<https://blog.csdn.net/helloguoqijun/article/details/77489398>

**最近定位了一个线程没有正常退出的bug，导致一直创建线程，然后调度超时挂死的bug，花了两天的时间，要是尽早用sp看一下，这个问题就结束了，所以别看命令简单，关键时候还是好用的。**

**1. pstree**

pstree以树结构显示进程  
$ pstree -p work | grep ad  
sshd(22669)---bash(22670)---ad\_preprocess(4551)-+-{ad\_preprocess}(4552)  
                                                |-{ad\_preprocess}(4553)  
                                                |-{ad\_preprocess}(4554)  
                                                |-{ad\_preprocess}(4555)  
                                                |-{ad\_preprocess}(4556)  
                                                `-{ad\_preprocess}(4557)

work为工作用户，-p为显示进程识别码，ad\_preprocess共启动了6个子线程，加上主线程共7个线程

**2. ps -Lf**

$ ps -Lf 4551  
UID        PID  PPID   LWP  C NLWP STIME TTY      STAT   TIME CMD  
work      4551 22670  4551  2    7 16:30 pts/2    Sl+    0:02 ./ad\_preprocess  
work      4551 22670  4552  0    7 16:30 pts/2    Sl+    0:00 ./ad\_preprocess  
work      4551 22670  4553  0    7 16:30 pts/2    Sl+    0:00 ./ad\_preprocess  
work      4551 22670  4554  0    7 16:30 pts/2    Sl+    0:00 ./ad\_preprocess  
work      4551 22670  4555  0    7 16:30 pts/2    Sl+    0:00 ./ad\_preprocess  
work      4551 22670  4556  0    7 16:30 pts/2    Sl+    0:00 ./ad\_preprocess  
work      4551 22670  4557  0    7 16:30 pts/2    Sl+    0:00 ./ad\_preprocess

进程共启动了7个线程

linux上进程有5种状态:

1. 运行(正在运行或在运行队列中等待)

2. 中断(休眠中, 受阻, 在等待某个条件的形成或接受到信号)

3. 不可中断(收到信号不唤醒和不可运行, 进程必须等待直到有中断发生)

4. 僵死(进程已终止, 但进程描述符存在, 直到父进程调用wait4()系统调用后释放)

5. 停止(进程收到SIGSTOP, SIGSTP, SIGTIN, SIGTOU信号后停止运行运行)

ps工具标识进程的5种状态码:

D 不可中断 uninterruptible sleep (usually IO)

R 运行 runnable (on run queue)

S 中断 sleeping

T 停止 traced or stopped

Z 僵死 a defunct (”zombie”) process

**3. pstack**

pstack显示每个进程的栈跟踪

$ pstack 4551  
Thread 7 (Thread 1084229984 (LWP 4552)):  
#0  0x000000302afc63dc in epoll\_wait () from /lib64/tls/libc.so.6  
#1  0x00000000006f0730 in ub::EPollEx::poll ()  
#2  0x00000000006f172a in ub::NetReactor::callback ()  
#3  0x00000000006fbbbb in ub::UBTask::CALLBACK ()  
#4  0x000000302b80610a in start\_thread () from /lib64/tls/libpthread.so.0  
#5  0x000000302afc6003 in clone () from /lib64/tls/libc.so.6  
#6  0x0000000000000000 in ?? ()  
Thread 6 (Thread 1094719840 (LWP 4553)):  
#0  0x000000302afc63dc in epoll\_wait () from /lib64/tls/libc.so.6  
#1  0x00000000006f0730 in ub::EPollEx::poll ()  
#2  0x00000000006f172a in ub::NetReactor::callback ()  
#3  0x00000000006fbbbb in ub::UBTask::CALLBACK ()  
#4  0x000000302b80610a in start\_thread () from /lib64/tls/libpthread.so.0  
#5  0x000000302afc6003 in clone () from /lib64/tls/libc.so.6  
#6  0x0000000000000000 in ?? ()  
Thread 5 (Thread 1105209696 (LWP 4554)):  
#0  0x000000302b80baa5 in \_\_nanosleep\_nocancel ()  
#1  0x000000000079e758 in comcm::ms\_sleep ()  
#2  0x00000000006c8581 in ub::UbClientManager::healthyCheck ()  
#3  0x00000000006c8471 in ub::UbClientManager::start\_healthy\_check ()  
#4  0x000000302b80610a in start\_thread () from /lib64/tls/libpthread.so.0  
#5  0x000000302afc6003 in clone () from /lib64/tls/libc.so.6  
#6  0x0000000000000000 in ?? ()  
Thread 4 (Thread 1115699552 (LWP 4555)):  
#0  0x000000302b80baa5 in \_\_nanosleep\_nocancel ()  
#1  0x0000000000482b0e in armor::armor\_check\_thread ()  
#2  0x000000302b80610a in start\_thread () from /lib64/tls/libpthread.so.0  
#3  0x000000302afc6003 in clone () from /lib64/tls/libc.so.6  
#4  0x0000000000000000 in ?? ()  
Thread 3 (Thread 1126189408 (LWP 4556)):  
#0  0x000000302af8f1a5 in \_\_nanosleep\_nocancel () from /lib64/tls/libc.so.6  
#1  0x000000302af8f010 in sleep () from /lib64/tls/libc.so.6  
#2  0x000000000044c972 in Business\_config\_manager::run ()  
#3  0x0000000000457b83 in Thread::run\_thread ()  
#4  0x000000302b80610a in start\_thread () from /lib64/tls/libpthread.so.0  
#5  0x000000302afc6003 in clone () from /lib64/tls/libc.so.6  
#6  0x0000000000000000 in ?? ()  
Thread 2 (Thread 1136679264 (LWP 4557)):  
#0  0x000000302af8f1a5 in \_\_nanosleep\_nocancel () from /lib64/tls/libc.so.6  
#1  0x000000302af8f010 in sleep () from /lib64/tls/libc.so.6  
#2  0x00000000004524bb in Process\_thread::sleep\_period ()  
#3  0x0000000000452641 in Process\_thread::run ()  
#4  0x0000000000457b83 in Thread::run\_thread ()  
#5  0x000000302b80610a in start\_thread () from /lib64/tls/libpthread.so.0  
#6  0x000000302afc6003 in clone () from /lib64/tls/libc.so.6  
#7  0x0000000000000000 in ?? ()  
Thread 1 (Thread 182894129792 (LWP 4551)):  
#0  0x000000302af8f1a5 in \_\_nanosleep\_nocancel () from /lib64/tls/libc.so.6  
#1  0x000000302af8f010 in sleep () from /lib64/tls/libc.so.6  
#2  0x0000000000420d79 in Ad\_preprocess::run ()  
#3  0x0000000000450ad0 in main ()

4、kill 终止进程

有十几种控制进程的方法，下面是一些常用的方法:

kill -STOP [pid]

发送SIGSTOP (17,19,23)停止一个进程，而并不消灭这个进程。

kill -CONT [pid]

发送SIGCONT (19,18,25)重新开始一个停止的进程。

kill -KILL [pid]

发送SIGKILL (9)强迫进程立即停止，并且不实施清理操作。

kill -9 -1

终止你拥有的全部进程。

SIGKILL 和 SIGSTOP 信号不能被捕捉、封锁或者忽略，但是，其它的信号可以。所以这是你的终极武器。