

实验报告

Lab 2

姓名：施君豪

班级：20 人工智能

学号：20307140008

实验报告填写要求

- 1.请在每个 exercise 之后简要叙述实验原理，详细描述实验过程。可以使用中文进行描述，不对语言做要求。
- 2.请将你认为的关键步骤附上必要的截图。
- 3.有需要写代码的实验，必须配有代码、注释以及对代码功能的说明。
- 4.你还可以列举包括但不限于以下方面:实验过程中碰到的问题你是如何解决的、实验之后你还留有哪些疑问和感想。
- 5.如果实验附有练习，请在每个练习之后作答，这是实验报告评分的重要部分。
- 6.Challenge 为加分选作题。每个 lab 可能有多个 challenge,我们会根据完成情况以及难度适当加分，具体情况会在课上说明。这部分的实验过程描述应该比 exercise 更加详细。（请注意，Lab0 为基础环境配置，不设置挑战问题。）
- 7.切勿抄袭亦或是去互联网复制粘贴答案。

【练习题模板】

- 1、Question
- 2、Code
- 3、Screenshot
- 4、Difficulties and solutions

实验练习一：

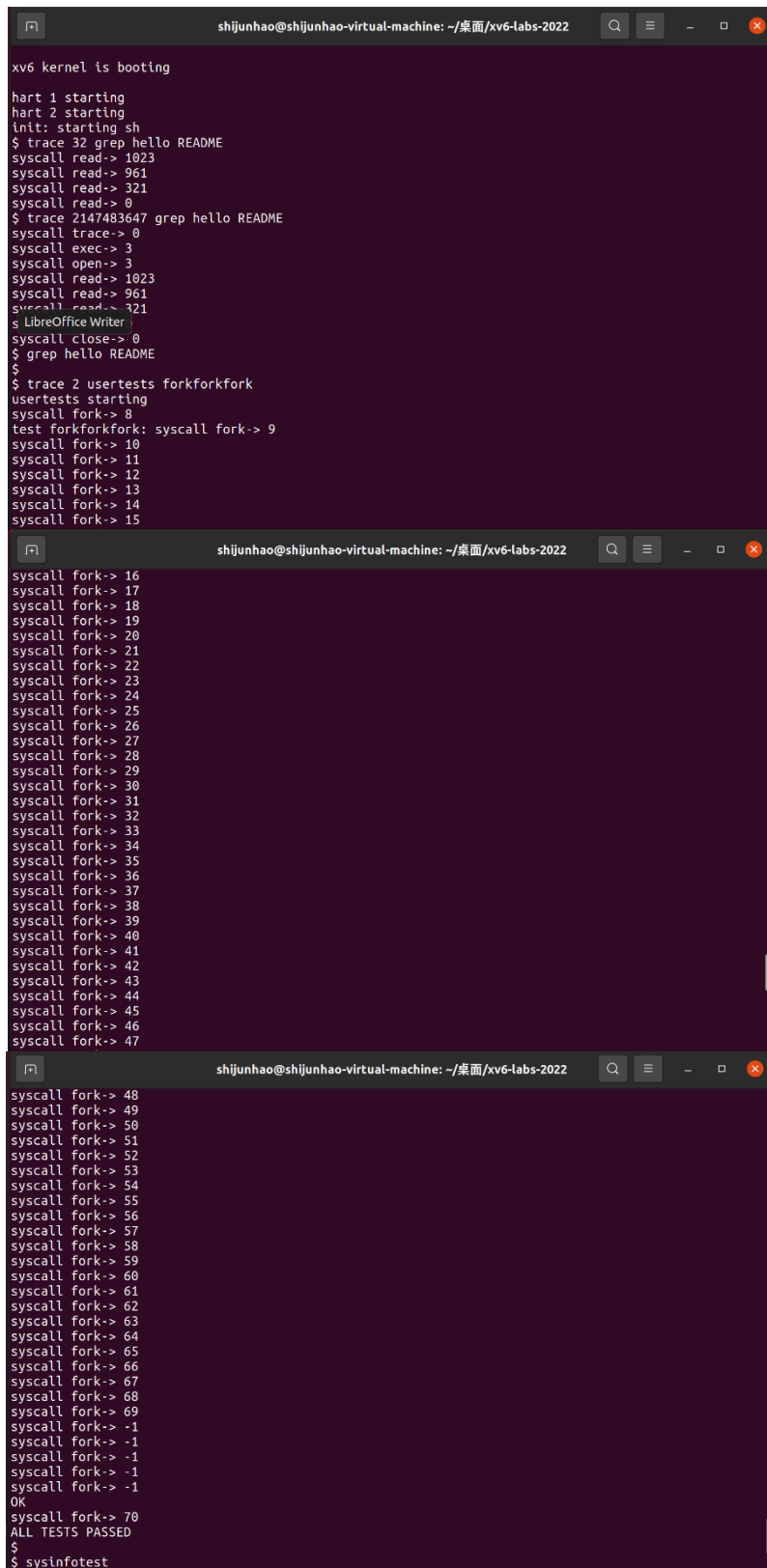
It should take one argument, an integer "mask", whose bits specify which system calls to trace.

