

Operating System Lab 2 Report

by PB18000227 艾语晨

添加Linux系统调用

大概描述实验过程

增加系统调用需要修改相关文件，和具体的函数实现

就是按照实验文档所说的，先看一下后面的背景知识，并借助上一次习题课的讲解，最终完成

展示实验结果

```
[/usr]# ./test
Give me a string:
123
in sys_print_val: 123
[/usr]#
```

实验代码（改变部分）

unistd.h，左边的是include文件夹下，右边的是文件系统里面

```
130 #define __NR_setreuid 70
131 #define __NR_setregid 71
132 #define __NR_print_val 72
133 #define __NR_str2num 73
134
```

sys.h

```
✓ include
  > asm
  ✓ linux
    C config.h
    C fdreg.h
    C fs.h
    C hreg.h
    C head.h
    C kernel.h
    C mm.h
    C sched.h
    C sys.h
    C tty.h
  > sys
C a.out.h
C const.h
C ctype.h
C errno.h
C fcntl.h
C signal.h
C stdarg.h
C stddef.h

72 extern int sys_setregid();
73 extern int sys_print_val(int a);
74 extern int sys_str2num(char *str, int str_len, long *ret);
75
76 fn_ptr sys_call_table[] = { sys_setup, sys_exit, sys_fork, sys_read,
77 sys_write, sys_open, sys_close, sys_waitpid, sys_creat, sys_link,
78 sys_unlink, sys_execve, sys_chdir, sys_time, sys_mknod, sys_chmod,
79 sys_chown, sys_break, sys_stat, sys_lseek, sys_getpid, sys_mount,
80 sys_umount, sys_setuid, sys_getuid, sys_stime, sys_ptrace, sys_alarm,
81 sys_fstat, sys_pause, sys_utime, sys_stty, sys_gtty, sys_access,
82 sys_nice, sys_ftime, sys_sync, sys_kill, sys_rename, sys_mkdir,
83 sys_rmdir, sys_dup, sys_pipe, sys_times, sys_prof, sys_brk, sys_setgid,
84 sys_getgid, sys_signal, sys_geteuid, sys_getegid, sys_acct, sys_phys,
85 sys_lock, sys_ioctl, sys_fcntl, sys_mpx, sys_setpgid, sys_ulimit,
86 sys_uname, sys_umask, sys_chroot, sys_ustat, sys_dup2, sys_getppid,
87 sys_getpgrp, sys_setsid, sys_sigaction, sys_sgetmask, sys_ssetmask,
88 sys_setreuid, sys_setregid, sys_print_val, sys_str2num };
89
```

system_call.s

```
61 nr_system_calls = 74
```

Makefile

```

33
34 OBJS = sched.o system_call.o traps.o asm.o fork.o \
35 panic.o printk.o vsprintf.o sys.o exit.o \
36 signal.o mktime.o print_str2num.o

```

```

86 ../include/asm/system.h ../include/asm/segment.h ../include/asm/io.h
87 vsprintf.s vsprintf.o: vsprintf.c ../include/stdarg.h ../include/string.h
88 print_str2num.s print_str2num.o: print_str2num.c ../include/asm/segment.h
89

```

print_str2num.c

```

C print_str2num.c x
kernel > C print_str2num.c > __LIBRARY__
1  #define __LIBRARY__
2  #include <linux/kernel.h>
3  #include <asm/segment.h>
4
5  int sys_print_val(int a)
6  {
7      printk("in sys_print_val: %d", a);
8  }
9
10 int sys_str2num(char *str, int str_len, long *ret)
11 {
12     /* *ret = (long)atoi(str); */
13     long sum = 0;
14     char temp;
15     for (int i = 0; i < str_len; i++)
16     {
17         temp = get_fs_byte(str + i);
18         sum = sum * 10 + temp - '0';
19     }
20     put_fs_long(sum, ret);
21 }
22

```

回答问题

- 简要描述如何在Linux-0.11添加一个系统调用

Answer :

1. 修改system_call.s中的系统调用数量
 2. 在sys.h、unistd.h中增加相应的声明、函数指针表内容
 3. 在kernel目录下实现相应的函数，并修改makefile
- 系统是如何通过系统调用号索引到具体的调用函数的

Answer :

在`unistd.h`中有系统调用编号，`_syscallx`函数根据这个编号到`sys.h`中的函数指针表找到相应函数原型，然后在kernel目录下找到具体的实现

- 在Linux-0.11中，系统最多支持几个参数？有什么方法可以超过这个限制吗？

Answer :

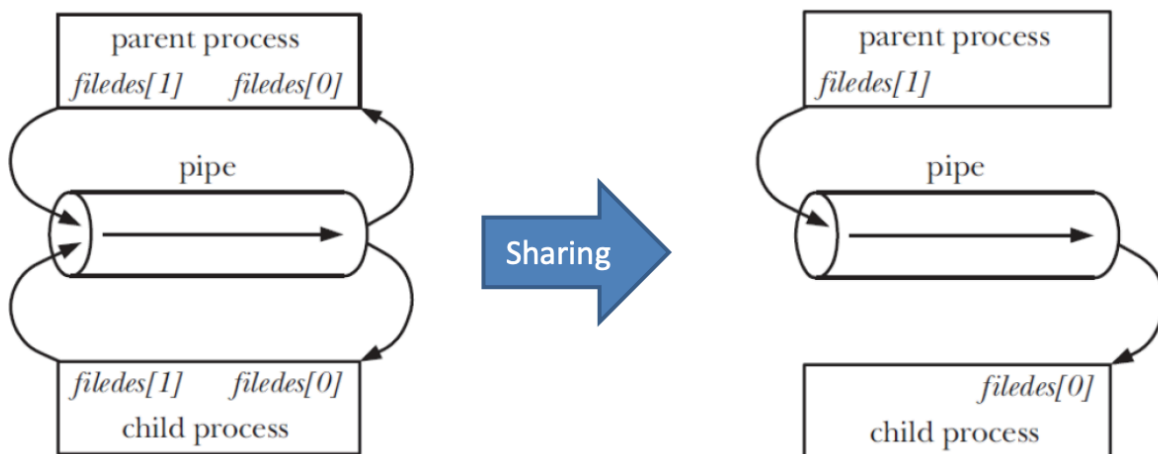
最多支持3个参数。因为在`unistd.h`里面只定义到了`_syscall3()`。超过限制的方式可以通过修改`unistd.h`文件，定义新的`syscallx()`函数

熟悉Linux下常见的系统调用函数

大概描述代码添加过程、思路

`os_open()`

用于pipe的开启工作，原理如下图（取自李老师PPT，ch5part1;Page37）



简单来说思路是生成匿名管道（第一次close关闭没有用的管道一端），重定向STDOUT/STDIN，然后解除管道原有的端口，而PARENT一端就相对简单，只是正常写/读数据即可

`os_system`

通过`strtok`函数进行字符串分割，来分解指令字符串，在执行各命令

```
char *strtok(char *__restrict__ __s, const char *__restrict__ __delim)
Divide S into tokens separated by characters in
DELIM.
```

`main`

分需要pipe和不需要两种

需要pipe:

- cmd1->buf: 开启管道，通过read函数写进缓冲区
- buf->cmd2: 开启管道，通过write函数从缓冲区写出

不需要pipe：直接调用os_system函数即可

展示实验结果

```
[/usr]# ./lab2_shell
os shell ->ls | grep 1
cmd1: ls
cmd2:  grep 1
10987.c
123qwe.txt
os shell ->ls | grep a
cmd1: ls
cmd2:  grep a
lab2_shell
lab2_shell.c
local
var
os shell ->
```