哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验设计报告

开课学期：2021 年秋

课程名称：操作系统实验

实验名称：实验 2：系统调用

实验性质：课内实验

实验时间：10/21 地点：T2608

学生班级：2019 级 4 班

学生学号：190110419

学生姓名：李怡凯

评阅教师：

报告成绩：

实验与创新实践教育中心印制

2018 年 12 月

一、 回答问题

1. 阅读 kernel/syscall.c，试解释函数 syscall()`如何根据系统调用号调用对应的系统调用处理函数（例如 sys_fork）？syscall()`将具体系统调用的返回值存放在哪里？

   通过函数指针将系统调用号映射为对应的系统调用处理函数；存放在进程对应的用户进程的寄存器 a0 当中。

2. 阅读 kernel/syscall.c，哪些函数用于传递系统调用参数？试解释 argraw()函数的含义。

   argint, argstr, argaddr; 读取寄存器 a0~a5 的值，用于内核态和用户态之间的参数传递；

3. 阅读 kernel/proc.c 和 proc.h，进程控制块存储在哪个数组中？进程控制块中哪个成员指示了进程的状态？一共有哪些状态？

   proc 数组；state；UNUSED，SLEEPING，RUNNABLE，RUNNING，ZOMBIE

4. 阅读 kernel/kalloc.c，哪个结构体中的哪个成员可以指示空闲的内存页？Xv6 中的一个页有多少字节？

   Kmem.freelist 为指向空闲内存页链表的指针；4096；

5. 阅读 kernel/vm.c，试解释 copyout()函数各个参数的含义。

   pagetable：指向进程的页文件

   dstva：目标位置起始地址

   src：待复制内容起始地址

   len：待复制字节数

二、 实验详细设计

1. 系统调用 trace

   接口定义：int trace(int mask)

   参数 mask 指示跟踪的系统调用号，将需要跟踪的系统调用号设置为 1；

   正常执行返回 0，否则返回-1；

   1.1 功能描述

   trace 系统调用的功能为设置进程的系统调用追踪，追踪 mask 设定的系统调用，当检测到系统调用发生时，按照如下格式打印系统调用的相关信息。

```
PID: sys_$name(arg0) -> return_value
```

1.2 设计方案

为了使进程能够记录自己所追踪的系统调用，需要修改 PCB 对应的 proc 结构体，添加变量 mask 记录追踪的系统调用。此外，当进程调用 fork 函数产生子进程时，其子进程也需要追踪父进程所追踪的系统调用，所以需要对 fork 函数的逻辑进行修改，添加子进程继承父进程 mask 的逻辑。

系统调用函数 sys_trace 流程图如图 1 所示。通过 sys_trace 函数能够为进程设置其跟踪的系统调用，记录在 proc 结构体 mask 变量中。

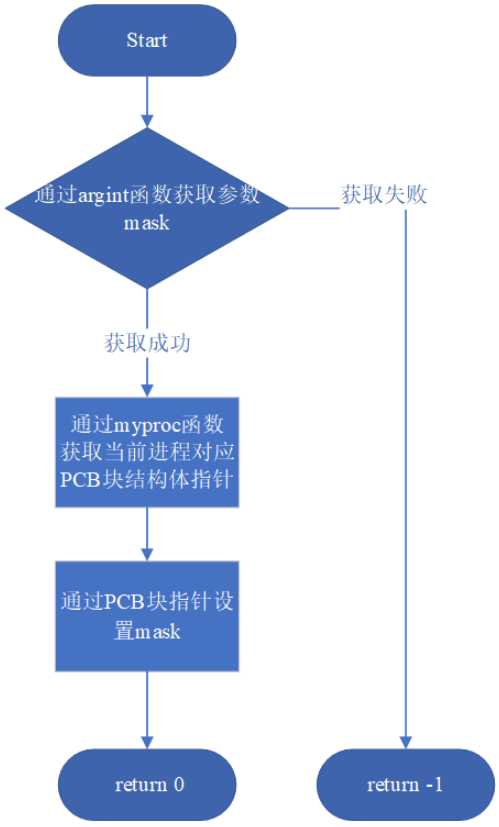

图 1 sys_trace 函数流程图

经过修改后的系统调用入口函数 syscall 流程图如图 2 所示。通过修改部分 syscall 函数的逻辑，使其在系统调用返回后判断系统调用是否是当前进程正在追踪的系统调用，如果是则打印所需要的信息。