For example, to trace the fork system call, a program calls `trace(1 << SYS_fork)`, where `SYS_fork` is a syscall number from `kernel/syscall.h`. You have to modify the xv6 kernel to print out a line when each system call is about to return, if the system call's number is set in the mask. The line should contain the process id, the name of the system call and the return value; you don't need to print the system call arguments.

The trace system call should enable tracing for the process that calls it and any children that it subsequently forks, but should not affect other processes.

一、代码：修改文件较多，见附件

二、运行截图：



The image displays three sequential screenshots of a terminal window running the xv6 kernel. The window title is 'shijunhao@shijunhao-virtual-machine: ~/桌面/xv6-labs-2022'. The terminal output shows the kernel booting and executing various tests.

```
xv6 kernel is booting
hart 1 starting
hart 2 starting
init: starting sh
$ trace 32 grep hello README
syscall read-> 1023
syscall read-> 961
syscall read-> 321
syscall read-> 0
$ trace 2147483647 grep hello README
syscall trace-> 0
syscall exec-> 3
syscall open-> 3
syscall read-> 1023
syscall read-> 961
syscall read-> 321
$ LibreOffice Writer
syscall close-> 0
$ grep hello README
$
$ trace 2 usertests forkforkfork
usertests starting
syscall fork-> 8
test forkforkfork: syscall fork-> 9
syscall fork-> 10
syscall fork-> 11
syscall fork-> 12
syscall fork-> 13
syscall fork-> 14
syscall fork-> 15

syscall fork-> 16
syscall fork-> 17
syscall fork-> 18
syscall fork-> 19
syscall fork-> 20
syscall fork-> 21
syscall fork-> 22
syscall fork-> 23
syscall fork-> 24
syscall fork-> 25
syscall fork-> 26
syscall fork-> 27
syscall fork-> 28
syscall fork-> 29
syscall fork-> 30
syscall fork-> 31
syscall fork-> 32
syscall fork-> 33
syscall fork-> 34
syscall fork-> 35
syscall fork-> 36
syscall fork-> 37
syscall fork-> 38
syscall fork-> 39
syscall fork-> 40
syscall fork-> 41
syscall fork-> 42
syscall fork-> 43
syscall fork-> 44
syscall fork-> 45
syscall fork-> 46
syscall fork-> 47

syscall fork-> 48
syscall fork-> 49
syscall fork-> 50
syscall fork-> 51
syscall fork-> 52
syscall fork-> 53
syscall fork-> 54
syscall fork-> 55
syscall fork-> 56
syscall fork-> 57
syscall fork-> 58
syscall fork-> 59
syscall fork-> 60
syscall fork-> 61
syscall fork-> 62
syscall fork-> 63
syscall fork-> 64
syscall fork-> 65
syscall fork-> 66
syscall fork-> 67
syscall fork-> 68
syscall fork-> 69
syscall fork-> -1
syscall fork-> -1
syscall fork-> -1
syscall fork-> -1
syscall fork-> -1
OK
syscall fork-> 70
ALL TESTS PASSED
$
$ sysinfotest
```

