## OS LAB2

刘芷辰 PB21111728 2023年4月27日

## 1. 实验题目

添加Linux系统调用

## 2. 实验目的

- 学习如何使用Linux系统调用: 实现一个简单的shell
- 学习如何添加Linux系统调用:实现一个简单的top

# 3. 实验环境

- OS: Ubuntu 20.04.4 LTS
- Linux内核版本: 4.9.263

## 4. 实验内容

### 实现一个简单的shell (代码填空)

- 1. 实现内置指令
  - cd: 调用chdir系统调用,传入第二个参数即目录
  - exit: 调用exit系统调用, exit (0)
  - kill: 调用kill系统调用,若只有两个参数,则指定kill信号为默认即15,否则将第三个参数作为 kill类型传入,两者都将第二个参数作为pid传入

#### 2. 重定向

>:将重定向符号后一个参数作为文件名打开。打开方式为可读可写、不存在则创建、截断长度为0,成功打开则将文件描述符重定向为写端

- >>: 同上, 只是将截断长度为0改为可追加
- <: 同上,只是打开方式改为只读,成功打开则将文件描述符重定向为读端

#### 3. execute函数

• 经过前面的处理,这里传入的是文件名和argv,因此采用exec族的execvp

#### 4. main函数

- 首先为了实现shell前置的文字描述,使用getcwd获取当前路径,存入字符串然后 输出
- 为了实现":"的多命令执行,使用给出的分割函数将原命令按照";"分开并得到 总的命令数,然后进行for循环单独执行每一条命令
- 没有管道符的且非内置指令的命令,创建一个子进程,在子进程中执行execute函数,并使用wait等待运行结束
- 如果是存在一个管道符,则在子进程1中需要关闭管道的读端,打开写端,最后关闭写端;同理,在子进程2中作为接受输出的一方,需要关闭写端,打开读端,最后关闭读端,并子进程已给出的代码一样按照空格分割后传入execute函数

### 实验一个简单的top

- 注册系统调用 在提交代码中给出
- 声明内核函数原型 在提交代码中给出
- 实现内核函数
  - 首先使用printk打印出相关信息,包括调用的系统调用名以及自己的学号
  - 然后使用for\_each\_process遍历每个task结构体,将task的pid、runtime、state和 comm存入数组,其中comm由于使用二维数组比较繁琐,这里使用一位数组将其 储存,每个comm之间使用';'隔开
  - 通过copy to user将数组传入到用户态
- 编写测试代码
  - 通过pdf给出的ps counter系统调用获取当前的进程数量
  - 调用编写的ps info系统调用在用户态获取task相关信息
  - 处理comm: 使用给出的分割函数按照';'分割开,存入name\_out二维数组
  - 处理pid: 直接传入pid out数组

- 处理state: 通过查看task\_struct结构体定义,发现state的注释中写到runable是0, 其余-1和1不是运行中,因此在用户态中做一个取反处理,用1表示正在运行,传入state\_out
- 处理runtime: ns转换为s, 即除以109后传入runtime out
- 处理CPU占用率:保存上一次获取的runtime,在刷新后得到新的runtime,二者之差再除以1s的间隔,注意单位转换,即为CPU占用率,存入CPU out
- 以CPU out为衡量因素,对得到的out数组进行冒泡排序
- 进行格式处理,将得到的结果输出

### 5. 实验结果

#### 实现一个简单的shell

• 单条指令:

```
shell:/home/lzcnutrition/oslab/lab2 -> ls
exec.sh get_ps_num.c myTop.c ps.txt shell.c shell.out test_shell
```

• cd:

```
shell:/home/lzcnutrition/oslab/lab2 -> cd ..
shell:/home/lzcnutrition/oslab -> echo aaa;echo nbb;echo ccc
```

• ";"多命令:

```
shell:/home/lzcnutrition/oslab -> echo aaa;echo nbb;echo ccc
aaa
nbb
ccc
```

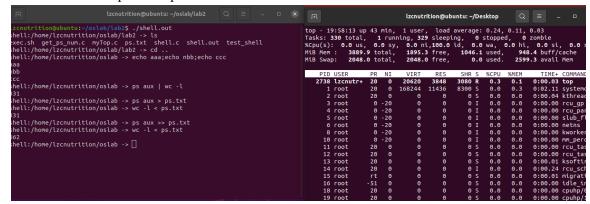
• 单个管道符: shell:/home/lzcnutrition/oslab -> ps aux | wc -l 331

```
shell:/home/lzcnutrition/oslab -> ps aux > ps.txt shell:/home/lzcnutrition/oslab -> wc -l < ps.txt

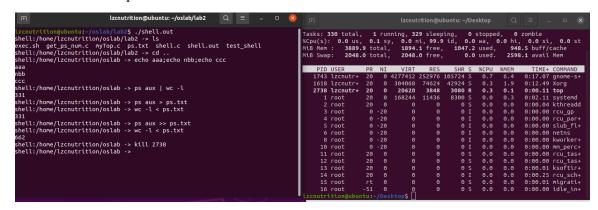
* 重定向符:

**shell:/home/lzcnutrition/oslab -> ps aux >> ps.txt shell:/home/lzcnutrition/oslab -> wc -l < ps.txt 662
```

• kill: 打开一个top找到其pid: 2738

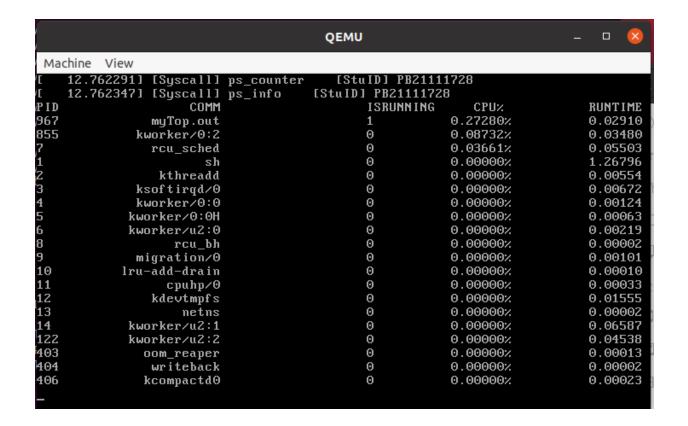


### 执行kill 2738: top命令被kill



exit:
cexit: shell:/home/lzcnutrition/oslab -> exit
cexit: zcnutrition@ubuntu:~/oslab/lab2\$

实验一个简单的top



# 6. 实验总结

- 本次实验很好的帮助我理解了系统调用的实现原理,对于操作系统这门课的学习很有帮助
- 实验文档很详细,介绍清楚
- 助教很耐心,有问题向助教询问总能得到靠谱的解决方式
- 但是本次实验开始只给了一周时间,导致时间安排比较紧,最后到检查的日子了才延期,希望以后能够更好地协调时间安排