ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA: CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

Chủ đề: Page Table

Môn học: Hệ điều hành

Giảng viên hướng dẫn: Lê Viết Long

Lớp: 22_1

Họ và tên	***MSSV***
Nguyễn Văn Hiến	22120101
Nguyễn Anh Trí	22120382
Trần Anh Tú	22120401

TP.Hồ Chí Minh, 11/2024

Mục lục

1 Giới thiệu

Trong hệ điều hành Xv6, page table đóng vai trò quan trọng trong việc quản lý bộ nhớ, giúp ánh xạ giữa địa chỉ logic và địa chỉ vật lý. Thông qua việc nghiên cứu và triển khai các chức năng liên quan đến page table, chúng em hiểu rõ hơn về cách hoạt động của cơ chế phân trang, quản lý bộ nhớ ảo, và tổ chức dữ liệu hiệu quả. Báo cáo này sẽ trình bày chi tiết cách chúng em giải quyết các bài toán liên quan đến page table trong Xv6.

2 Phân công công việc

Bài tập	Họ và tên	Mức độ hoàn thành
In bảng trang	Trần Anh Tú	100%
Tăng tốc system call	Nguyễn Anh Trí	100%
Tổng hợp báo cáo	Nguyễn Văn Hiến	100%
Phát hiện các trang nào đã được truy cập	Nguyễn Văn Hiến	100%

Bảng 1 – Bảng phân công công việc

Do các thành viên được chia ra để giải quyết từng bài nhỏ nên từng thành viên sẽ tự viết nội dung báo cho bài mà bản thân đảm nhiệm.

3 Các yêu cầu

3.1 Tăng tốc system call

3.1.1 Mục tiêu

- Tăng tốc lệnh gọi hệ thống bằng cách chia sẻ dữ liệu trong một vùng chỉ đọc giữa user space và kernel, tối ưu hóa lệnh gọi hệ thống getpid() và từ đó học được cách chèn ánh xạ vào bảng trang. Mục tiêu cuối cùng: bài kiểm tra ugetpid vượt qua khi chạy pgtbltest.

3.1.2 Thực hiện

- **Bước 1:** Trong kernel/memlayout.h đã được map struct usyscall với USYSCALL. Ta sẽ thêm con trỏ struct usyscall trong kernel/proc.h để lưu địa chỉ của trang được chia sẻ.
- **Bước 2:** Trong kernel/proc.c, ở hàm proc_pagetable(), ta thêm vào đoạn mã để map vùng nhớ USYSCALL vào bảng trang ngay sau đoạn mã map 2 bảng trang TRAMPOLINE và TRAPFRAME.
- **Bước 3:** Cũng trong proc.c, ở hàm allocproc(), ta sẽ thêm đoạn mã để cấp phát trang bộ nhớ cho usyscall. Nếu thất bại, nó sẽ giải phóng tài nguyên và trả về 0, còn nếu thành công thì process ID của tiến trình sẽ được gán cho usyscall.
- **Bước 4:** Ở trong hàm freeproc(), ta thêm vào đoạn code để giải phóng usyscall khi tiến trình không cần nữa. Ở hàm proc_freepagetable(), ta thêm vào dòng code đê gỡ ánh xạ trang USYSCALL, giải phóng toàn bộ bảng trang, tránh rò rỉ bộ nhớ

3.1.3 Khó khăn gặp phải

- Khi code trên VSCode xuất hiện những đoạn báo lỗi cho ánh xạ USYSCALL và struct usyscall khiến em gặp khó khăn trong việc xử lý. Cùng với đó, việc thêm code vào không đúng chỗ trong hàm cũng gây ra một số lỗi khó khăn cần xử lý (như việc free page cũ).

3.2 In bảng trang

3.2.1 Mục tiêu

- Hiểu và hiển thị cấu trúc cây bảng trang trong hệ thống RISC-V.
- Hỗ trợ gỡ lỗi ánh xạ địa chỉ ảo sang địa chỉ vật lý của tiến trình đầu tiên (init).
- Đáp ứng yêu cầu kiểm tra tự động của 3 bài tập với lệnh make grade.

3.2.2 Thực hiện

- Hàm ptePrint() (đệ quy duyệt bảng trang):
  - Cách thực hiện:
    * Duyệt qua các mục PTE từ 0 đến 511.
    * Kiểm tra tính hợp lệ (PTE_V) để chỉ in các PTE có ánh xạ hợp lệ.
    * In thông tin chỉ số PTE, giá trị PTE và địa chỉ vật lý (PTE2PA).

