## 哪些原因会导致JAVA进程退出?

* linux的OOM killer杀死
* JVM自身故障
* jvm的OOM导致进程退出(很罕见，我至今没遇见过)

### 1.OOM killer

Linux 内核有个机制叫OOM killer（Out-Of-Memory killer），该机制会监控那些占用内存过大，尤其是瞬间很快消耗大量内存的进程，为了防止内存耗尽而内核会把该进程杀掉。  
因此，你发现java进程突然没了，首先要怀疑是不是被linux的OOM killer给干掉了！  
你可以去下面这个文件里翻

系统报错日志位置:/var/log/messages

你执行命令

egrep -i 'killed process' /var/log/messages

去日志里进行查询。

当然，你也可以去内核日志里头查询。有时Linux系统或者系统上运行的java或者其它进程，会发生一些莫名其妙的问题，比如突然挂掉了，比如突然重启等等。在软件上找不到问题所在，此时我们应该怀疑硬件或者内核的问题，此时我们就可以使用 dmesg来查看：

dmesg | grep java

可以看到内核对java进程的操作

[5673702.665338] Out of memory: Kill process 29953 (java) score 431 or sacrifice child  
[5673702.665338] Killed process 29953, UID 500, (java) total-vm:9805316kB, anon-rss:2344496kB, file-rss:128kB

### **JVM自身故障**

当JVM发生致命错误导致崩溃时，会生成一个hs\_err\_pid\_xxx.log这样的文件，该文件包含了导致 JVM crash 的重要信息，我们可以通过分析该文件定位到导致 JVM Crash 的原因，从而修复保证系统稳定。  
默认情况下，该文件是生成在工作目录下的，当然也可以通过 JVM 参数指定生成路径：

-XX:ErrorFile=/var/log/hs\_err\_pid<pid>.log

这个文件的内容主要有如下内容

* 日志头文件
* 导致 crash 的线程信息
* 所有线程信息
* 安全点和锁信息
* 堆信息
* 本地代码缓存
* 编译事件
* gc 相关记录
* jvm 内存映射
* jvm 启动参数
* 服务器信息

### **JVM的OOM**

JVM的OOM导致java进程退出的情况很少。  
因为，一般情况下，出现OOM异常，JVM的GC会进行回收，是不会导致JVM进程退出的。要真说唯一导致退出的情况，那就是内存泄漏，由于内存占用越来越大，结果就会崩掉。  
不过这种JVM的OOM导致的异常，很好排查。  
因为，你注意两个个参数

-XX:+HeapDumpOnOutOfMemoryError  
-XX:HeapDumpPath=\*/java.hprof；

然后去对应目录找dump快照文件，接下来借助VisualVM这种可视化工具分析就行。很容易定位问题。