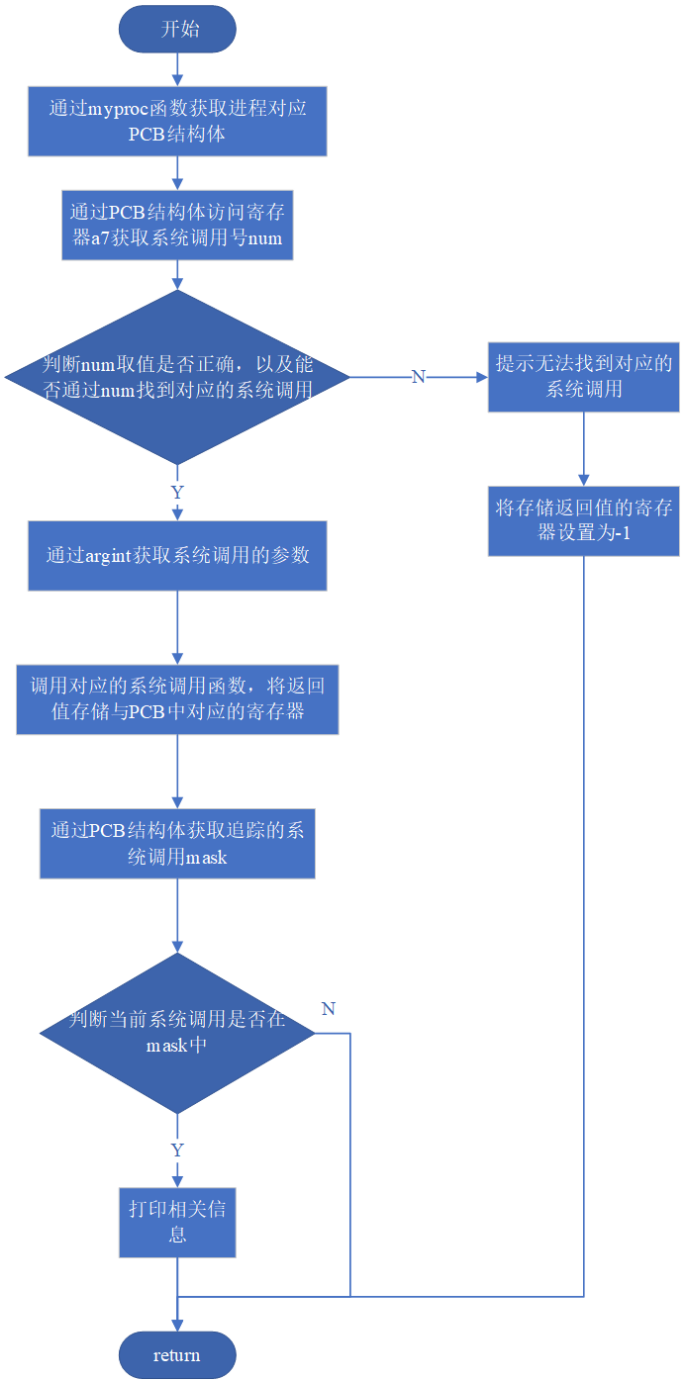

图 2 syscall 函数流程图

2. 系统调用 sysinfo

接口定义：int sysinfo(struct sysinfo *)

输入参数：sysinfo 结构体指针，其中包含剩余可用内存空间，剩余未使用进程数，以及剩余可用文件描述符的数量；

返回值：成功返回 0，否则返回-1

2.1 功能介绍

sysinfo 系统调用用于查看当前系统与进程的相关信息。

2.2 设计方案

由于系统信息需要由内核态来获取，所以该系统调用需要在内核态中完

成信息的收集，然后由内核态拷贝到用户态对应的地址空间中实现数据传送。

其中，计算剩余可用空间可以通过遍历保存可用页表的链表，计算出可用页表的数量，然后乘以页表的大小得到；计算 UNUSED 进程数可以通过遍历保存进程 PCB 的数组，统计其中 state 变量为 UNUSED 的进程数量得到；计算进程剩余可用文件描述符的数量可以通过遍历 PCB 块中的 ofile 数组，统计其中为空的元素数量得到。

