Lab2 System calls 实验要求

一、实验简介

本次 Lab 目的是为 xv6 系统添加新的系统调用函数，了解 xv6 内核的一些内部机制。

二、实验准备

「1」获取 xv6 Lab2 源码：

```
$ git clone git://g.csail.mit.edu/xv6-labs-2022
$ git checkout syscall
$ make clean
```

「2」建议先阅读 xv6 book 的「Chapter 2」和「Chapter 4 的 4.3、4.4」两节以及相关的代码文件.

* The user-space “stubs” that route system calls into the kernel are in ***user/usys.S***, which is generated by ***user/usys.pl*** when you run make. Declarations are in ***user/user.h.***

* The kernel-space code that routes a system call to the kernel function that implements it is in ***kernel/syscall.c*** and ***kernel/syscall.h***.

* Process-related code is ***kernel/proc.h*** and ***kernel/proc.c***.

三、实验要求

3.1 代码实现

■ Part A: System call tracing

「该部分需要实现系统调用跟踪功能，需要创建一个新的系统调用 `trace`」

> 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.

「提示」

• Add $U/_trace to UPROGS in Makefile

• Run make qemu and you will see that the compiler cannot compile *user/trace.c*, because the user-space stubs for the system call don't exist yet: add a prototype for the system call to *user/user.h*, a stub to *user/usys.pl*, and a syscall number to *kernel/syscall.h*. The Makefile invokes the perl script *user/usys.pl*, which produces *user/usys.S*, the actual system call stubs, which use the RISC-V ecall instruction to transition to the kernel. Once you fix the compilation issues, run trace 32 grep hello README; it will fail because you haven't implemented the system call in the kernel yet.

• Add a sys_trace() function in *kernel/sysproc.c* that implements the new system call by remembering its argument in a new variable in the proc structure (see *kernel/proc.h*). The functions to retrieve system call arguments from user space are in *kernel/syscall.c*, and you can see examples of their use in *kernel/sysproc.c*.

• Modify fork() (see *kernel/proc.c*) to copy the trace mask from the parent to the child process.

• Modify the syscall() function in *kernel/syscall.c* to print the trace output. You will need to add an array of syscall names to index into.

■ Part B: Sysinfo

「该部分需要创建一个新的系统调用 sysinfo，以实现系统信息收集功能」

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.

In addition, you need to add a structure containing a string type field in the *kernel/sysinfo.h* . This field can be named arbitrarily to indicate your student number. You need also print your student ID when calling the sysinfo function.

We provide a test program sysinfotest. You pass this assignment if it prints "sysinfotest: OK".

「提示」

• Add $U/_sysinfotest to UPROGS in Makefile.

• Run make qemu; *user/sysinfotest.c* will fail to compile. Add the system call sysinfo, following the same steps as in the previous assignment. To declare the prototype for sysinfo() in *user/user.h* you need predeclare the existence of struct sysinfo:

```
struct sysinfo;
int sysinfo(struct sysinfo *);
```

Once you fix the compilation issues, run sysinfotest; it will fail because you haven't implemented the system call in the kernel yet.

• sysinfo needs to copy a struct sysinfo back to user space;
see sys_fstat() (*kernel/sysfile.c*) and filestat() (*kernel/file.c*) for examples of how to do that using copyout().

• To collect the amount of free memory, add a function to *kernel/kalloc.c*.

• To collect the number of processes, add a function to *kernel/proc.c*.

提示非常详细，请同学们仔细参考。

3.2 问题回答

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

（2）kernel/syscall.h 是干什么的，如何起作用的？

（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？

请独立完成实验内容，如果参考或借鉴了网上代码或回答，请在实验报告中标注。

四、实验预期结果

Part A: System call tracing

```
xv6 kernel is booting

hart 1 starting
hart 2 starting
init: starting sh
$
$ trace 32 grep hello README
4: syscall read -> 1023
4: syscall read -> 961
4: syscall read -> 321
4: syscall read -> 0
$
$ trace 2147483647 grep hello README
6: syscall exec -> 3
6: syscall open -> 3
6: syscall read -> 1023
6: syscall read -> 961
6: syscall read -> 321
6: syscall read -> 0
6: syscall close -> 0
$
$ grep hello README
$
```

```
$ trace 2 usertests forkforkfork
usertests starting
10: syscall fork -> 11
test forkforkfork: 10: syscall fork -> 12
12: syscall fork -> 13
13: syscall fork -> 14
13: syscall fork -> 15
14: syscall fork -> 16
13: syscall fork -> 17
14: syscall fork -> 18
13: syscall fork -> 19
14: syscall fork -> 20
13: syscall fork -> 21
13: syscall fork -> 22
14: syscall fork -> 23
14: syscall fork -> 24
22: syscall fork -> 25
13: syscall fork -> 26
14: syscall fork -> 27
13: syscall fork -> 28
26: syscall fork -> 29
26: syscall fork -> 30
26: syscall fork -> 31
26: syscall fork -> 32
27: syscall fork -> 33
13: syscall fork -> 34
13: syscall fork -> 35
13: syscall fork -> 36
14: syscall fork -> 37
13: syscall fork -> 38
13: syscall fork -> 39
25: syscall fork -> 40
22: syscall fork -> 41
22: syscall fork -> 42
35: syscall fork -> 43
13: syscall fork -> 44
17: syscall fork -> 45
15: syscall fork -> 46
15: syscall fork -> 47
OK
10: syscall fork -> 48
ALL TESTS PASSED
$
```

Part B: Sysinfo

```
xv6 kernel is booting

hart 2 starting
hart 1 starting
init: starting sh
$
$
$ sysinfotest
sysinfotest: start
my student number is 1234567890
my student number is 1234567890
my student number is 1234567890
my student number is 1234567890
my student number is 1234567890
my student number is 1234567890
my student number is 1234567890
my student number is 1234567890
sysinfotest: OK
```

五、实验提交

（1）实验代码文件，提交前请 make clean.

（2）实验报告（中文即可，PDF 格式），需包含以下内容：

「1」实验实现的思路，相关代码截图，最终测试结果的截图.

「2」问题回答.

「3」实验中碰到何种问题，如何解决的.

「4」实验感想.

（3）报告截止日期：2022 年 10 月 25 日晚 22 点. 请按时提交.

（4）提交邮箱：各班级助教邮箱.

请将（1）、（2）放入一个文件夹，命名为：学号_姓名_oslab2.zip.

附：

助教邮箱

施恬安	302 室	21210240071@m.fudan.edu.cn
张卫华	204、205 室	15751006003@163.com
乔羽	202 室	22212010027@m.fudan.edu.cn