

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN



REPORT

PROJECT 2:

System call

Instructor : Thầy Lê Viết Long

Subject : Hệ điều hành

Class : CQ2023/22

HO CHI MINH CITY, NOVEMBER 2025

GROUP MEMBERS

| NO. | Full Name | Student ID | Assignments | |
|-----|------------------------|------------|---------------------------------------|--------------|
| 1 | Ngô Thị Thục Quyên | 23120348 | Sysinfo | Viết báo cáo |
| 2 | Trần Đình Thi (Leader) | 23120359 | Merge code, xử lý xung đột và nộp bài | |
| 3 | Nguyễn Hoàng Anh Thư | 23120365 | Sysinfo | |
| 4 | Lục Hoàng Tuấn | 23120393 | Trace | |
| 5 | Huỳnh Trọng Viên | 23120403 | Trace | |

Mục lục

| | |
|--------------------------------|----------|
| I. TRACE..... | 1 |
| 1. Mục tiêu..... | 1 |
| 2. Thiết kế và triển khai..... | 1 |
| 3. Kiểm thử:..... | 2 |
| II. SYSINFO..... | 3 |
| 1. Mục tiêu:..... | 3 |
| 2. Triển khai:..... | 3 |
| 3. Kiểm thử..... | 5 |
| 4. Khó khăn..... | 5 |

I. TRACE

1. Mục tiêu

Mục đích của bài thực hành là triển khai một tính năng theo dõi (tracing) các lời gọi hệ thống (system call) trong hệ điều hành xv6. Tính năng này sẽ hỗ trợ quá trình gỡ lỗi (debug) cho các bài thực hành sau.

Yêu cầu cụ thể:

- + Thêm system call mới: **trace(int mask)**.
- + Chức năng: Bật chế độ theo dõi cho tiến trình gọi và các tiến trình con của nó.
- + Cơ chế: Kernel sẽ in ra thông tin mỗi khi system call được gọi nằm trong **mask** (dựa trên bitmask).
- + Định dạng output: **PID: syscall name -> return_value**.
- + Viết chương trình người dùng **user/trace.c** để thực thi các lệnh khác dưới chế độ theo dõi.

2. Thiết kế và triển khai

a) Khai báo prototype

- kernel/defs.h: thêm `uint64 sys_trace(void);`

// Để toàn bộ kernel có thể gọi được hàm xử lý system call mới mà không gặp lỗi biên dịch, ta cần khai báo prototype cho hàm **sys_trace**.

b) Cập nhật cấu trúc dữ liệu và định nghĩa:

- struct **proc** trong kernel/proc.h: thêm trường **tracemask**.

// Nếu bit thứ **i** được bật, hệ thống sẽ in log cho system call thứ **i**.

- kernel/syscall.h: thêm `#define SYS_trace 22`

c) System Call Handler

- kernel/sysproc.c: thêm hàm `sys_trace(void)`

d) Copy mask trong fork

- Hàm `fork()` trong kernel/proc.c: thêm `np->tracemask = p->tracemask;`

e) In log trong Syscall Dispatcher

- kernel/syscall.c:

+ Khai báo mảng ánh xạ tên: `syscall char *syscalls[] = { "fork", "exit", ... };`

+ Chèn logic kiểm tra vào hàm `syscall()`.

f) Thêm prototype và stub cho user

- user/user.h: khai báo hàm `int trace(int);`
- user/usys.pl: thêm entry để tạo mã assembly stub `entry("trace")`

Mục **UPPROGS** trong Makefile: thêm `$U/_trace\`

3. Kiểm thử:

- Lệnh: `$ trace 32 grep hello README`

```
xv6 kernel is booting
hart 2 starting
hart 1 starting
init: starting sh
$ trace 32 grep hello README
3 : syscall read -> 1023
3 : syscall read -> 991
3 : syscall read -> 438
3 : syscall read -> 0
$
```

- Lệnh: `$ trace 2145483647 grep hello README`

```
$ trace 2145483647 grep hello README
5 : syscall trace -> 0
5 : syscall read -> 1023
5 : syscall read -> 991
5 : syscall read -> 438
5 : syscall read -> 0
5 : syscall close -> 0
$
```

