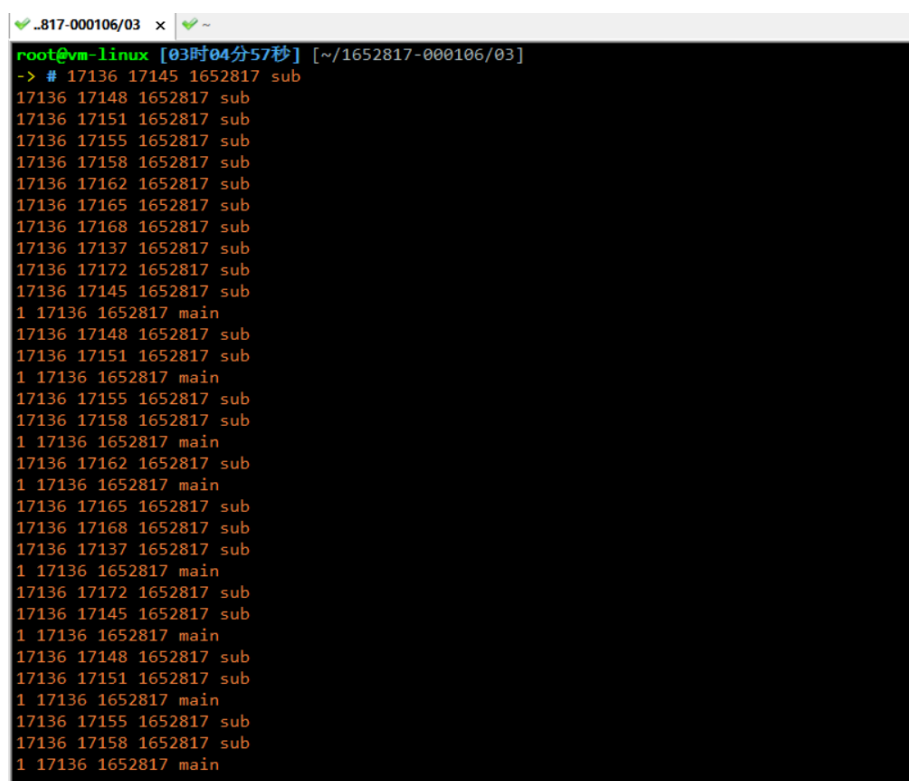


第 00 章作业 - Linux 知识补充 - 守护进程的编写及使用方法 03

1652817 钟钰琛 计算机科学与技术

2018 年 10 月 17 日

1. 执行 test3-1.c 执行到子进程全部结束



```
root@vm-linux [03时04分57秒] [~/1652817-000106/03]
-> # 17136 17145 1652817 sub
17136 17148 1652817 sub
17136 17151 1652817 sub
17136 17155 1652817 sub
17136 17158 1652817 sub
17136 17162 1652817 sub
17136 17165 1652817 sub
17136 17168 1652817 sub
17136 17137 1652817 sub
17136 17172 1652817 sub
17136 17145 1652817 sub
1 17136 1652817 main
17136 17148 1652817 sub
17136 17151 1652817 sub
1 17136 1652817 main
17136 17155 1652817 sub
17136 17158 1652817 sub
1 17136 1652817 main
17136 17162 1652817 sub
1 17136 1652817 main
17136 17165 1652817 sub
17136 17168 1652817 sub
17136 17137 1652817 sub
1 17136 1652817 main
17136 17172 1652817 sub
17136 17145 1652817 sub
1 17136 1652817 main
17136 17148 1652817 sub
17136 17151 1652817 sub
1 17136 1652817 main
17136 17155 1652817 sub
17136 17158 1652817 sub
1 17136 1652817 main
```

图 1: 执行 test3-1

```
..817-000106/03 x ~
17136 17155 1652817 sub
17136 17158 1652817 sub
1 17136 1652817 main
17136 17162 1652817 sub
1 17136 1652817 main
17136 17165 1652817 sub
17136 17168 1652817 sub
17136 17137 1652817 sub
1 17136 1652817 main
17136 17172 1652817 sub
17136 17145 1652817 sub
1 17136 1652817 main
17136 17148 1652817 sub
17136 17151 1652817 sub
1 17136 1652817 main
17136 17155 1652817 sub
17136 17158 1652817 sub
1 17136 1652817 main
17136 17162 1652817 sub
1 17136 1652817 main
17136 17165 1652817 sub
17136 17168 1652817 sub
1 17136 1652817 main
17136 17172 1652817 sub
1 17136 1652817 main
1 17136 1652817 main
1 17136 1652817 main
1 17136 1652817 main
1 17136 1652817 main
1 17136 1652817 main
1 17136 1652817 main
1 17136 1652817 main
1 17136 1652817 main
1 17136 1652817 main
```