    - Nếu PTE không chứa các bit quyền (RWX), tức là nó trỏ đến bảng trang con, sử dụng đệ quy để duyệt cấp tiếp theo.
  - Mục đích:
    - Bảng trang có cấu trúc cây nhiều cấp, nên đệ quy là cách tiếp cận tự nhiên để xử lý tất cả các cấp.
    - Kiểm tra tính hợp lệ (PTE_V) giúp bỏ qua các mục không cần thiết, tập trung vào các ánh xạ thực tế.
    - Sử dụng đệ quy cho bảng trang con đảm bảo mã gọn gàng và dễ mở rộng.

- Hàm vmprint():
  - Cách thực hiện:
    - In địa chỉ bảng trang gốc.
    - Gọi hàm ptePrint() với level = 2 để bắt đầu từ cấp cao nhất.
  - Mục đích:
    - Hàm vmprint() đóng vai trò là điểm bắt đầu, giúp phân chia rõ ràng logic in bảng trang và logic xử lý bảng trang con.
    - In địa chỉ bảng trang gốc giúp gỡ lỗi dễ dàng hơn, xác định chính xác bảng trang nào đang được xử lý.

- Chèn vào exec.c:
  - Cách thực hiện:
    - Kiểm tra tiến trình pid == 1 (tiến trình init).
    - Gọi vmprint(p->pagetable) trước lệnh return argc để in bảng trang khi tiến trình hoàn thành exec().
  - Mục đích:
    - Tiến trình init là tiến trình đầu tiên được tạo trong hệ thống, nên bảng trang của nó cần được kiểm tra.
    - Giới hạn in cho pid == 1 giúp tránh in quá nhiều thông tin từ các tiến trình khác, giảm nhiễu trong kết quả đầu ra.

3.3 Phát hiện các trang nào đã được truy cập

3.3.1 Mục tiêu

- Theo dõi các trang bộ nhớ mà tiến trình đã truy cập, qua đó xác định trang nào được sử dụng thường xuyên.
- Dựa trên thông tin về các trang được truy cập, hệ điều hành có thể áp dụng các chiến lược như thu hồi các trang ít được sử dụng (LRU) hoặc tối ưu hóa phân bổ bộ nhớ.
- Giúp nhà phát triển hiểu được hoạt động của tiến trình liên quan đến bộ nhớ, hỗ trợ gỡ lỗi hoặc kiểm tra hiệu suất của hệ thống quản lý bộ nhớ.

3.3.2 Thực hiện

- Định nghĩa PTE_A (bit truy cập) trong kernel/riscv.h.
  - Mục đích: để đánh dấu bit Accessed trong Page Table Entry (PTE), giúp theo dõi trang bộ nhớ đã được truy cập.
- Hàm pgaccess():
  - Cách thực hiện:
    * Sử dụng các hàm argaddr() và argint() để lấy địa chỉ và số lượng trang từ không gian người dùng.
    * Kiểm tra và khởi tạo:
      · Khởi tạo mask bằng 0, biến này sẽ lưu kết quả kiểm tra các trang bộ nhớ.
      · Nếu số lượng trang cần kiểm tra (page_count) lớn hơn giới hạn số lượng trang quét, trả về lỗi.
    * Duyệt qua các trang bộ nhớ:
      · Duyệt qua từng trang từ address đến address + (page_count-1) * PGSIZE.
      · Lấy Page Table Entry (PTE) của mỗi trang và kiểm tra xem bit Accessed (PTE_A) có được thiết lập không.
    * Cập nhật kết quả:
      · Nếu trang đã được truy cập (bit PTE_A được thiết lập), đánh dấu trang đó trong mask bằng cách thiết lập bit tương ứng.
      · Xóa bit Accessed (PTE_A) trong PTE của trang để chuẩn bị cho lần kiểm tra sau.
    * Trả kết quả:
      · Sao chép giá trị result_mask (kết quả kiểm tra trang) từ bộ nhớ kernel vào địa chỉ người dùng (result_addr).
      · Nếu sao chép thành công, trả về 0; nếu không, trả về lỗi (-1).
  - Mục đích: kiểm tra các trang bộ nhớ đã được truy cập và trả về một bản đồ các trang đó trong một biến result_mask. Mỗi bit trong result_mask sẽ có giá trị 1 nếu trang bộ nhớ tương ứng đã được truy cập và 0 nếu chưa được truy cập. Sau đó, hàm này sẽ sao chép kết quả trở lại không gian người dùng.

Tài liệu

[1] Lab: page tables, MIT

[2] Project 03 Page table, GV: Lê Viết Long

[3] HƯỚNG DẪN PROJECT 03 PAGE TABLE, GV: Lê Viết Long

[4] Lab pgtbl: page tables, Ahri's Blog

[5] amirR01/xv6-improvments