**Hướng dẫn project 3**

1. **Cài đặt xv6-labs-2023**

* Cài đặt các công cụ cần thiết:

sudo apt-get install git build-essential gdb-multiarch qemu-system-misc gcc-riscv64-linux-gnubinutils-riscv64-linux-gnu

* Tải xv6-labs-2023

git clone git://g.csail.mit.edu/xv6-labs-2023

* Bắt đầu lab pagetable

git fetch

git checkout pgtbl

make clean

1. **Tăng tốc system call**

* Để tối ưu hóa lệnh gọi hệ thống getpid() trong hệ điều hành **xv6** bằng cách sử dụng vùng bộ nhớ chia sẻ giữa không gian người dùng và kernel, Ta cần map struct usyscall với USYSCALL nằm ở trong kernel/memlayout.h
* trong struct proc ở kernel/proc.h ta thêm con trỏ struct usyscall để lưu địa chỉ của cái page được share.
* Thực hiện map này được thực hiện thông qua hàm proc\_pagetable() ở kernel/pro.c proc\_pagetable() ở kernel/pro.c
* Tiếp theo, ta khởi tạo page và lưu PID của tiến trình hiện tại, cũng như free page cũ.

1. **In Bảng Trang**

* Trong kernel/vm.c đọc hàm freewalk() để hiểu cách thức duyệt qua các trang, dựa theo đó viết hàm vmprint() để in các thông tin của một trang (PTE - Page Table Entry)
* Định nghĩa prototype cho vmprint() trong kernel/defs.h để có thể gọi hàm này từ exec.c.
* Gọi hàm vmprint tại kernel/exec.c để in pagetable của tiến trình Init

1. **Phát hiện trang đang truy cập**

yêu cầu cài đặt syscall **pgaccess** giúp xác định những trang đã được truy cập. Nhận vào các tham số lần lượt: địa chỉ bắt đầu, số lượng trang cần kiểm tra, và địa chỉ bộ đệm để lưu kết quả.

* **Định nghĩa access bit** Trong kernel/riscv có giá trị 26 (64). Bit thứ 6 trong một PTE được đặt để chỉ ra rằng trang đã được truy cập.
* Viết hàm sys\_pgaccess () trong kernel / sysproc .c
* Gọi hàm walk() để duyệt qua các pte.
* Với mỗi pte trả về, kiểm tra flag PTE\_A có đang bật hay không và xóa nó sau khi đã
* duyệt qua.
* Bật bit i tương ứng ở trong mask.
* Dùng copyout() để trả về kết quả cho user
* Chạy chương trình pgtbltest để kiểm tra kết quả