

第 00 章作业 - Linux 知识补充 - 守护进程的编写及使用方法 05

1652817 钟钰琛 计算机科学与技术

2018 年 10 月 17 日

1. 512MB 的时候，分裂了 3347 个进程

```
root@vm-linux [14时11分39秒] [~/1652817-000106/05]
-> # ./test5-1 10000
已分裂0个子进程
root@vm-linux [14时11分42秒] [~/1652817-000106/05]
-> # 已分裂200个子进程
已分裂400个子进程
已分裂600个子进程
已分裂800个子进程
已分裂1000个子进程
已分裂1200个子进程
已分裂1400个子进程
已分裂1600个子进程
已分裂1800个子进程
已分裂2000个子进程
已分裂2200个子进程
已分裂2400个子进程
已分裂2600个子进程
已分裂2800个子进程
已分裂3000个子进程
已分裂3200个子进程
成功分裂3347个子进程
bash kill_sh.sh
kill_sh.sh: fork: retry: 资源暂时不可用
kill_sh.sh: fork: retry: 资源暂时不可用
kill_sh.sh: fork: retry: 资源暂时不可用
kill_sh.sh: fork: retry: 资源暂时不可用
^C
kill_sh.sh: fork: 资源暂时不可用
```

想杀掉这些进程(我写的一个批kill shell)无法启动，因为没法再开启新的进程

图 1: 512MB 的情况

而后我想 kill 掉这些进程，但是因为不能 fork 新的进程而不能 kill! 所

以现在 linux 什么事也做不了!

2. 1024MB 的时候，分裂了 7737 个进程

```
已分裂800个子进程  
已分裂1000个子进程  
已分裂1200个子进程  
已分裂1400个子进程  
已分裂1600个子进程  
已分裂1800个子进程  
已分裂2000个子进程  
已分裂2200个子进程  
已分裂2400个子进程  
已分裂2600个子进程  
已分裂2800个子进程  
已分裂3000个子进程  
已分裂3200个子进程  
已分裂3400个子进程  
已分裂3600个子进程  
已分裂3800个子进程  
已分裂4000个子进程  
已分裂4200个子进程  
已分裂4400个子进程  
已分裂4600个子进程  
已分裂4800个子进程  
已分裂5000个子进程  
已分裂5200个子进程  
已分裂5400个子进程  
已分裂5600个子进程  
已分裂5800个子进程  
已分裂6000个子进程  
已分裂6200个子进程  
已分裂6400个子进程  
已分裂6600个子进程  
已分裂6800个子进程  
已分裂7000个子进程  
已分裂7200个子进程  
成功分裂7377个子进程
```

图 2: 1024MB 的情况

3. 2048MB 的时候，分裂了 14154 个进程

```
已分裂7600个子进程
已分裂7800个子进程
已分裂8000个子进程
已分裂8200个子进程
已分裂8400个子进程
已分裂8600个子进程
已分裂8800个子进程
已分裂9000个子进程
已分裂9200个子进程
已分裂9400个子进程
已分裂9600个子进程
已分裂9800个子进程
已分裂10000个子进程
已分裂10200个子进程
已分裂10400个子进程
已分裂10600个子进程
已分裂10800个子进程
已分裂11000个子进程
已分裂11200个子进程
已分裂11400个子进程
已分裂11600个子进程
已分裂11800个子进程
已分裂12000个子进程
已分裂12200个子进程
已分裂12400个子进程
已分裂12600个子进程
已分裂12800个子进程
已分裂13000个子进程
已分裂13200个子进程
已分裂13400个子进程
已分裂13600个子进程
已分裂13800个子进程
已分裂14000个子进程
成功分裂14154个子进程
```

图 3: 2048MB 的情况

从上面三个实验来看, 分裂的进程和内存大小基本服从线性关系.

4. 修改为 `char str[1024*10]`, 分别测试 512MB, 1024MB, 2048MB 下和之前相比, 分裂的进程数几乎没有变化, 只是略微减少了一些.

```
root@vm-linux [14时28分51秒] [~/1652817-000106/05]
-> # make
gcc -o test5-1 test5-1.c
root@vm-linux [14时28分52秒] [~/1652817-000106/05]
-> # ./test5-1 10000
已分裂0个子进程
root@vm-linux [14时28分54秒] [~/1652817-000106/05]
-> # 已分裂200个子进程
已分裂400个子进程
已分裂600个子进程
已分裂800个子进程
已分裂1000个子进程
已分裂1200个子进程
已分裂1400个子进程
已分裂1600个子进程
已分裂1800个子进程
已分裂2000个子进程
已分裂2200个子进程
已分裂2400个子进程
已分裂2600个子进程
已分裂2800个子进程
已分裂3000个子进程
已分裂3200个子进程
成功分裂3346个子进程
```

