实验二：进程控制

班级：安全 1601　　姓名：张天悦　　学号：16281153

根据课堂所学内容和基础知识介绍，完成实验题目。

- 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。

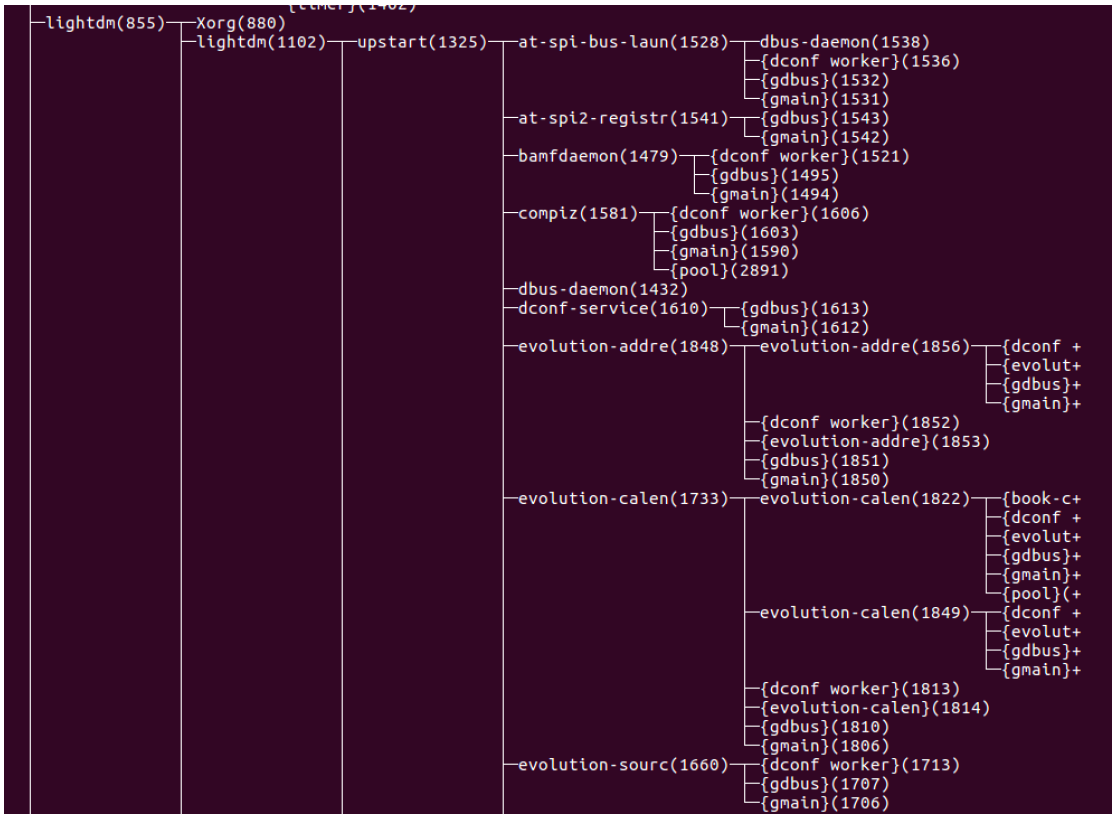
```
phoebe@ubuntu:~$ ps -A | grep vi
   699 ?        00:00:00 VGAuthService
  1575 ?        00:00:03 hud-service
  1610 ?        00:00:00 dconf-service
 65284 pts/1    00:00:00 vi
phoebe@ubuntu:~$ ps -A | grep 65284
 65284 pts/1    00:00:00 vi
phoebe@ubuntu:~$ ps -lax |grep 65284
0  1000  65284  21199  20   0  32040  3700 poll_s S+   pts/1      0:00 vi aa.c
0  1000  65342  65319  20   0  14228  1076 pipe_w S+   pts/4      0:00 grep --color=auto 65284
phoebe@ubuntu:~$ ps -lax |grep 21199
0  1000  21199  21192  20   0  29444  2676 wait   Ss   pts/1      0:00 bash
0  1000  65284  21199  20   0  32040  3700 poll_s S+   pts/1      0:00 vi aa.c
0  1000  65358  65319  20   0  14228  1004 pipe_w S+   pts/4      0:00 grep --color=auto 21199
phoebe@ubuntu:~$ ps -lax |grep 21192
0  1000  21192   1325  20   0 662676 25748 poll_s Sl   ?          0:01 /usr/lib/gnome-terminal/g
terminal-server
0  1000  21199  21192  20   0  29444  2676 wait   Ss   pts/1      0:00 bash
0  1000  65319  21192  20   0  22376  4772 wait   Ss   pts/4      0:00 bash
0  1000  65362  65319  20   0  14228   972 pipe_w S+   pts/4      0:00 grep --color=auto 21192
```

