# 进程控制实验报告

安全 1601 沈香港 16281077

#### task1

(1) 打开一个新的终端,输入 vi,按回车,开启一个新的进程,显示结果如下

```
Selection Shen@shen-Lenovo-Rescuer-15ISK: ~

VIM - Vi IMproved

版本 7.4.1689
维护人 Bram Moolenaar 等
修改者 pkg-vim-maintainers@lists.alioth.debian.org
Vim 是可自由分发的开放源代码软件

帮助乌干达的可怜儿童!

帮助乌干达的可怜儿童!
输入 :help iccf<Enter> 查看说明
输入 :q<Enter> 退出
输入 :help<Enter> 或 <F1> 查看在线帮助
输入 :help version7<Enter> 查看版本信息
```

(2) 打开另一个终端,用于 ps 命令。先找到 vi 进程号,然后只显示名字为 vi 的进程。

通过命令 ps -A 查找 vi 进程的 id 为 4953

```
4935 ? 00:00:00 kworker/3:3

4935 ? 00:00:00 gnome-termin

4940 pts/1 00:00:00 bash

4953 pts/1 00:00:00 vi

5014 pts/6 00:00:00 bash
```

接下来通过命令 ps -ef|grep 4953 只显示 vi 进程

(3) 利用 grey 命令搜索其父进程,直到找到进程号为1为止(重新打开 vi,进程号为5377)

```
shen@shen-Lenovo-Rescuer-15ISK: ~
shen@shen-Lenovo-Rescuer-15ISK:~$ ps -ef |grep 5377
           5377 5365 0 11:37 pts/1
shen
                                          00:00:00 vi vi.txt
           5626 5613 0 11:45 pts/6
shen
                                          00:00:00 grep --color=auto 5377
shen@shen-Lenovo-Rescuer-15ISK:~$ ps -ef |grep 5365
shen 5365 5360 0 11:37 pts/1 00:00:00 bas
                                          00:00:00 bash
shen
           5377
                        0 11:37 pts/1
                                           00:00:00 vi vi.txt
           5631 5613 0 11:46 pts/6
                                          00:00:00 grep --color=auto 5365
shen
shen@shen-Lenovo-Rescuer-15ISK:~$ ps -ef |grep 5360
shen
                3785 0 11:37 ?
                                           00:00:03 /usr/lib/gnome-terminal/gnome-te
rminal-server
                5360 0 11:37 pts/1
5360 0 11:45 pts/6
shen
           5365
                                           00:00:00 bash
shen
           5613
                                           00:00:00 bash
           5634 5613 0 11:46 pts/6
                                           00:00:00 grep --color=auto 5360
lshen
shen@shen-Lenovo-Rescuer-15ISK:~$
```

```
shen@shen-Lenovo-Rescuer-15ISK:~$ ps -ef |grep 3785
shen 3785 3660 0 11:14 ? 00:00:00 /sbin/upstart --user
```

```
😰 🖨 📵 shen@shen-Lenovo-Rescuer-15ISK: ~
shen@shen-Lenovo-Rescuer-15ISK:~$ ps -ef |grep 3660
                   982 0 11:14 ?
root
                                             00:00:00 lightdm --session-child 10 16
                                             00:00:00 /sbin/upstart --user
shen
           3785
                         0 11:14 ?
                 5726
                        0 11:47 pts/6
                                             00:00:00 grep --color=auto 36
shen
           5739
shen@shen-Lenovo-Rescuer-15ISK:~$ ps -ef |grep 982
                                             00:00:00 /usr/sbin/lightdm
00:01:52 /usr/lib/xorg/Xorg -core :0 -sea
                     1 0 10:29 ?
root
           3554
                        5 11:14 tty7
root
t seat0 -auth /var/run/lightdm/root/:0 -nolisten tcp vt7 -novtswitch root 3660 982 0 11:14 ? 00:00:00 lightdm --session-child 10 16
           5741
                 5726 0 11:47 pts/6
shen
                                             00:00:00 grep --color=auto 982
shen@shen-Lenovo-Rescuer-15ISK:~$
```

