CSC10007 – HỆ ĐIỀU HÀNH

BÀI TẬP THỰC HÀNH LẦN 2

I. Thông tin

STT	MSSV	Họ và tên
Thành viên 1	22120060	Trương Tiến Đạt
Thành viên 2	22120128	Bùi Quốc Huy
Thành viên 3	22120458	Quách Hải Đăng

II. Báo cáo

1. Trace

- **Kernel Space:**
  - Ta cập nhật số hiệu syscall là 22 cho syscall trace trong syscall.h.
  - Ta thêm trace_mask vào process trong proc.h để xác định xem những syscall nào sẽ được trace.
  - Ta thêm hàm sys_trace vào trong sysproc.c, nhằm mục đích copy trace_mask vào trong myproc().
- **User Space:**
  - Ta khai báo $U/_trace vào UPROGS trong Makefile.
  - Ta định nghĩa trace trong user.h và thêm entry point trong usys.pl.
  - Trong proc.c, ta cập nhật hàm fork để đảm bảo trace sau khi bật theo dõi có thể tiếp tục từ tiến trình cha sang tiến trình con phân nhánh.
  - Cập nhật hàm syscall trong syscall.c để in ra kết quả trace lên terminal, trong đó:
    - Khai báo sys_trace và thêm sys_trace vào syscall_array.
    - Mảng syscall_names là một mảng tên syscall để lập chỉ mục vào.
    - Điều chỉnh hàm syscall để in kết quả, trong đó, điều kiện “p->trace_mask & (1 << num)” kiểm tra liệu syscall tương ứng với num hiện tại có bit được bật lên hay không.
  - Trong chương trình phần mềm trace.c
    - Điều kiện 1 kiểm tra người dùng đã cung cấp đầy đủ arguments và trace mask có phải dưới dạng chữ số hay không (kiểm tra phần tử đầu tiên).
    - Điều kiện 2 thử trace syscall từ trace_mask

- Vòng lặp cuối cùng nối tiếp việc thực hiện câu lệch qua mảng nargv.

- Khó khăn: xv6 sử dụng xen kẽ kiểu dữ liệu uint64 và int ở một vài bước khác nhau nên đôi khi gặp lỗi kiểu dữ liệu.

2. Sysinfo

- Ở **user space**: Để chương trình này có thể gọi được thì đầu tiên ta phải thêm prototype của sysinfo cũng như khai báo struct sysinfo vào trong file user/user.h và user/usys.pl.
- Ở **kernel space**: định nghĩa lệnh gọi hệ thống là 23. Thêm thông tin hàm sys_sysinfo vào trong syscall.

Tiếp theo, tiến hành cài đặt hàm sys_info:

Tham số address là địa chỉ tới vùng nhớ của trong user space người dùng truyền vào. Tạo 1 biến info với kiểu dữ liệu sysinfo và gán giá trị các trường của nó bằng việc gọi các hàm get_freemem() và get_nproc. Trong đó get_freemem và get_nproc được định nghĩa trong kalloc.c và proc.c: Với get_freemem dùng 1 con trỏ kiểu run để thực hiện đến số page trống (xem bộ nhớ có trống không) và get_nproc dùng con trỏ proc tìm các tiến trình đang không ở trạng thái UNUSED.

Sau đó ta thực hiện copyout(p->pagetable, addr, (char *)&info, sizeof(info)) < 0 để đẩy dữ liệu từ kernel space về user space ở đúng địa chỉ mà người dùng đang muốn lấy dữ liệu.

Cuối cùng thực hiện kiểm tra bằng hàm syscalltest.c trong userspace như sau:

Kiểm tra bộ nhớ tự do (sử dụng hàm countfree): Bằng cách sử dụng hàm sbrk() để cấp phát và giải phóng bộ nhớ ta có thể đo lường sự thay đổi trong bộ nhớ tự do và so sánh với thông tin mà syscall sysinfo trả về. Điều này giúp xác định liệu syscall có chính xác khi báo cáo về lượng bộ nhớ còn trống hay không.

Kiểm tra số lượng tiến trình (testproc): Syscall sysinfo cung cấp số lượng tiến trình đang chạy trong hệ thống. Một trong những bài kiểm tra là tạo một tiến trình con bằng cách sử dụng hàm fork() và sau đó kiểm tra xem số lượng tiến trình đã thay đổi đúng hay chưa.

Kiểm tra hành vi khi đối số không hợp lệ (testcall): Để đảm bảo syscall xử lý lỗi đúng cách kiểm tra trường hợp khi đối số truyền vào syscall không hợp lệ. Nếu đối số không hợp lệ, syscall cần trả về lỗi và không làm gián đoạn hoạt động của hệ thống.

Khó khăn: trong sử dụng kiểu dữ liệu unit64 và int, thiếu 1 số cú pháp khai báo cấu trúc sysinfo lúc đầu.