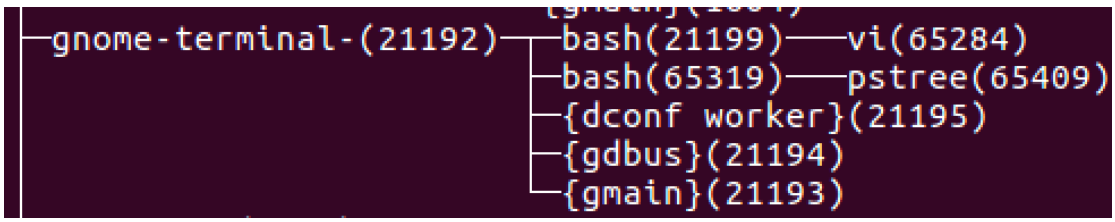
```
phoebe@ubuntu:~$ ps -lax |grep 1325
4  1000   1325   1102  20   0  53544  2236 poll_s Ss   ?
```

```
phoebe@ubuntu:~$ ps -lax |grep 1102
4     0   1102    855  20   0 230396   660 -      Sl   ?          0:00 lightdm --session-child
4  1000   1325   1102  20   0  53544  2236 poll_s Ss   ?          0:00 /sbin/upstart --user
0  1000  65380  65319  20   0  14228   944 pipe_w S+   pts/4      0:00 grep --color=auto 1102
phoebe@ubuntu:~$ ps -lax |grep 855
4     0    699      1  20   0  85524   420 -      Ss   ?          0:00 /usr/bin/VGAuthService
4     0    855      1  20   0 292192   608 -      SLsl ?          0:00 /usr/sbin/lightdm
4     0    880    855  20   0 396356 30964 -      Ss+  tty7       1:03 /usr/lib/xorg/Xorg -cor
 /var/run/lightdm/root/:0 -nolisten tcp vt7 -novtswitch
4     0   1102    855  20   0 230396   660 -      Sl   ?          0:00 lightdm --session-child
0  1000  65382  65319  20   0  14228   940 pipe_w S+   pts/4      0:00 grep --color=auto 855
phoebe@ubuntu:~$ ps -lax |grep 1
4     0      1      0  20   0 185380  4276 -      Ss   ?          0:06 /lib/systemd/systemd --
1
1     0      2      0  20   0      0     0 -      S    ?          0:00 [kthreadd]
```

65284>21199>21192>1325>1102>855>1

pstree:





- 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。

  程序：

```c
#include <unistd.h>
#include <stdio.h>
int main()
{
        pid_t pid;
        int count=0;
        pid = fork ();
        if (pid > 0)
                printf ("error in fork!\n", pid);
        else if (pid==0)
                excel("/usr/bin/vi","vi",NULL);
        else
{
                for (;;){}
}
return 0;
}
```

```
                  VIM - Vi IMproved

                   version 7.4.1689
                by Bram Moolenaar et al.
    Modified by pkg-vim-maintainers@lists.alioth.debian.org
          Vim is open source and freely distributable

                Help poor children in Uganda!
        type  :help iccf<Enter>       for information

        type  :q<Enter>               to exit
        type  :help<Enter>  or  <F1>  for on-line help
        type  :help version7<Enter>   for version info
```

