实验一 进程控制

安全 1601 16281120 于星

1. 实验目的：

加深对进程概念的理解，明确进程和程序的区别。

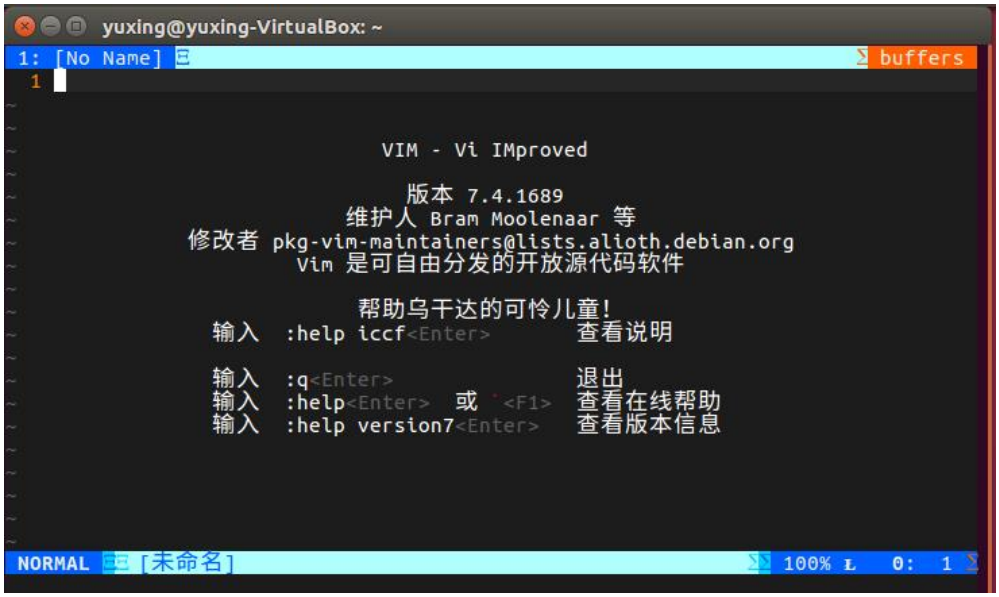
掌握 Linux 系统中的进程创建，管理和删除等操作。

熟悉使用 Linux 下的命令和工具，如 man, find, grep, whereis, ps, pgrep, kill, ptree, top, vim, gcc，gdb，管道等。

2. 实验题目：

1、打开一个 vi 进程。通过 ps 命令以及选择合适的参数，只显示名字为 vi 的进程。寻找 vi 进程的父进程，直到 init 进程为止。记录过程中所有进程的 ID 和父进程 ID。将得到的进程树和由 pstree 命令的得到的进程树进行比较。

（1）首先打开 vim



（2）新建终端使用 ps 命令查看 vi 进程 id，可得知为 2294

```
2294 pts/11    00:00:00 vi
```

（3）使用 ps -p2294，查看 CPU 的运行时间

```
yuxing@yuxing-VirtualBox:~$ ps -p2294
  PID TTY          TIME CMD
 2294 pts/11    00:00:00 vi
```

（4）ps -lax|grep 2294，查看进程号为 3652 的进程信息

UID：用户号

PID：vi 的进程号

PPID：vi 的父进程号

PRI：内核调度优先级

NI：进程优先级

VSZ：总虚拟内存大小

RSS：进程使用的总物理内存数

STAT：进程状态

TTY：终端的次要装置号码

TIME：使用 cpu 的时间

```
yuxing@yuxing-VirtualBox:~$ ps -lax|grep 2294
0  1000  2294  2260  20   0  67048 10568 poll_s S+   pts/11     0:00 vi
0  1000  2825  2671  20   0  21312  1080 pipe_w S+   pts/1      0:00 grep --colo
r=auto 2294
```

（5）寻找父进程

```
yuxing@yuxing-VirtualBox:~$ ps -lax|grep 2260
0  1000  2260  2255  20   0  29736  4976 wait   Ss   pts/11     0:00 bash
```

```
yuxing@yuxing-VirtualBox:~$ ps -lax|grep 2255
0  1000  2255  1307  20   0 619244 46552 poll_s Sl   ?          0:02 /usr/lib/gn
ome-terminal/gnome-terminal-server
```

```
yuxing@yuxing-VirtualBox:~$ ps -lax|grep 1307
4  1000  1307  1171  20   0  53712  4988 poll_s Ss   ?          0:00 /sbin/upsta
rt --user
```

```
yuxing@yuxing-VirtualBox:~$ ps -lax|grep 1171
4     0  1171   906  20   0 230304  6492 -      Sl   ?          0:00 lightdm --s
ession-child 12 19
```

```
yuxing@yuxing-VirtualBox:~$ ps -lax|grep 906
4     0   906     1  20   0 350412  6192 -      SLsl ?          0:00 /usr/sbin/l
ightdm
```

