[**僵尸进程ZOMBIE**](http://www.cnblogs.com/zhaoyl/archive/2012/05/03/2481294.html)

　　僵尸进程是指它的父进程已经退出(父进程没有等待(调用wait/waitpid)它)，而该进程dead之后没有进程接受，就成为僵尸进程，也就是(zombie)进程。

　　一个进程在调用exit命令结束自己的生命的时候，其实它并没有真正的被销毁，而是留下一个称为僵尸进程(Zombie)的数据结构(系统调用exit，它的作用是使进程退出，但也仅仅限于将一个正常的进程变成一个僵尸进程，并不能将其完全销毁)。

　　在Linux进程的状态中，僵尸进程是非常特殊的一种，它已经放弃了几乎所有内存空间，没有任何可执行代码，也不能被调度，仅仅在进程列表中保留一个位置，记载该进程的退出状态等信息供其他进程收集，除此之外，僵尸进程不再占有任何内存空间。它需要它的父进程来为它收尸。

　　如果他的父进程没安装SIGCHLD信号处理函数调用wait或waitpid()等待子进程结束，又没有显式忽略该信号**[参见APUE 10.7]**，那么它就一直保持僵尸状态，如果这时父进程结束了，那么init进程自动会接手这个子进程，为它收尸，它还是能被清除的。

　　但是如果父进程是一个循环，不会结束，那么子进程就会一直保持僵尸状态，这就是为什么系统中有时会有很多的僵尸进程。系统所能使用的进程号是有限的,如果大量的产生僵死进程,将因为没有可用的进程号而导致系统不能产生新的进程.

**僵尸进程的避免**

　　1、父进程通过wait和waitpid等函数等待子进程结束，这会导致父进程挂起

　　2、如果父进程很忙，那么可以用signal函数为SIGCHLD安装handler，因为子进程结束后，父进程会收到该信号，可以在handler中调用wait回收

　　3、如果父进程不关心子进程什么时候结束，那么可以用signal(SIGCHLD, SIG\_IGN) 通知内核，自己对子进程的结束不感兴趣，那么子进程结束后，内核会回收，并不再给父进程发送信号

　　4、还有一些技巧，就是fork两次，父进程fork一个子进程，然后继续工作，子进程fork一个孙进程后退出，那么孙进程被init接管，孙进程结束后，init会回收。不过子进程的回收还要自己做。实现可以参考 <http://blog.csdn.net/lzueclipse/article/details/5777264>

**子进程结束后为什么要进入僵尸状态**

　　因为父进程可能要取得子进程的退出状态等信息。

**僵尸状态是每个子进程必然经过的状态**

　　任何一个子进程(init除外)在exit()之后，并非马上就消失掉，而是留下一个称为僵尸进程(Zombie)的数据结构(它占用一点内存资源,也就是进程表里还有一个记录)，等待父进程处理。这是每个子进程在结束时都要经过的阶段。如果子进程在exit()之后，父进程没有来得及处理，这时用ps命令就能看到子进程的状态是“Z”。

　　如果父进程能及时处理，可能用ps命令就来不及看到子进程的僵尸状态，但这并不等于子进程不经过僵尸状态。

　　如果父进程在子进程结束之前退出，则子进程将由init接管。init将会以父进程的身份对僵尸状态的子进程进行处理。

**如何查看僵尸进程**

　　在linux中，利用命令ps，可以看到有标记为Z的进程就是僵尸进程。

　　ps -ef|grep defunc可以找出僵尸进程.

　　可以用ps的-l选项,得到更详细的进程信息. F(Flag)：一系列数字的和，表示进程的当前状态。这些数字的含义为：

　　00：若单独显示，表示此进程已被终止。  
　　01：进程是核心进程的一部分，常驻于系统主存。如：sched、 vhand 、bdflush 等。  
　　02：Parent is tracing process.  
　　04：Tracing parent’s signal has stopped the process; the parent　is waiting ( ptrace(S)).  
　　10：进程在优先级低于或等于25时，进入休眠状态，而且不能用信号唤醒，例如在等待一个inode被创建时　　　  
　　20：进程被装入主存(primary memory)  
　　40：进程被锁在主存，在事务完成前不能被置换

　　S(state of the process )

　　O：进程正在处理器运行　  
　　S：休眠状态(sleeping)  
　　R：等待运行(runable)　　　  
　　I：空闲状态(idle)  
　　Z：僵尸状态(zombie)　　　  
　　T：跟踪状态(Traced)  
　　B：进程正在等待更多的内存页  
　　C：cpu利用率的估算值(cpu usage)

**僵尸进程清除的方法**

**三、如何避免僵屍進程的産生   
  
1、父进程通过wait和waitpid等函数等待子进程结束，这会导致父进程挂起   
  
2. 如果父进程很忙，那么可以用signal函数为SIGCHLD安装handler，因为子进程结束后，父进程会收到该信号，可以在handler中调用wait回收（没有试验过）   
  
3. 如果父进程不关心子进程什么时候结束，那么可以用signal（SIGCHLD, SIG\_IGN）通知内核，自己对子进程的结束不感兴趣，那么子进程结束后，内核会回收，并不再给父进程发送信号   
  
4. 还有一些技巧，就是fork两次，父进程fork一个子进程，然后继续工作，子进程fork一个孙进程后退出，那么孙进程被init接管，孙进程结束后，init会回收。不过子进程的回收估计还要自己做。**

　　5. kill -18 PPID　(PPID是其父进程,没测试!!!)

　　这个信号是告诉父进程，该子进程已经死亡了，请收回分配给他的资源。

　　SIGCONT也是一个有意思的信号。如前所述，当进程停止的时候，这个信号用来告诉进程恢复运行。该信号的有趣的地方在于：它不能被忽略或阻塞，但可以被捕获。缺省行为是丢弃该信号。

　　6.终止父进程

　　如果方法2不能终止，可采用终止其父进程的方法(如果其父进程不需要的话)父进程死后，僵尸进程成为”孤儿进程”，过继给1号进程init，init始终会负责清理僵尸进程．它产生的所有僵尸进程也跟着消失。

　　先看其父进程又无其他子进程，如果有，可能需要先kill其他子进程，也就是兄弟进程。方法是：

　　kill –15 PID1 PID2　(PID1,PID2是僵尸进程的父进程的其它子进程)。

　　然后再kill父进程：kill –15 PPID

　　这样僵尸进程就可能被完全杀掉了。

　　附一段测试代码：

[复制代码](javascript:void(0);)

#include <sys/types.h>

#include <unistd.h>

#include <stdio.h>

#include <sys/wait.h>

int main(){

int name=1;

int status;

pid\_t id;

id=fork();

if(id==0){

name=2;

　　　　 exit(7);

}

if(id>0){

sleep(5);//在另一个窗口观察 ps -ef -o pid,ppid,comm,stat|grep a.out 这个期间是Z状态   
　　　　　　　　　wait(&status);//这时候父进程捕获子进程   
　　　　　　　　 printf("child exit status is: %d\n",WEXITSTATUS(status)); sleep(5);//子进程僵尸成功退出   
　　　　　}   
　　　　 if(id<0)  
　　　　{   
　　　　　　perror("fork");   
　　　　 return0;   
　　　　}   
　　　　printf("%d:%d\n",getpid(),name);   
　　　　return0;   
}

[复制代码](javascript:void(0);)