上海交通大學

SHANGHAI JIAOTONG UNIVERSITY

操作系统课程设计报告 - Project2

Unix Shell & Linux Kernel Module for Task Information

姓名：薛春宇

学号：5180219106698

完成时间：2020/5/26

一、实验目的

1. 通过实现简易功能的 Unix Shell，了解 Linux 进程和管道通信的基本使用；

2. 学习编写 Linux kernel module，以及文件系统的使用。

二、实验内容

1. 用 C 语言实现一个具有如下功能的简易 Unix Shell：

   （1）历史功能；

   （2）输入、输出重定向；

   （3）进程创建和选择性并行；

   （4）进程间管道通信；

2. 编写一个 kernel modules：利用输入重定向查询特定进程并获取信息。

三、内容一：Unix Shell

1. 实验原理：

该部分由一个作为接受命令行中用户命令并在子进程下执行命令的 Shell 接口的 C 程序组成。其基本外观和使用与真正的 Shell 类似。

实现 Shell 接口的核心技术是父进程首先读入用户在命令行输入的命令，进行相关参数转化和基本判断，然后利用 fork() 来创建一个子进程以完成命令。

除非使用&符号说明，父进程在继续之前会 wait(NULL) 等待子进程退

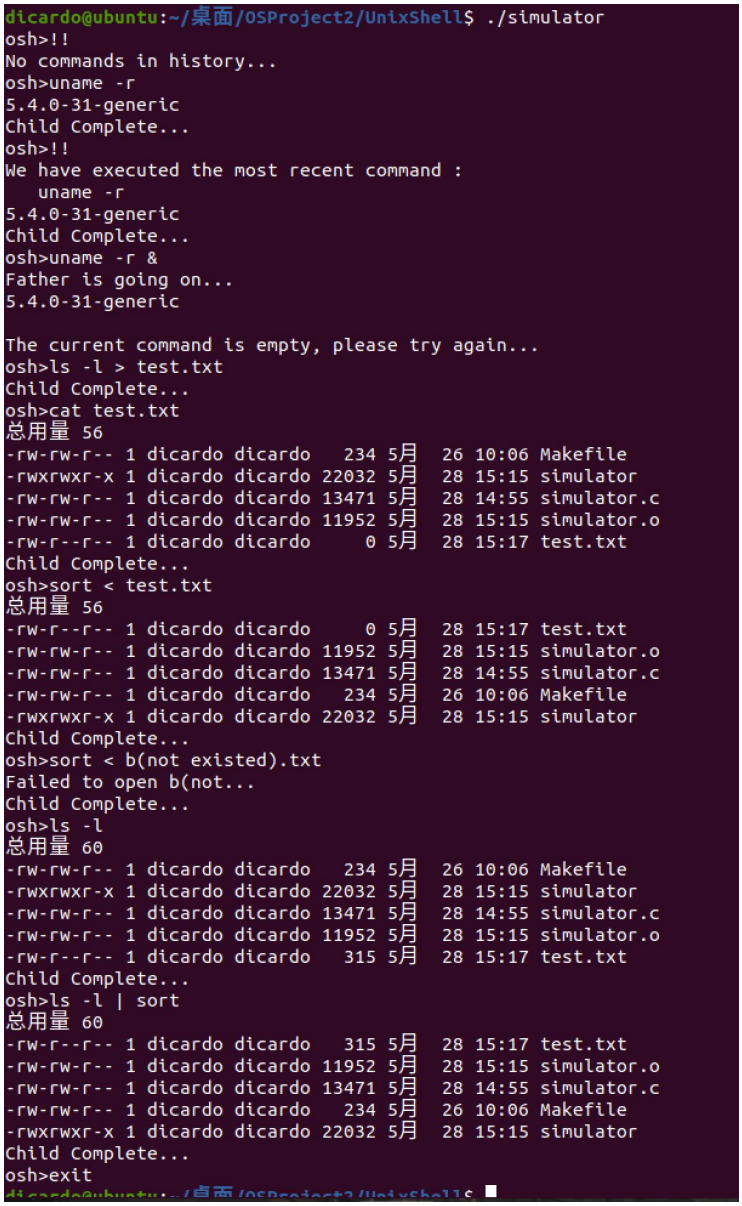
出。

输入、输出的重定向主要利用了 dup2() 函数。

管道通信主要利用了 pipe() 函数，子进程会再次创建一个子进程，并将其输出作为自己的输入。

2. 实验代码：见附件/**UnixShell/simulator.c**

3. 实验结果截图：



四、内容二：Linux Module for Task Information

1. 实验原理：

该部分利用了在 Linux 环境下面向内核编程的技术。通过编写 proc_read() 和 proc_write() 函数，实现了从命令行中读取参数，在内核空间中根据参数访问，将文件系统信息拷贝到用户空间，并最终在命令行中展示。