2294->2260->2255->1307->1171->906->1

(6)使用 pstree -p

```
├─gnome-terminal-(2255)─┬─bash(2260)───vi(2294)
                        ├─bash(2671)───pstree(2912)
                        ├─{dconf worker}(2256)
                        ├─{gdbus}(2258)
                        └─{gmain}(2257)
```

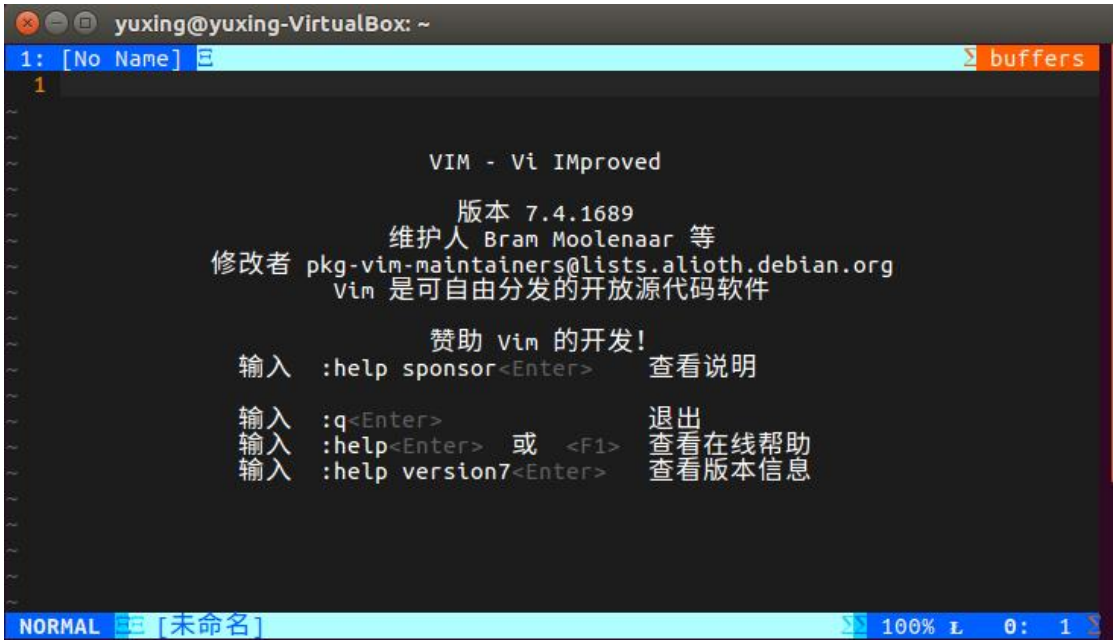
2、编写程序，首先使用 fork 系统调用，创建子进程。在父进程中继续执行空循环操作；在子进程中调用 exec 打开 vi 编辑器。然后在另外一个终端中，通过 ps –Al 命令、ps aux 或者 top 等命令，查看 vi 进程及其父进程的运行状态，理解每个参数所表达的意义。选择合适的命令参数，对所有进程按照 cpu 占用率排序。

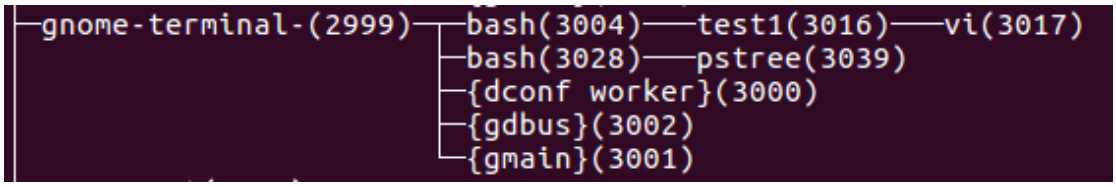
```
#include <unistd.h>

#include <stdio.h>

int main (){

pid_t fpid;

int count=0;

fpid=fork();

if (fpid < 0)

printf("error in fork!");

else if (fpid == 0) {

execl("/usr/bin/vi","vi",NULL);

}

else { for(;;){}
```

}

return 0;

