# 死锁的分析

## 死锁代码的实现



## 死锁的dump文件

使用命令

jps先查看进程号

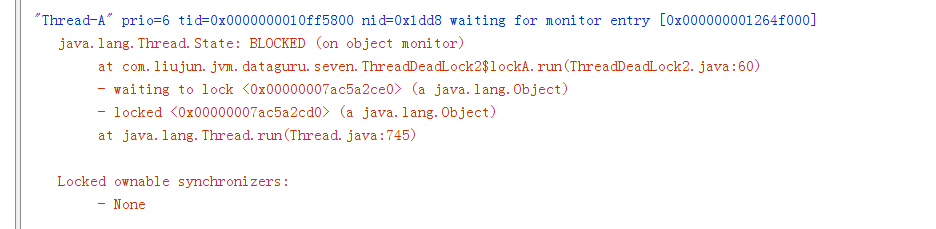
再使用jstack -l 7740 > d:/java/work/jvm/07/deallock2.tdump导出



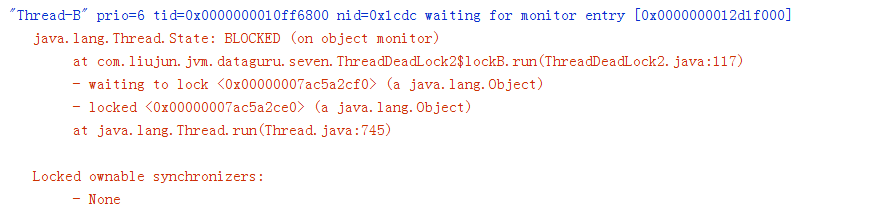
生成的线程信息



## 分析死锁的dump文件

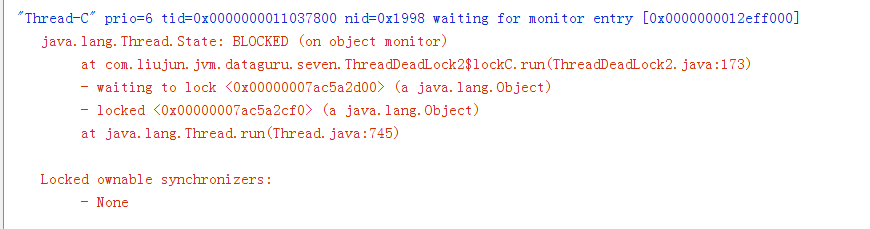


当前线程A持有0x00000007ac5a2cd0,等待0x00000007ac5a2ce0这个对象的锁释放



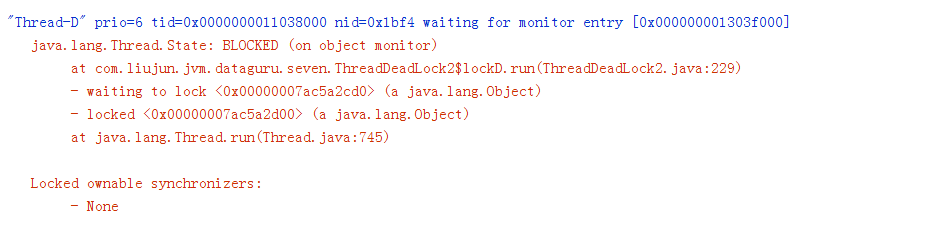
而当前A线程所等待的锁被0x00000007ac5a2ce0被线程B所持有，

而线程B等待0x00000007ac5a2cf0对象被释放



线程B所等待的0x00000007ac5a2cf0对象，被线程C所持有,

而线程C等待0x00000007ac5a2d00对象被释放

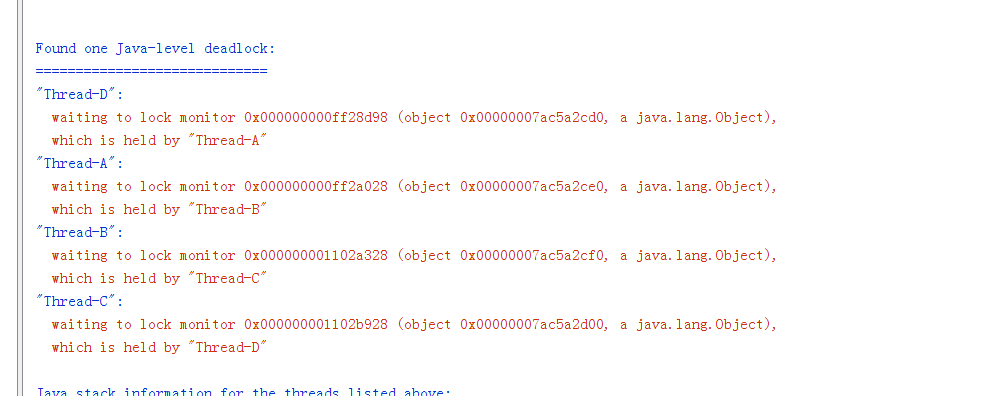


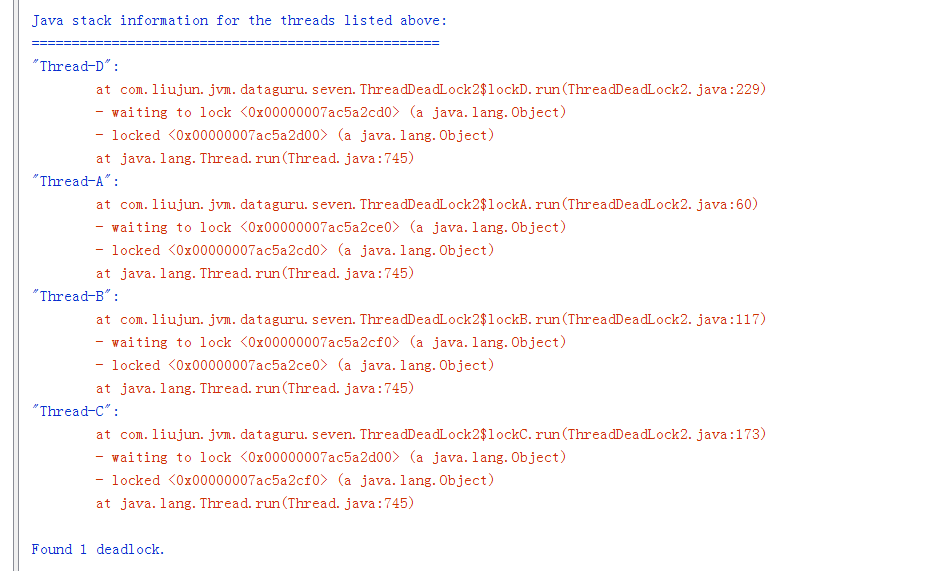
线程C所等待的0x00000007ac5a2d00对象，被线程D所持有,

而线程D等待线程A的0x00000007ac5a2cd0释放

这样就形成了一个环形的锁，导致所有的对象都不能被释放，而又形成相互等待

在线程信息中也会有死锁的信息





## 使用jvisualvm工具

