[使用GDB调试JNI代码](http://blog.csdn.net/kongxx/article/details/7329515)

分类：***JavaCLinux***

 （5180）  （2）

**本文链接：http://blog.csdn.net/kongxx/article/details/7329515**

最近在碰到一段JNI代码不停的出现core dump，用gdb看了一下core文件，信息也不是很清楚，于是想到了运行是debug JNI代码

，对于JNI代码要想动态debug还是有点小工作要做的，如下：

1. 首先修改java参数，添加-Xdebug -Xrunjdwp:transport=dt\_socket,address=8888,server=y,suspend=n作为java参数，其中8888是端口号，suspend=n意思是java启动的时候不等待客户端debug连接，如果自己的程序是在一启动就需要调用JNI，这里可以将其值设置为suspend=n，这样java进程就会启动，然后等待客户端比如eclipse去远程调试，等于eclipse的远程调试连接后，java才会继续往下走，所以修改后的运行命令类似下面的语句

**[java]** [view plain](http://blog.csdn.net/kongxx/article/details/7329515) [copy](http://blog.csdn.net/kongxx/article/details/7329515)

1. $ java -Xms64m -Xmx512m -Xdebug -Xrunjdwp:transport=dt\_socket,address=8888,server=y,suspend=n MyClass

2. 在java中需要调用JNI代码的地方设置断点，然后使用eclipse远程调试连接java程序运行的机器和端口；

3. 等到eclipse连接后，在命令行看一下上面启动的java程序的进程PID；

4. 使用gdb连接上面的进程，命令如下

**[plain]** [view plain](http://blog.csdn.net/kongxx/article/details/7329515) [copy](http://blog.csdn.net/kongxx/article/details/7329515)

1. $ gdb -p <pid>

5. 设置断点在需要调试的JNI方法位置，如下：

**[plain]** [view plain](http://blog.csdn.net/kongxx/article/details/7329515) [copy](http://blog.csdn.net/kongxx/article/details/7329515)

1. (gdb) break <method\_name>

6. 然后就可以使用各种gdb命令来调试和打印各个变量了。