(grade-syscall 的 txt 输出已附在附件中)

三、困难与解决办法

一开始在按照提示加函数名的时候遗漏了在 syscall.c 中添加 trace 名，导致 make qemu 后无法识别函数名，完成用户态和内核中间的通道，在检查后得以解决。

除此以外，对 myproc 的结构不熟悉也在一开始对编写代码造成了一定困难，在查阅相关代码后顺利解决。

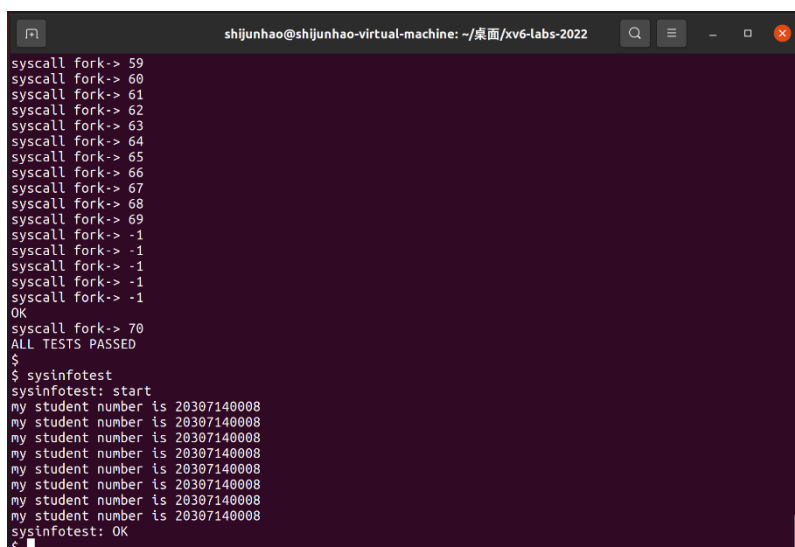
实验练习二：

In this assignment you will add a system call, sysinfo, that collects information about the running system. The system call takes one argument: a pointer to a struct sysinfo (see kernel/sysinfo.h). The kernel should fill out the fields of this struct: the freemem field should be set to the number of bytes of free memory, and the nproc field should be set to the number of processes whose state is not UNUSED. We provide a test program sysinfotest; you pass this assignment if it prints "sysinfotest: OK"

一、代码：

详见附件

二、屏幕截图

A terminal window titled 'shijunhao@shijunhao-virtual-machine: ~/桌面/xv6-labs-2022'. The terminal shows a series of 'syscall fork->' commands with values from 59 to 70. After syscall 70, it says 'ALL TESTS PASSED'. Then, it runs '\$ sysinfotest', which outputs 'sysinfotest: start' followed by eight lines of 'my student number is 20307140008'. Finally, it outputs 'sysinfotest: OK'.

(grade-syscall 的 txt 输出已附在附件中)

三、困难与解决办法

一开始在编写新函数时遗漏了在 def 中新增函数名的定义，导致最初函数编译没有通过。除此以外，在调用 copyout 函数时在查看 sysfstat 和 filestat 后仍然存在疑惑，在反复调试并上网查询后得以解决。在查看内存相关数据时遇到了一些困惑，在查阅 xv6 手册后顺利解决。

问题回答：

(1) System calls Part A 部分，简述一下 trace 全流程.