查看父进程

```
phoebe@ubuntu:~$ ps -A | grep vi
  846 ?        00:00:00 VGAuthService
 1915 ?        00:00:00 hud-service
 1940 ?        00:00:00 dconf-service
 2783 pts/17   00:00:00 vi
phoebe@ubuntu:~$ ps -A |grep 2783
 2783 pts/17   00:00:00 vi
```

pstree

```
                        {gmain}
  ├─gnome-terminal-─┬─bash
                    ├─bash───pstree
```

按 cpu 占用率排序

```
phoebe@ubuntu:~$ ps -aux | sort -k3,3nr
phoebe      1923  0.3  7.5 1181160 74912 ?       Ssl  20:08   0:08 compiz
root         982  0.3  4.9 382572 49660 tty7     Ss+  19:58   0:08 /usr/lib/xorg/Xorg
re :0 -seat seat0 -auth /var/run/lightdm/root/:0 -nolisten tcp vt7 -novtswitch
phoebe      2443  0.2  9.7 712444 97364 ?        SNl  20:09   0:04 /usr/bin/python3 /u
bin/update-manager --no-update --no-focus-on-map
phoebe      2086  0.1  4.4 867740 44068 ?        Sl   20:08   0:02 nautilus -n
phoebe      2089  0.1  1.7 535272 17664 ?        Sl   20:08   0:02 /usr/bin/vmtoolsd -
musr --blockFd 3
phoebe      2348  0.1  3.2 661072 32308 ?        Sl   20:08   0:03 /usr/lib/gnome-term
l/gnome-terminal-server
root           1  0.1  0.4 185312  4532 ?        Ss   19:58   0:02 /sbin/init auto nop
ot
root         388  0.1  0.4 190108  4572 ?        Ssl  19:58   0:02 /usr/bin/vmtoolsd
avahi        827  0.0  0.2  44912  2464 ?        Ss   19:58   0:00 avahi-daemon: runni
[ubuntu.local]
avahi        858  0.0  0.0  44784    20 ?        S    19:58   0:00 avahi-daemon: chroo
```

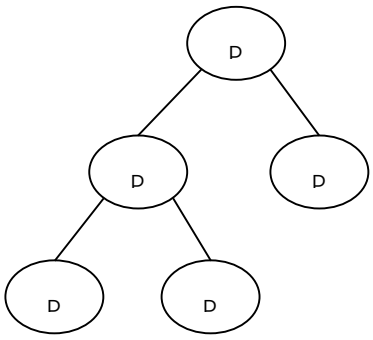
top 命令

```
top - 20:30:56 up 32 min,  1 user,  load average: 0.08, 0.04, 0.05
Tasks: 248 total,   1 running, 247 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.3 us,  0.0 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :   998468 total,    58348 free,   709016 used,   231104 buff/cache
KiB Swap:  1046524 total,   854636 free,   191888 used.   221972 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 2089 phoebe    20   0  535272  17128  12220 S  0.3  1.7   0:01.76 vmtoolsd
 2742 root      20   0       0      0      0 S  0.3  0.0   0:00.11 kworker/u2+
 2847 phoebe    20   0   41904   3704   2996 R  0.3  0.4   0:00.20 top
    1 root      20   0  185312   4532   2988 S  0.0  0.5   0:02.71 systemd
```

- 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。



```c
#include "unistd.h"
#include "stdlib.h"

#define HASPRO -10

int main()
{
    pid_t p1,p2,p3,p4,p5;

    int cnt=0;

    while((p1=fork()) == -1)
        sleep(0.1);

    if(!p1)
    {
        while((p2=fork()) == -1)
                sleep(0.1);

        if(!p2)
        {
            while ((p4=fork())==-1);
            if (!p4)
            {
                while(1)
                {
                    printf("Node p4 is p2's child with pid %d, it's parent pid %d.\n",getpid(),getppid());
                    sleep(0.1);
                }
            }
            else
            {
                while ((p5=fork())==-1)
                        sleep(0.1);

                if (!p5)
                {
                    while(1)
                    {
                        printf("Node p5 is p2's child with pid %d, it's parent pid %d.\n",getpid(),getppid());
                        sleep(0.1);
                    }

                }

            }
```