- Lệnh: `$ grep hello README`

```
$ grep hello README
$
```

- Lệnh: `$ trace 2 usertests forkforkfork`

```
$ trace 2 usertests forkforkfork
usertests starting
7 : syscall fork -> 8
test forkforkfork: 7 : syscall fork -> 9
9 : syscall fork -> 10
10 : syscall fork -> 11
10 : syscall fork -> 12
11 : syscall fork -> 13
10 : syscall fork -> 14
11 : syscall fork -> 15
10 : syscall fork -> 16
11 : syscall fork -> 17
10 : syscall fork -> 18
11 : syscall fork -> 19
10 : syscall fork -> 20
12 : syscall fork -> 21
10 : syscall fork -> 22
11 : syscall fork -> 23
10 : syscall fork -> 24
12 : syscall fork -> 25
12 : syscall fork -> 26
10 : syscall fork -> 27
11 : syscall fork -> 28
```

II. SYSINFO

1. Mục tiêu:

Phần Sysinfo yêu cầu bổ sung một system call mới tên là sysinfo vào hệ điều hành xv6. System call này nhận một con trỏ đến struct sysinfo và kernel sẽ ghi vào hai trường:

- + freemem – số byte bộ nhớ trống.
- + nproc – số tiến trình có trạng thái khác UNUSED.

Bài làm được xem là hoàn thành khi chương trình sysinfotest chạy và in đúng thông báo:

sysinfotest: OK

2. Triển khai:

a) Cấu trúc sysinfo:

File (kernel/sysinfo.h) chứa cấu trúc:

```
struct sysinfo {
    uint64 freemem;
    uint64 nproc;
```

```
};
```

user/user.h: thêm `struct sysinfo`; và `int sysinfo(struct sysinfo *)`;

b) Thu thập dung lượng bộ nhớ trống (getfreemem):

kernel/kalloc.c: thêm hàm `getfreemem(void)`

Việc lấy số lượng bộ nhớ trống được thực hiện bằng cách:

- + Duyệt danh sách các trang trống của allocator(freelist).
- + Mỗi node tương ứng 1 trang 4096 bytes.
- + Tổng bộ nhớ trống được tính bằng số node nhân với kích thước trang.

Việc truy cập freelist được bảo vệ bằng lock để tránh xung đột với allocator.

c) Thu thập số tiến trình đang hoạt động (getnproc):

kernel/proc.c: thêm hàm `getnproc`

Quy trình:

- + Duyệt toàn bộ bảng proc[].
- + Mỗi tiến trình được lock trước khi đọc trạng thái để đảm bảo an toàn.
- + Mỗi tiến trình có trạng thái khác UNUSED được tính vào kết quả.

Việc sử dụng lock giúp tránh lỗi đọc sai trạng thái do các tiến trình khác cập nhật không đồng bộ.

d) Xử lý system call sysinfo:

kernel/sysproc.c: thêm `sys_sysinfo(void)`

- + Dùng `argaddr()` để lấy địa chỉ vùng nhớ user truyền vào.
- + Một biến `struct sysinfo` tạm thời trong kernel được khởi tạo.
- + Hai trường `freemem` và `nproc` được gán giá trị thông qua hàm đã triển khai trước đó.
- + Dữ liệu được sao chép ra user space bằng `copyout()`.

e) Cập nhật syscall table:

- kernel/syscall.h: Thêm mã syscall `SYS_sysinfo`
- user/usys.pl: thêm stub
- Bảng `syscalls[]` trong kernel/syscall.c: thêm entry

Mục **UPROGS** trong Makefile: thêm `$U/_sysinfotest`

3. Kiểm thử

```
xv6 kernel is booting

hart 1 starting
hart 2 starting
init: starting sh
$ sysinfotest
sysinfotest: starting
sysinfotest: test sysinfo
sysinfo success
sysinfotest: test memory
free memory: 133238784 bytes
memory test success
sysinfotest: test processes
number of processes: 3
process test success
sysinfotest: OK
$
```

4. Khó khăn

Lỗi: Quên define syscall trong sysc

```
#define SYS_sysinfo 22
```

Sau đó chạy lại make clean && make qemu để cập nhật bảng syscalall.h

- + Nguyên do: Mã số system call SYS_sysinfo không được khai báo trong kernel/syscall.h, khiến kernel không ánh xạ được syscall.
- + Cách khắc phục: Thêm dòng #define SYS_sysinfo 22
Sau đó chạy lại make clean && make qemu để cập nhật bảng syscall.