chuyển sang trạng thái block và ngừng lại, chờ JoinRelease để thực hiện tiếp.

**\*Cài đặt ExitRelease() ở lớp PCB:**

+Gọi exitsem🡪V() để giải phóng tiến trình đang chờ.

**\*Cài đặt JoinUpdate(int id) ở lớp Ptable.:**

+Kiểm tra processID pID có hợp lệ và tồn tại không .Nếu không thỏa, ta báo lỗi và trả về-1.

+ Kiểm tra tiến trình đang chạy có là cha của tiến trình cần Join hay không .Nếu không phải, ta báo lỗi và trả về -1.

+Tăng numwait bằng hàm IncNumWait() và gọi JoinWait() để chờ tiến trình con thực hiện xong. Tiến trình đã được giải phóng

+Xử lý exitcode. ExitRelease() để cho phép tiến trình con kết thúc.

**\*Quá trình xử lý của system call Join:**

+Đọc id của tiến trình cần Join từ thanh ghi r4.

+Gọi thực hiện pTab->JoinUpdate(id) và lưu kết quả thực hiện của hàm vào thanh ghi r2



4.3. Exit:

**\*Cài đặt JoinRelease() ở lớp PCB:**

-Gọi joinsem->V() để giải phóng tiến trình gọi JoinWait().

**\*Cài đặt ExitWait () ở lớp PCB:**

-Gọi exitsem-->V() để tiến trình chuyển sang trạng thái block và ngừng lại, chờ ExitReleaseđể thực hiện tiếp.

**\*Cài đặt ExitUpdate(int exitcode) ở lớp Ptable:**

-Kiểm tra pID có tồn tại hay không. Nếu không tòn tại ta báo lỗi và trả ra tiến trình không tồn tại.

-Nếu tiến trình gọi là main process thì gọi Halt().

-Gọi SetExitCode để đặt exitcode cho tiến trình gọi.

-Giảm numwait bằng hàm DecNumWait() và gọi JoinRelease để giải phóng tiến trình cha đang đợi nó(nếu có) và ExitWait() để xin tiến trình cha cho phép thoát.

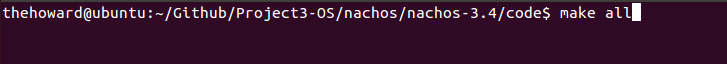
-Remove pID.

Diagram

Description automatically generated

5. Hướng dẫn sử dụng:

\*Chạy biên dịch chương trình (trước khi chạy các chương trình):



\*Chạy Chương trình Ping bằng lệnh: ./userprog/nachos -rs 1023 -x ./test/ping Text

Description automatically generated

\*Chạy Chương trình Pong bằng lệnh: ./userprog/nachos -rs 1023 -x ./test/pong

Text

Description automatically generated

\*Chạy Chương trình Scheduler bằng lệnh: ./userprog/nachos -rs 1023 -x ./test/scheduler

Text

Description automatically generated

6. Tài liệu tham khảo:

<https://spinovel.wixsite.com/codeerrorkiller/single-post/2016/04/21/project-nachos-h%C4%91h?fbclid=IwAR0lpjruC02pDOLceLrcgNEiRfZ9AiPzSMcLWdx5MhNG7wLwdXTwTZtL7V8>