2. 实验代码：见附件/**TaskInfo/pid.c**

3. 实验结果截图：

```
dicardo@ubuntu:~/桌面/OSProject2/TaskInfo$ sudo insmod pid.ko
[sudo] dicardo 的密码:
dicardo@ubuntu:~/桌面/OSProject2/TaskInfo$ ls /proc
1       168     186     2109    2301  309  373  827          kmsg
10      169     1864    211     24    310  374  832          kpagecgroup
1019    17      187     2117    26    311  38   841          kpagecount
1021    170     18762   2132    27    312  39   842          kpageflags
1028    171     189     2134    277   313  394  845          loadavg
11      172     190     21357   278   314  395  847          locks
1150    173     191     2136    279   315  396  852          mdstat
1196    1737    192     21381   28    316  4    856          meminfo
12      174     1936    2140    280   317  40   863          misc
12708   175     194     2142    281   318  41   893          modules
13      1751    195     2144    282   319  42   894          mounts
136     1752    1953    21444   283   32   43   9            mpt
137     176     1957    2145    284   320  435  900          mtrr
1378    1762    1958    2147    286   321  464  943          net
138     1764    1961    21515   287   322  465  983          pagetypeinfo
139     1769    1964    21517   2872  323  466  acpi         partitions
14      177     1967    2153    2875  324  469  asound       pid
140     1771    1971    2154    289   325  473  buddyinfo    pressure
141     178     1976    2156    29    326  482  bus          sched_debug
142     1787    1984    2158    290   327  483  cgroups      schedstat
143     179     1985    2169    291   328  486  cmdline      scsi
144     1792    1989    2171    292   329  489  consoles     self
15      1798    1992    21791   293   33   506  cpuinfo      slabinfo
150     18      2       2180    294   330  508  crypto       softirqs
151     180     20      21801   295   331  511  devices      stat
154     18010   2001    2182    296   332  572  diskstats    swaps
155     1805    2014    2183    297   333  594  dma          sys
156     1809    2020    2187    298   334  6    driver       sysrq-trigger
157     181     2030    2191    299   335  762  execdomains  sysvipc
158     1815    2036    22      3     336  763  fb           thread-self
159     1819    2040    2200    30    337  776  filesystems  timer_list
16      182     2057    2201    300   338  779  fs           tty
160     1826    207     2203    301   339  799  interrupts   uptime
161     183     2083    22204   302   34   800  iomem        version
162     1831    20860   22247   303   340  803  ioports      version_signature
163     1838    2089    22249   304   35   806  irq          vmallocinfo
164     184     2094    2237    305   36   807  kallsyms     vmstat
165     1841    20951   2262    306   368  811  kcore        zoneinfo
166     185     21      227     307   369  812  keys
167     1850    2107    23      308   370  825  key-users
dicardo@ubuntu:~/桌面/OSProject2/TaskInfo$ echo 154 > /proc/pid
dicardo@ubuntu:~/桌面/OSProject2/TaskInfo$ cat /proc/pid
command = kthrotld pid = 154 state = 1026
dicardo@ubuntu:~/桌面/OSProject2/TaskInfo$ echo 1378 > /proc/pid
dicardo@ubuntu:~/桌面/OSProject2/TaskInfo$ cat /proc/pid
command = colord pid = 1378 state = 1
dicardo@ubuntu:~/桌面/OSProject2/TaskInfo$ sudo rmmod pid
dicardo@ubuntu:~/桌面/OSProject2/TaskInfo$
```

五、实验结果

在经过两天的 coding 之后完成该项目。个人感受是 Unix Shell 的编写难度要远大于 Task Information Module 的难度。前者需要考虑进程创建、进程间通信等第一次着手实现的功能，debug 时需要考虑的因素很多，代码量也较大；而后者的整体框架与 Project 1 类似，因此可以在以前的框架下进一步利用系统调用完善功能。

例如在 Unix Shell 中，为了保证在实现&时的输出格式正确，我额外使用了一次管道通信，以协调父子进程的输出关系。这些 bug 都是在实际运行中慢慢发现并解决的。

六、实验反思

1. Debug 能力和方法还有待提升；
2. 充分利用网络资源，学习相关函数和系统调用用法；