图 4: 512MB 的情况

```
已分裂800个子进程
已分裂1000个子进程
已分裂1200个子进程
已分裂1400个子进程
已分裂1600个子进程
已分裂1800个子进程
已分裂2000个子进程
已分裂2200个子进程
已分裂2400个子进程
已分裂2600个子进程
已分裂2800个子进程
已分裂3000个子进程
已分裂3200个子进程
已分裂3400个子进程
已分裂3600个子进程
已分裂3800个子进程
已分裂4000个子进程
已分裂4200个子进程
已分裂4400个子进程
已分裂4600个子进程
已分裂4800个子进程
已分裂5000个子进程
已分裂5200个子进程
已分裂5400个子进程
已分裂5600个子进程
已分裂5800个子进程
已分裂6000个子进程
已分裂6200个子进程
已分裂6400个子进程
已分裂6600个子进程
已分裂6800个子进程
已分裂7000个子进程
已分裂7200个子进程
成功分裂7378个子进程
```

图 5: 1024MB 的情况

```
已分裂7600个子进程  
已分裂7800个子进程  
已分裂8000个子进程  
已分裂8200个子进程  
已分裂8400个子进程  
已分裂8600个子进程  
已分裂8800个子进程  
已分裂9000个子进程  
已分裂9200个子进程  
已分裂9400个子进程  
已分裂9600个子进程  
已分裂9800个子进程  
已分裂10000个子进程  
已分裂10200个子进程  
已分裂10400个子进程  
已分裂10600个子进程  
已分裂10800个子进程  
已分裂11000个子进程  
已分裂11200个子进程  
已分裂11400个子进程  
已分裂11600个子进程  
已分裂11800个子进程  
已分裂12000个子进程  
已分裂12200个子进程  
已分裂12400个子进程  
已分裂12600个子进程  
已分裂12800个子进程  
已分裂13000个子进程  
已分裂13200个子进程  
已分裂13400个子进程  
已分裂13600个子进程  
已分裂13800个子进程  
已分裂14000个子进程  
成功分裂14147个子进程
```

图 6: 2048MB 的情况

5. test5-2.c

我的做法是子进程退出后让 init 进程回收，然后可以创建新的进程。

```
54 int main(int argc, char* argv){
55     if(argc != 2){
56         printf("Usage: ./test4-1 number\n");
57         exit(EXIT_FAILURE);
58     }
59     int n = atoi(argv[1]);
60     //printf("loop %d times\n", n);
61
62     create_daemon();
63
64     int i;
65     int flag = 1;
66     for (i = 0; i < n; ++i){
67         pid_t pid = fork();
68
69         // loop until fork successfully
70         while(pid < 0){
71             sleep(1);
72             printf(".");
73             pid = fork();
74         }
75
76         // child
77         if (pid == 0){
78             char str[1024];
79             str[0] = 'z';
80             str[511] = 'y';
81             str[1023] = 'c';
82
83             sleep(20);
84             exit(EXIT_SUCCESS);
85         }
86     }
```

54,25 82%

图 7: test5-2.c 部分代码

为了让子进程退出后自动回收,和 T4 最后一问一样,添加 signal(SIGCHLD, SIG_IGN).