可以看到搜索父进程顺序为:  $5377 \rightarrow 5365 \rightarrow 5360 \rightarrow 3785 \rightarrow 3660 \rightarrow 982 \rightarrow 1$  输入命令 pstree -p 查看进程树

```
| Comparison | Com
```

发现两种方式所得结果一样。(截屏不完整,但数据真实)

#### Task2

(1) 编译并运行所编写程序

```
❷●□ shen@shen-Lenovo-Rescuer-15ISK: ~/操作系统/实验2
shen@shen-Lenovo-Rescuer-15ISK: ~/操作系统/实验2$ vim q2.c
shen@shen-Lenovo-Rescuer-15ISK: ~/操作系统/实验2$ gcc q2.c -o q2
shen@shen-Lenovo-Rescuer-15ISK: ~/操作系统/实验2$ ./q2
```

```
● ■ shen@shen-Lenovo-Rescuer-15ISK: ~/操作系统/实验2
shen@shen-Lenovo-Rescuer-15ISK: ~/操作系统/实验2$ vim q2.c
shen@shen-Lenovo-Rescuer-15ISK: ~/操作系统/实验2$ gcc q2.c -o q2
shen@shen-Lenovo-Rescuer-15ISK: ~/操作系统/实验2$ ./q2
子进程的pid为6739
父进程的pid为6738
```

### (2) 查看 vi 进程及其父进程的运行状态

#### vi 讲程

```
● ■ shen@shen-Lenovo-Rescuer-15ISK: ~/操作系统/实验2
shen@shen-Lenovo-Rescuer-15ISK: ~/操作系统/实验2$ ps -aux |grep -w vi
shen 6875 0.0 0.0 15984 972 pts/1 S+ 15:25 0:00 grep --color=au
to -w vi
shen@shen-Lenovo-Rescuer-15ISK: ~/操作系统/实验2$ ps -ef |grep -w vi
shen 6877 6835 0 15:25 pts/1 00:00:00 grep --color=auto -w vi
shen@shen-Lenovo-Rescuer-15ISK: ~/操作系统/实验2$
```

 shen@shen-Lenovo-Rescuer-15ISK:~/操作系统/实验2\$ ps -ef |grep -w vi

 shen
 6961
 6941
 0 15:29 pts/1
 00:00:00 grep --color=auto -w vi

 可以看到 vi 进程的进程号为 6961,父进程的进程号为 6941.

## vi 父进程

```
shen@shen-Lenovo-Rescuer-15ISK:~/操作糸统/实验2$ ps -aux |grep -w 6941
shen
               0.0 0.0
                        24436
                                5144 pts/1
                                             Ss
                                                  15:28
                                                          0:00 bash
shen
         6988 0.0
                  0.0
                        15984
                               1016 pts/1
                                             R+
                                                  15:32
                                                          0:00 grep --color=au
to -w 6941
```