在输入命令后，user/trace.c 调用 trace 系统调用，随后 markfile 调用 usys.pl 代码，由于已经在 usys.pl 中已经添加了入口，首先 user/usys.S 中对应的汇编代码段将会得到执行：它会将对应的编号 (trace 为 22 号, kernel/syscall.h 中添加) 传入 a7 寄存器，随后执行 ecall 汇编指令使得进程陷入内核态。(下为

user/usys.S 中对应代码段)

```
li a7, SYS_trace
```

```
ecall
```

```
ret
```

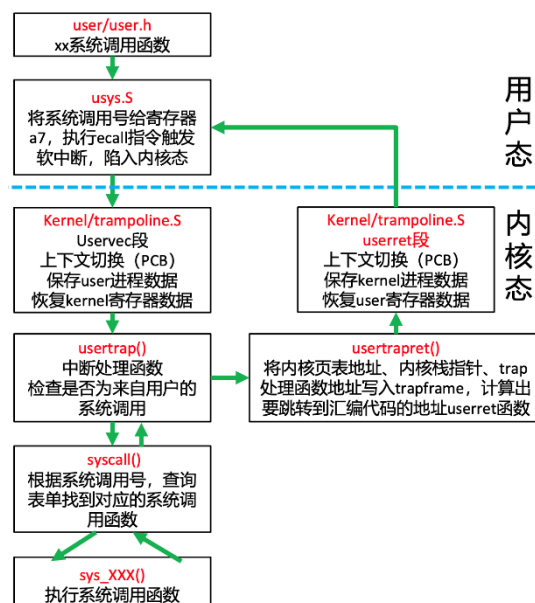
陷入内核态时, kernel/trampoline.S 中 uservec 段的汇编代码将会得到执行, 在内存 TRAPFRAME 中保存用户的所有寄存器 (把 a7 寄存器中的系统编号 22 入了 TRAPFRAME), 然后恢复一些内核所必要的寄存器, 最后调用 usertrap()

usertrap() 函数 (kernel/trap.c) 处理所有用户空间导致的 trap, 把返回地址设为当前 pc 加 4, 返回到 ecall 的下一条指令, 然后调用 syscall();

syscall() 在获取系统调用编号 22 后, 然后找到对应系统调用的处理函数 sys_trace() (即自己添加的部分), 获取调用的 tracenum, 然后返回 syscall, 在满足条件时打印输出直至结束;

返回时, 首先执行 usertrapret() (kernel/trap.c), 然后执行 trampoline.S 中的 userret 段汇编代码, 中断返回用户态。

(下为引用的 xv6 运行流程图)



(2) kernel/syscall.h 是干什么的，如何起作用的？

kernel/syscall.h 作为头文件定义了系统调用函数对应的系统调用号，syscall 通过获取的系统调用号在 syscall.h 中找到对应的系统函数，从而完成对应功能。

```
// System call numbers

#define SYS_fork    1

#define SYS_exit    2

#define SYS_wait    3
```

(3) 命令 “trace 32 grep hello README”中的 trace 字段是用户态下的还是实现的系统调用函数 trace？

前者（用户态），此处的 trace 对应的是 usys.pl 中的 entry (“trace”)，然后调用 ecall 发生中断进入内核态。系统调用函数 sys_trace 为通过 syscall 调用，无法直接通过命令调用。

（下为 usys.pl 中代码）

```
sub entry {

    my $name = shift;

    print ".global $name\n";

    print "${name}:\n";

    print " li a7, SYS_${name}\n";

    print " ecall\n";

    print " ret\n";

}
```


Part 3: The Ending

通过本实验我了解了 xv6 系统用户态与内核态的工作流程，加深了对系统函数调用过程的理解，同时并在实验过程中尝试了自己手动添加系统调用。

一个小建议是如果可以的话可以在 pdf 中单独列出本次实验会涉及到修改的文件名称，有时候文件太多会疏漏其中一两个的修改。

(以及感受到文件版本迭代管理的重要性了，git 在看了在看了呜呜)

附流程图引用网址：

<https://xyfjason.top/2021/11/30/xv6-mit-6-S081-2020-Lab2-syscall/>