C ▾ Tab Width

```
Node p4 is p2's child with pid 3115, it's parent pid 3114.
Node p5 is p2's child with pid 3116, it's parent pid 3114.
Node p3 is p1's child with pid 3113, it's parent pid 1654.
Node p2 is p1's child with pid 3114, it's parent pid 1654.
Node p5 is p2's child with pid 3116, it's parent pid 3114.
Node p3 is p1's child with pid 3113, it's parent pid 1654.
Node p2 is p1's child with pid 3114, it's parent pid 1654.
Node p4 is p2's child with pid 3115, it's parent pid 3114.
Node p3 is p1's child with pid 3113, it's parent pid 1654.
Node p2 is p1's child with pid 3114, it's parent pid 1654.
Node p4 is p2's child with pid 3115, it's parent pid 3114.
Node p5 is p2's child with pid 3116, it's parent pid 3114.
Node p2 is p1's child with pid 3114, it's parent pid 1654.
Node p4 is p2's child with pid 3115, it's parent pid 3114.
Node p5 is p2's child with pid 3116, it's parent pid 3114.
Node p3 is p1's child with pid 3113, it's parent pid 1654.
Node p5 is p2's child with pid 3116, it's parent pid 3114.
```

- 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。

```
Node p3 is p1's child with pid 3149, it's parent pid 1654.
Node p5 is p2's child with pid 3152, it's parent pid 1654.
Node p3 is p1's child with pid 3149, it's parent pid 1654.
Node p4 is p2's child with pid 3151, it's parent pid 1654.
Node p3 is p1's child with pid 3149, it's parent pid 1654.
Node p4 is p2's child with pid 3151, it's parent pid 1654.
Node p5 is p2's child with pid 3152, it's parent pid 1654.
Node p5 is p2's child with pid 3152, it's parent pid 1654.
Node p4 is p2's child with pid 3151, it's parent pid 1654.
Node p3 is p1's child with pid 3149, it's parent pid 1654.
Node p4 is p2's child with pid 3151, it's parent pid 1654.
Node p3 is p1's child with pid 3149, it's parent pid 1654.
Node p5 is p2's child with pid 3152, it's parent pid 1654.
Node p3 is p1's child with pid 3149, it's parent pid 1654.
Node p5 is p2's child with pid 3152, it's parent pid 1654.
Node p4 is p2's child with pid 3151, it's parent pid 1654.
Node p5 is p2's child with pid 3152, it's parent pid 1654.
Node p4 is p2's child with pid 3151, it's parent pid 1654.
Node p3 is p1's child with pid 3149, it's parent pid 1654.
Node p4 is p2's child with pid 3151, it's parent pid 1654.
```

父进程 p1，子进程 p4 p5 继续运行，p3 与其无关。

```
Node p3 is p1's child with pid 4955, it's parent pid 4290.
Node p5 is p2's child with pid 4958, it's parent pid 4290.
Node p4 is p2's child with pid 4957, it's parent pid 4290.
Node p5 is p2's child with pid 4958, it's parent pid 4290.
Node p3 is p1's child with pid 4955, it's parent pid 4290.
Node p5 is p2's child with pid 4958, it's parent pid 4290.
Node p3 is p1's child with pid 4955, it's parent pid 4290.
Node p4 is p2's child with pid 4957, it's parent pid 4290.
Node p3 is p1's child with pid 4955, it's parent pid 4290.
Node p5 is p2's child with pid 4958, it's parent pid 4290.
Node p3 is p1's child with pid 4955, it's parent pid 4290.
Node p5 is p2's child with pid 4958, it's parent pid 4290.
Node p4 is p2's child with pid 4957, it's parent pid 4290.
Node p5 is p2's child with pid 4958, it's parent pid 4290.
Node p3 is p1's child with pid 4955, it's parent pid 4290.
Node p4 is p2's child with pid 4957, it's parent pid 4290.
Node p5 is p2's child with pid 4958, it's parent pid 4290.
Node p3 is p1's child with pid 4955, it's parent pid 4290.
Node p4 is p2's child with pid 4957, it's parent pid 4290.
Node p3 is p1's child with pid 4955, it's parent pid 4290.
Node p5 is p2's child with pid 4958, it's parent pid 4290.
```