## 对所有进程按照 cpu 占用率排序

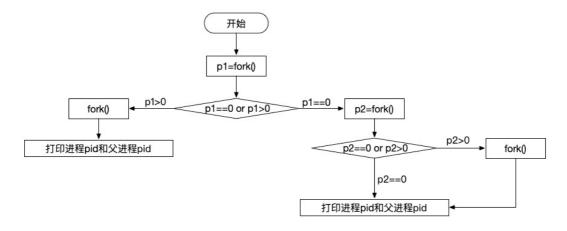
```
7.7%]
6.0%]
11.9%]
11.6%]
2.91G/11.6G]
0K/11.9G]
                                           11.4%
                                                                       7.2%]
4.8%]
7.5%]
Mem[||||||||||
                                        Tasks: 111, 333 thr; 2 running
                                        Load average: 0.50 0.41 0.32
 Swp
                                        Uptime: 05:22:20
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+
                                                   5.2 10:42.24 /usr/lib/firefox/
4726 shen
                     0 2738M
                                    244M S 21.6
                               131M 77880 S 16.2
4249 shen
                20
                     0 1286M
                                                  1.1
                                                        3:23.98 compiz
1315 shen
                     0 422M 22704 19224 S 14.9
                20
                                                  0.2 0:00.30 gnome-screenshot
                     0 556M
                20
                                    129M S 6.1
                                                        0:45.40 /usr/lib/xorg/Xor
3578 root
                                                  1.4
4563 shen
                20
                     0 2741M
                                    249M S 2.0
                                                  4.0
                                                        5:05.34 /usr/lib/firefox/
4737 shen
                20
                     0 2738M
                                     244M S 2.0
                                                  5.2
                                                        0:02.39 /usr/lib/firefox,
1313 shen
                20
                                             1.4
                                                        0:00.14 htop
                     0 27892
                               3848
                                     3204 R
                                                  0.0
6881 shen
                20
                     0 1196M 45236 36012 S
                                             1.4
                                                  0.4
                                                        0:12.76 gnome-screenshot
4746 shen
                20
                     0 2738M 616M
                                    244M S
                                             1.4
                                                  5.2
                                                        0:02.43 /usr/lib/firefox
                                     244M S
4747 shen
                20
                     0 2738M
                                             1.4
                                                  5.2
                                                        0:02.29 /usr/lib/firefox
4600 shen
                20
                               471M
                                     249M S
                                             0.7
                                                  4.0
                                                        0:16.04 /usr/lib/firefox
                     0 2741M
                                     249M S
4603 shen
                20
                     0
                               471M
                                             0.7
                                                   4.0
                                                        0:14.19
```

## task3

```
(1) 实验代码(以下有参考孙汉武同学实验)
#include<stdio.h>
#include<unistd.h>
int main()
       pid_t p1, p2;
       p1=fork();//
       if(p1<0)
       {
               printf("我是进程%d,在创建子进程的过程中出错\n! ",getpid());
       if(p1==0)
       {
               p2=fork();
               if(p2>0)
                      fork();
               pid_t p,pp;
               p=getpid();
               pp=getppid();//给变量赋值主要是为了调试方便
               printf("我的进程号是%d,我的父进程号是%d\n",p,pp);
               sleep(1);
       if(p1>0)
               fork();
               pid_t p,pp;
               p=getpid();
               pp=getppid();
               printf("我的进程号是%d、我的父进程号是%d\n",p,pp);
               sleep(1);
       }
       return 0;
}
```

#### (2) 代码解析:

fork 函数的功能:执行后创建两个几乎完全相同的进程,一个父进程一个子进程。在子进程中,fork 函数返回 0,在父进程中,fork 返回新创建子进程的进程 id,返回负值则说明创建失败。这两个进程没有固定的先后顺序,哪一个先执行要看系统的进程调度策略。程序流程图如下:



## (3) 程序解释:

结合上面的程序流程图,可以看到程序执行最初的那个进程就是父进程 A,在一开始就通过 fork 函数创建一个子进程 B,并用 p1 接受 fork 的返回值,由于 p1 在进程 A 和进程 B 的值不同,所以两个进程接下来执行的代码部分并不相同。进程 A 执行 p1>0 的部分代码,并且在 p1>0 的那部分代码中,进程 A 再次通过 fork 创建了一个子进程 C。随后进程 A 和进程 C 分别打印自己的 pid 和 ppid(父进程 id)。在来看进程 B,在 p1==0 的那部分代码中,进程 B 也是通过 fork 创建了一个子进程 D,并将 fork 的返回值交给 p2,在父进程 B 中,p2 的值大于 0,所以父进程 B 还创建了一个子进程 E,而在子进程 D 中,p2==0,所以不执行 p2>0 代码块里面的语句,直接打印 pid 和 ppid,之后分别是进程 E 和 B 打印 pid 和 ppid。到此满足实验要求的进程树创建完毕。