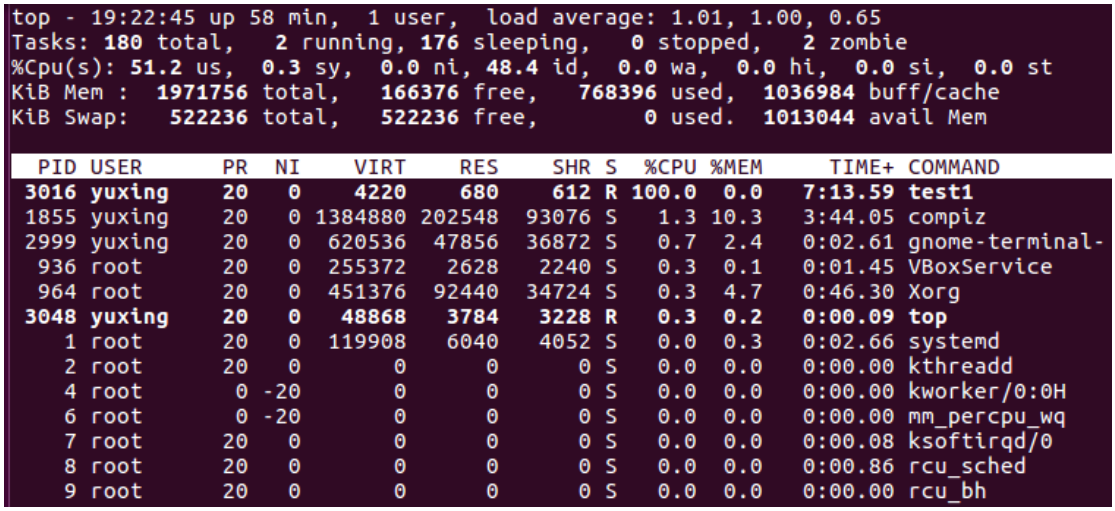
}





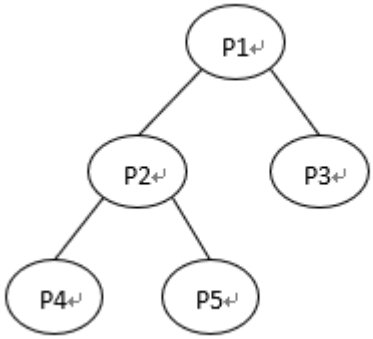
可以看到第一行 vi 是 1 的子程序

使用 top;c 按 cpu 使用排序：



```
top - 19:22:45 up 58 min,  1 user,  load average: 1.01, 1.00, 0.65
Tasks: 180 total,   2 running, 176 sleeping,   0 stopped,   2 zombie
%Cpu(s): 51.2 us,  0.3 sy,  0.0 ni, 48.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1971756 total,   166376 free,   768396 used,  1036984 buff/cache
KiB Swap:   522236 total,   522236 free,        0 used.  1013044 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 3016 yuxing    20   0     4220    680    612 R 100.0  0.0   7:13.59 test1
 1855 yuxing    20   0  1384880 202548  93076 S   1.3 10.3   3:44.05 compiz
 2999 yuxing    20   0   620536  47856  36872 S   0.7  2.4   0:02.61 gnome-terminal-
  936 root      20   0   255372   2628   2240 S   0.3  0.1   0:01.45 VBoxService
  964 root      20   0   451376  92440  34724 S   0.3  4.7   0:46.30 Xorg
 3048 yuxing    20   0    48868   3784   3228 R   0.3  0.2   0:00.09 top
    1 root      20   0   119908   6040   4052 S   0.0  0.3   0:02.66 systemd
    2 root      20   0        0      0      0 S   0.0  0.0   0:00.00 kthreadd
    4 root       0 -20        0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    6 root       0 -20        0      0      0 S   0.0  0.0   0:00.00 mm_percpu_wq
    7 root      20   0        0      0      0 S   0.0  0.0   0:00.08 ksoftirqd/0
    8 root      20   0        0      0      0 S   0.0  0.0   0:00.86 rcu_sched
    9 root      20   0        0      0      0 S   0.0  0.0   0:00.00 rcu_bh
```