子进程全部结束，而父进程未结束

图 2: 执行 test3-1

2. 关系: 子进程的父进程的 id 就是守护进程的 id
3. 这时候在另一个控制台查看僵尸进程 ('Z'). 如果杀死父进程, 那么僵尸进程将会被 init 进程收尸

```
root@vm-linux [03时07分41秒] [~]
-> # ps aux|grep -w 'Z'
root    17137  0.0  0.0      0  0 ?      Z   03:04   0:00 [test3-1] <defunct>
root    17145  0.0  0.0      0  0 ?      Z   03:04   0:00 [test3-1] <defunct>
root    17148  0.0  0.0      0  0 ?      Z   03:05   0:00 [test3-1] <defunct>
root    17151  0.0  0.0      0  0 ?      Z   03:05   0:00 [test3-1] <defunct>
root    17155  0.0  0.0      0  0 ?      Z   03:05   0:00 [test3-1] <defunct>
root    17158  0.0  0.0      0  0 ?      Z   03:05   0:00 [test3-1] <defunct>
root    17162  0.0  0.0      0  0 ?      Z   03:05   0:00 [test3-1] <defunct>
root    17165  0.0  0.0      0  0 ?      Z   03:05   0:00 [test3-1] <defunct>
root    17168  0.0  0.0      0  0 ?      Z   03:05   0:00 [test3-1] <defunct>
root    17172  0.0  0.0      0  0 ?      Z   03:05   0:00 [test3-1] <defunct>
root    17303  0.0  0.0 114884 1036 pts/0    S+  03:07   0:00 grep --color=auto --excl
lude-dir=.bzip --exclude-dir=CVS --exclude-dir=.git --exclude-dir=.hg --exclude-dir=.svn -
w Z
root@vm-linux [03时07分42秒] [~]
-> # kill -9 17136
root@vm-linux [03时08分02秒] [~]
-> # ps aux|grep -w 'Z'
root    17332  0.0  0.0 114880 1036 pts/0    S+  03:08   0:00 grep --color=auto --excl
ude-dir=.bzip --exclude-dir=CVS --exclude-dir=.git --exclude-dir=.hg --exclude-dir=.svn -
w Z
root@vm-linux [03时08分06秒] [~]
-> #
```

图 3: 僵尸进程

僵尸进程：父进程调用 fork 创建子进程后，子进程运行直至其终止，它立即从内存中移除，但进程描述符仍然保留在内存中（进程描述符占有极少的内存空间）。子进程的状态变成 EXIT_ZOMBIE，并且向父进程发开 SIGCHLD 信号，父进程此时应该调用 wait() 系统调用来获取子进程的退出状态以及其它的信息。在 wait 调用之后，僵尸进程就完全从内存中移除。因此一个僵尸存在于其终止到父进程调用 wait 等函数这个时间的间隙，一般很快就消失，但如果编程不合理，父进程从不调用 wait 等系统调用来收集僵尸进程，那么这些进程会一直存在内存中

4. 杀死僵尸进程的办法就是杀死其父进程，这样僵尸进程就会被 1 号进程收尸

5. test3-2.c: 不产生僵尸进程

最简单的方法就是加一句:

```

11     if(pid < 0){
12         exit(EXIT_FAILURE);
13     }
14
15     // exit the parent process
16     if(pid > 0){
17         exit(EXIT_SUCCESS);
18     }
19
20     // create a new sid for the child process
21     pid_t sid = setsid();
22     if (sid < 0){
23         exit(EXIT_FAILURE);
24     }
25     signal(SIGCHLD, SIG_IGN);
26
27     pid = fork();
28
29     if(pid < 0){
30         exit(EXIT_FAILURE);
31     }
32
33     if(pid > 0){
34         exit(EXIT_SUCCESS);
35     }
36
37     // set new file permissions
38     umask(0);
39
40     // change the current working directory
41     if((chdir("/") < 0){
42         exit(EXIT_FAILURE);
43     }
44
-- 插入 --
26,27-34 24%

```

图 4: signal(SIGCHLD, SIG_IGN)

这句话通知父进程对子进程的结束不关心，忽略子进程结束发送的 SIGCHLD 信号。子进程由内核来回收。

实验表明确实没有僵尸进程

(kill 是在另一个控制台执行的.)

```
root@vm-linux [03时32分27秒] [~/1652817-000106/03]
-> # ps -ef|grep test3-2
root      18462      1  0 03:29 ?        00:00:00 ./test3-2
root      18690  14260  0 03:32 pts/1    00:00:00 grep --color=auto --exclude-dir=.bzip --exclude-dir=CVS --exclude-dir=.git --exclude-dir=.hg --exclude-dir=.svn test3-2
root@vm-linux [03时32分31秒] [~/1652817-000106/03]
-> # ps aux|grep -w 'Z'
root      18699  0.0  0.0 114880  1036 pts/1    S+   03:32   0:00 grep --color=auto --exclude-dir=.bzip --exclude-dir=CVS --exclude-dir=.git --exclude-dir=.hg --exclude-dir=.svn -w Z
root@vm-linux [03时32分35秒] [~/1652817-000106/03]
-> #
```

图 5: 杀死父进程后，没有产生僵尸进程