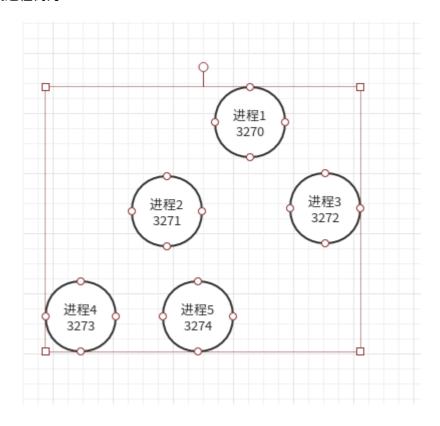
## (4) 编译执行

```
shen@shen-Lenovo-Rescuer-15ISK:~/操作系统/实验2$ vim q3.c
shen@shen-Lenovo-Rescuer-15ISK:~/操作系统/实验2$ gcc q3.c -o q3
shen@shen-Lenovo-Rescuer-15ISK:~/操作系统/实验2$ ./q3
```

### 执行结果

```
shen@shen-Lenovo-Rescuer-15ISK:~/操作系统/实验2$ ./q3
我的进程号是3270,我的父进程号是3216
我的进程号是3272,我的父进程号是3270
我的进程号是3271,我的父进程号是3270
我的进程号是3274,我的父进程号是3271
我的进程号是3273,我的父进程号是3271
```

#### (5) 可以得到其进程树为



# task4(原始代码为 q4.c)

(1) kill -9

```
我的进程号是4636,我的父进程号是4623
我的进程号是4640,我的父进程号是4637
我的进程号是4639,我的父进程号是4637
我的进程号是4638,我的父进程号是4636
我的进程号是4636,我的父进程号是4623
我的进程号是4637,我的父进程号是4636
我的进程号是4637,我的父进程号是4636
我的进程号是4638,我的父进程号是4636
```

执行完 kill -9 4637 (也就是 p2 后) 的结果

```
我的进程号是4640,我的父进程号是4637
我的进程号是4638,我的父进程号是4636
我的进程号是4636,我的父进程号是4623
我的进程号是4639,我的父进程号是4637
我的进程号是4636,我的父进程号是4623
我的进程号是4638,我的父进程号是4636
我的进程号是4640,我的父进程号是4637
```

发现删除之 p2 的子进程 p4 和 p5 挂载 init 上面。然后进程 p1,p2,p3 还是原来的进程,进程 p2 虽然被杀死,但是变成 Z+的状态,成为退出状态进程的僵尸进程。 同时发现进程 p2 的输出已经没有了

(2) 正常退出(执行代码为 q4\_2.c)

```
找的进程号是4916,我的父进程号是4903
我的进程号是4918,我的父进程号是4916
我的进程号是4916,我的父进程号是4903
我的进程号是4916,我的父进程号是4903
我的进程号是4918,我的父进程号是4903
我的进程号是4916,我的父进程号是4903
我的进程号是4918,我的父进程号是4903
允0,^A^[^A^[^A^]
```

可以看到 p2(进程 4917)在执行十次以后输出就没有了

(3)段错误退出(执行代码为  $q4_3.c$ ) 在进程 p2 的进程段里面定义一个野指针,野指针没有初始化会产生段错误导致进程退出。

```
我的进程号是5076,我的父进程号是5032
我的进程号是5078,我的父进程号是5076
我的进程号是5077,我的父进程号是5076
我的进程号是5079,我的父进程号是5077
我的进程号是5080,我的父进程号是5077
```

```
我的进程号是5079,我的父进程号是5077
我的进程号是5076,我的父进程号是5032
我的进程号是5078,我的父进程号是5076
我的进程号是5079,我的父进程号是5077
我的进程号是5078,我的父进程号是5077
我的进程号是5076,我的父进程号是5032
我的进程号是5079,我的父进程号是5077
我的进程号是5076,我的父进程号是5077
```

# 看出进程 2(5077)在执行一段时间后结束

综上所述,三种方式的终止进程都会将进程变成僵尸进程,进程在退出的过程中,系统回收资源,除了task\_struct 结构(以及少数资源)以外。于是进程就只剩下 task\_struct 这么个空壳,故称为僵尸。之所以保留 task\_struct,是因为 task\_struct 里面保存了进程的退出码、以及一些统计信息。而其父进程很可能会关心这些信息。这也是为什么进程 p2 终止之后在进程树中还能看到进程 p2 的信息等。