3、使用 fork 系统调用，创建如下进程树，并使每个进程输出自己的 ID 和父进程的 ID。观察进程的执行顺序和运行状态的变化。




```
Node p2 is p1's child with pid 3100, it's parent pid 1307.
Node p5 is p2's child with pid 3102, it's parent pid 3100.
Node p4 is p2's child with pid 3101, it's parent pid 3100.
Node p3 is p1's child with pid 3099, it's parent pid 1307.
Node p5 is p2's child with pid 3102, it's parent pid 3100.
Node p4 is p2's child with pid 3101, it's parent pid 3100.
Node p3 is p1's child with pid 3099, it's parent pid 1307.
Node p2 is p1's child with pid 3100, it's parent pid 1307.
Node p4 is p2's child with pid 3101, it's parent pid 3100.
Node p3 is p1's child with pid 3099, it's parent pid 1307.
Node p2 is p1's child with pid 3100, it's parent pid 1307.
Node p5 is p2's child with pid 3102, it's parent pid 3100.
Node p3 is p1's child with pid 3099, it's parent pid 1307.
Node p2 is p1's child with pid 3100, it's parent pid 1307.
Node p5 is p2's child with pid 3102, it's parent pid 3100.
Node p4 is p2's child with pid 3101, it's parent pid 3100.
Node p2 is p1's child with pid 3100, it's parent pid 1307.
Node p5 is p2's child with pid 3102, it's parent pid 3100.
Node p3 is p1's child with pid 3099, it's parent pid 1307.
Node p4 is p2's child with pid 3101, it's parent pid 3100.
Node p5 is p2's child with pid 3102, it's parent pid 3100.
Node p3 is p1's child with pid 3099, it's parent pid 1307.
Node p4 is p2's child with pid 3101, it's parent pid 3100.
Node p2 is p1's child with pid 3100, it's parent pid 1307.
Node p4 is p2's child with pid 3101, it's parent pid 3100.
Node p3 is p1's child with pid 3099, it's parent pid 1307.
Node p2 is p1's child with pid 3100, it's parent pid 1307.
Node p5 is p2's child with pid 3102, it's parent pid 3100.
Node p3 is p1's child with pid 3099, it's parent pid 1307.
Node p2 is p1's child with pid 3100, it's parent pid 1307.
Node p5 is p2's child with pid 3102, it's parent pid 3100.
Node p4 is p2's child with pid 3101, it's parent pid 3100.
Node p2 is p1's child with pid 3100, it's parent pid 1307.
Node p5 is p2's child with pid 3102, it's parent pid 3100.
Node p4 is p2's child with pid 3101, it's parent pid 3100.
Node p3 is p1's child with pid 3099, it's parent pid 1307.
Node p5 is p2's child with pid 3102, it's parent pid 3100.
```

```
├─test2(3099)
├─test2(3100)─┬─test2(3101)
              └─test2(3102)
```

程序名为 test2，可以看到进程树呈上图相同结构。

4、修改上述进程树中的进程，使得所有进程都循环输出自己的 ID 和父进程的 ID。然后终止 p2 进程(分别采用 kill -9 、自己正常退出 exit()、段错误退出)，观察 p1、p3、p4、p5 进程的运行状态和其他相关参数有何改变。

修改程序，使 p2 exit(0);

```
Node p3 is p1's child with pid 3178, it's parent pid 1307.
Node p4 is p2's child with pid 3180, it's parent pid 1307.
Node p4 is p2's child with pid 3180, it's parent pid 1307.
Node p5 is p2's child with pid 3181, it's parent pid 1307.
Node p3 is p1's child with pid 3178, it's parent pid 1307.
Node p5 is p2's child with pid 3181, it's parent pid 1307.
Node p4 is p2's child with pid 3180, it's parent pid 1307.
Node p5 is p2's child with pid 3181, it's parent pid 1307.
Node p3 is p1's child with pid 3178, it's parent pid 1307.
Node p4 is p2's child with pid 3180, it's parent pid 1307.
Node p3 is p1's child with pid 3178, it's parent pid 1307.
Node p5 is p2's child with pid 3181, it's parent pid 1307.
Node p4 is p2's child with pid 3180, it's parent pid 1307.
Node p5 is p2's child with pid 3181, it's parent pid 1307.
Node p4 is p2's child with pid 3180, it's parent pid 1307.
Node p3 is p1's child with pid 3178, it's parent pid 1307.
Node p4 is p2's child with pid 3180, it's parent pid 1307.
Node p3 is p1's child with pid 3178, it's parent pid 1307.
Node p5 is p2's child with pid 3181, it's parent pid 1307.
Node p3 is p1's child with pid 3178, it's parent pid 1307.
Node p5 is p2's child with pid 3181, it's parent pid 1307.
Node p4 is p2's child with pid 3180, it's parent pid 1307.
Node p3 is p1's child with pid 3178, it's parent pid 1307.
```

kill -9 p2:

子进程继续运行，父进程变为 p1，p3 与其无关，继续运行。

```
Node p3 is p1's child with pid 3178, it's parent pid 1307.
Node p4 is p2's child with pid 3180, it's parent pid 1307.
Node p4 is p2's child with pid 3180, it's parent pid 1307.
Node p5 is p2's child with pid 3181, it's parent pid 1307.
Node p3 is p1's child with pid 3178, it's parent pid 1307.
Node p5 is p2's child with pid 3181, it's parent pid 1307.
Node p4 is p2's child with pid 3180, it's parent pid 1307.
Node p5 is p2's child with pid 3181, it's parent pid 1307.
Node p3 is p1's child with pid 3178, it's parent pid 1307.
Node p4 is p2's child with pid 3180, it's parent pid 1307.
Node p3 is p1's child with pid 3178, it's parent pid 1307.
Node p5 is p2's child with pid 3181, it's parent pid 1307.
Node p4 is p2's child with pid 3180, it's parent pid 1307.
Node p5 is p2's child with pid 3181, it's parent pid 1307.
Node p4 is p2's child with pid 3180, it's parent pid 1307.
Node p3 is p1's child with pid 3178, it's parent pid 1307.
```