打印这么多点是因为父进程在等待的时候打点

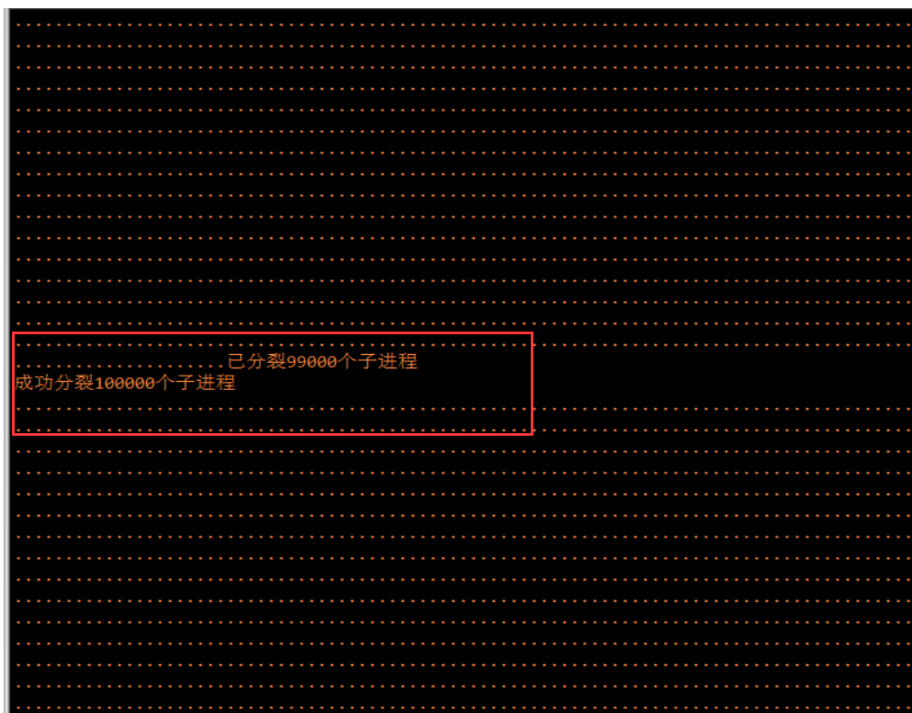


图 8: 512MB 内存下执行 test5-2.c, 成功 fork100000 个进程

6. 最大进程号

(为了清楚起见把打印. 删了) 最大进程号 131071

```
已分裂70000个子进程
已分裂71000个子进程
已分裂72000个子进程
已分裂73000个子进程
已分裂74000个子进程
已分裂75000个子进程
已分裂76000个子进程
已分裂77000个子进程
已分裂78000个子进程
已分裂79000个子进程
已分裂80000个子进程
已分裂81000个子进程
已分裂82000个子进程
已分裂83000个子进程
已分裂84000个子进程
已分裂85000个子进程
已分裂86000个子进程
已分裂87000个子进程
已分裂88000个子进程
已分裂89000个子进程
已分裂90000个子进程
已分裂91000个子进程
已分裂92000个子进程
已分裂93000个子进程
已分裂94000个子进程
已分裂95000个子进程
已分裂96000个子进程
已分裂97000个子进程
已分裂98000个子进程
已分裂99000个子进程
成功分裂100000个子进程
最大进程号为131071
```

图 9: 最大进程号

7. test5-3.c

如果强行要求父进程来回收, 就必须要用 `wait/waitpid` 了. 同时删掉之前 `signal(SIGCHLD, SIG_IGN)` 具体可以这么使用


```

82
83     // child
84     if (pid == 0){
85         char str[1024];
86         str[0] = 'z';
87         str[511] = 'y';
88         str[1023] = 'c';
89
90         sleep(1);
91         exit(EXIT_SUCCESS);
92     }
93
94     if(fork_num % 1000 == 0){
95         while(waitpid(-1, NULL, WNOHANG) > 0)
96             recycle_num++;
97         printf("已分裂%d个子进程\n", i+1);
98         printf("分裂进程数:%d, 回收进程数%d\n", fork_num, recycle_num);
99     };
100     sleep(3);
101 }
102
103 printf("成功分裂%d个子进程\n", i);
104 printf("最大进程号为%d\n", max_pid);
105 fflush(stdout);
106 while(1){
107     while(waitpid(-1, NULL, WNOHANG) > 0)
108         recycle_num++;
109     printf("分裂进程数:%d, 回收进程数%d\n", fork_num, recycle_num);
110     sleep(5);
111 }
112 }

```

"test5-3.c" 112L, 1943C 107,3-17 底端

图 10: waitpid

结果如下

```
分裂进程数:41000, 回收进程数40000
已分裂42000个子进程
分裂进程数:42000, 回收进程数41000
已分裂43000个子进程
分裂进程数:43000, 回收进程数42000
已分裂44000个子进程
分裂进程数:44000, 回收进程数43000
已分裂45000个子进程
分裂进程数:45000, 回收进程数44000
已分裂46000个子进程
分裂进程数:46000, 回收进程数45000
已分裂47000个子进程
分裂进程数:47000, 回收进程数46000
已分裂48000个子进程
分裂进程数:48000, 回收进程数47000
已分裂49000个子进程
分裂进程数:49000, 回收进程数48000
已分裂50000个子进程
分裂进程数:50000, 回收进程数49000
成功分裂50000个子进程
最大进程号为131071
分裂进程数:50000, 回收进程数50000
分裂进程数:50000, 回收进程数50000
分裂进程数:50000, 回收进程数50000
分裂进程数:50000, 回收进程数50000
分裂进程数:50000, 回收进程数50000
分裂进程数:50000, 回收进程数50000
分裂进程数:50000, 回收进程数50000
分裂进程数:50000, 回收进程数50000
分裂进程数:50000, 回收进程数50000
分裂进程数:50000, 回收进程数50000
分裂进程数:50000, 回收进程数50000
分裂进程数:50000, 回收进程数50000
```

图 11: 执行 test5-3