sysinfo 函数流程图如图 3 所示。

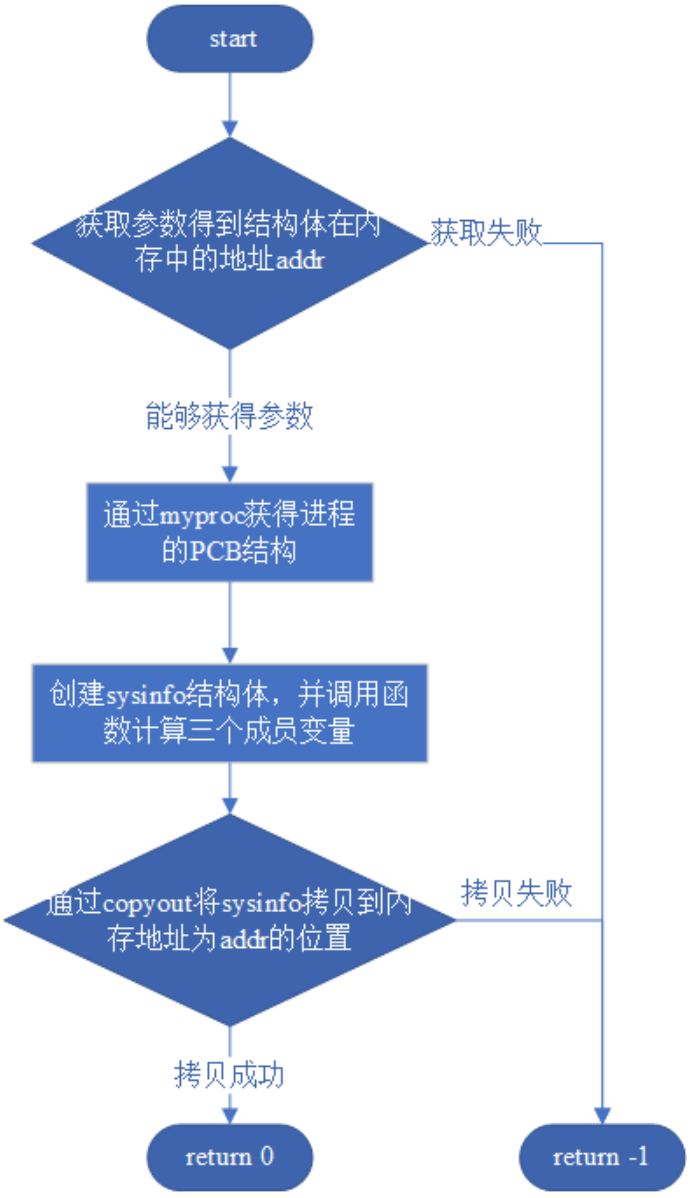

图 3 sysinfo 函数流程图

三、 实验结果截图

实验结果截图如图 4 所示。

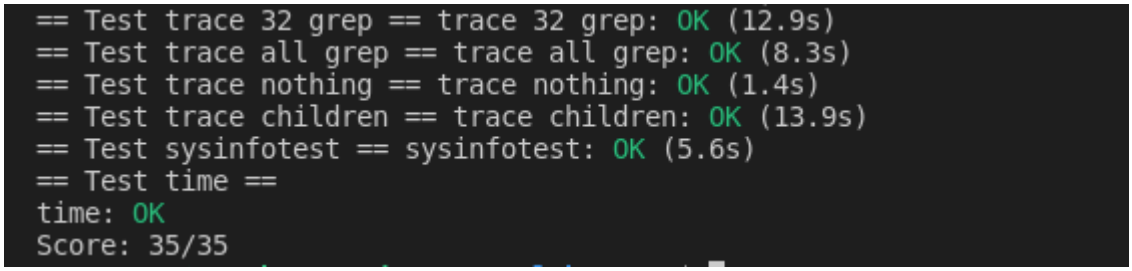
== Test trace 32 grep == trace 32 grep: OK (12.9s)
== Test trace all grep == trace all grep: OK (8.3s)
== Test trace nothing == trace nothing: OK (1.4s)
== Test trace children == trace children: OK (13.9s)
== Test sysinfotest == sysinfotest: OK (5.6s)
== Test time ==
time: OK
Score: 35/35

图 4 实验结构截图