昵称：yanpeng
园龄：8年7个月
粉丝：7
关注：0
+加关注

< 2009年6月 >

日	一	二	三	四	五	六
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类

C#(6)
Java相关知识(18)
SQL(3)
感悟摘抄(4)
计算机硬件网络(9)

随笔档案

2013年5月 (1)
2012年5月 (1)
2011年1月 (1)
2010年12月 (4)
2010年9月 (1)
2010年7月 (1)
2010年6月 (3)
2010年4月 (4)
2009年10月 (3)
2009年6月 (9)
2009年5月 (2)
2009年3月 (1)
2009年2月 (8)
2009年1月 (3)
2008年11月 (1)
2008年10月 (1)
2008年9月 (1)
2008年8月 (1)
2008年7月 (1)

文章分类

About System
Code

Friends

咚咚锵，柴米油盐

最新评论

1. Re:word2007不能编辑文档的问题
有用！
--始于弱冠

2. Re:Ant学习
写得很详细，3Q
--逍遥K杰

3. Re:word2007不能编辑文档的问题
HKEY_CURRENT_USER\Software\Microsoft Office\12.0\Word\Addins
找不到这个目录呀

javap 学习日记~1

一直在学习Java,碰到了很多问题，碰到了很多关于i++和++i的难题，以及最经典的String str = "abc" 共创建了几个对象的疑难杂症。 知道有一日知道了java的反汇编 命令 javap。现将学习记录做一小结，以供自己以后翻看。如果有错误的地方，请指正

1.javap是什么：

where options include:
-c Disassemble the code
-classpath <pathlist> Specify where to find user class files
-extdirs <dirs> Override location of installed extensions
-help Print this usage message
-J<flag> Pass <flag> directly to the runtime system
-l Print line number and local variable tables
-public Show only public classes and members
-protected Show protected/public classes and members
-package Show package/protected/public classes
and members (default)
-private Show all classes and members
-s Print internal type signatures
-bootclasspath <pathlist> Override location of class files loaded
by the bootstrap class loader
-verbose Print stack size, number of locals and args for met
hods
If verifying, print reasons for failure

以上为百度百科里对它的描述，只是介绍了javap的一些参数和使用方法，而我们要用的就是这一个：-c Disassemble the code。

明确一个问题：javap是什么？网上有人称之为 反汇编器，可以查看java编译器为我们生成的字节码。通过它，我们可以对照源代码和字节码，从而了解很多编译器内部的工作。

2.初步认识javap

从一个最简单的例子开始：

```
1  /**
2   * For Test Javap
3   *
4   * @author YanPeng
5   */
6  public class TestJavap {
7
8    /**
9     * @param args
10    */
11   public static void main(String[] args) {
12     int i = 2;
13     int j = 3;
14   }
15 }
```

这个例子中，我们只是简单的声明了两个int型变量并赋上初值。下面我们看看javap给我们带来了什么：（当然执行javap命令前，你得首先配置好自己的环境，能用javac编译通过了，即：**javac TestJavap.java** ）

```
D:¥Yanpeng¥workspace¥HelloWorld¥src>javap -c TestJavap
Compiled from "TestJavap.java"
public class TestJavap extends java.lang.Object{
public TestJavap();
  Code:
   0:   aload_0
   1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   4:   return

public static void main(java.lang.String[]);
  Code:
   0:   iconst_2
   1:   istore_1
   2:   iconst_3
   3:   istore_2
   4:   return

}
```

我们只看（方便起见，将注释写到每句后面）

```
Code:
  0:  iconst_2   //把2放到栈顶
  1:  istore_1   //把栈顶的值放到局部变量1中，即i中
  2:  iconst_3   //把3放到栈顶
  3:  istore_2   //把栈顶的值放到局部变量1中，即j中
  4:  return
```

是不是很简单？（当然，估计需要点数据结构的知识），那我们就补点java的关于堆栈的知识：

--_小小强

4. Re:javap 学习日记~1

堆栈概念我知道的！就是有点理解不了：把栈顶的值放到局部变量1中，即i中

这些概念！ X86体系的，栈就是编译时候确定的，堆就是调用malloc之类的函数，即分配器来分配的但是暂时有点理解不了jav.....

--拌面

5. Re:javap 学习日记~1

@拌面java把内存分成两种，一种堆内存，一种叫做栈内存。一般 new出来的对象是放在堆内存中的栈内存与对内存相比速度会快。仅次于寄存器，一般只放一些基础类型。比如int、char、long等"额。有.....

--yanpeng

阅读排行榜

1. Java连接oracle数据库实例(43286)
2. word2007不能编辑文档的问题(11933)
3. MyEclipse优化浅析(4588)
4. 看硬盘编号识硬盘规格(单碟还是双碟)(3719)
5. Ant学习(2698)

评论排行榜

1. javap 学习日记~1(3)
2. javap 学习日记~2(2)
3. word2007不能编辑文档的问题(2)
4. Ant学习(1)

推荐排行榜

1. javap 学习日记~1(1)
2. Myeclipse使用中内存溢出的问题(1)

对于 int i = 2;首先它会在栈中创建一个变量为i的引用，然后查找有没有字面值为2的地址，没找到，就开辟一个存放2这个字面值的地址，然后将i指向2的地址。

看了这段话，再比较下上面的注释，是不是完全吻合？

为了验证上面这一说法，我们继续实验：

```
1  /**
2   * For Test Javap
3   *
4   * @author YanPeng
5   */
6  public class TestJavap {
7
8    /**
9     * @param args
10    */
11   public static void main(String[] args) {
12     int i = 2;
13     int j = 2;
14   }
15 }
16
```

我们将 i 和 j的值都设为2。按照以上理论，在声明j的时候，会去栈中招有没有字面值为2的地址，由于在栈中已经有2这个字面值，便将j直接指向2的地址。这样，就出现了i与j同时均指向2的情况。

拿出javap -c进行反编译：结果如下：

```
D:¥Yanpeng¥workspace¥HelloWorld¥src>javap -c TestJavap
Compiled from "TestJavap.java"
public class TestJavap extends java.lang.Object{
public TestJavap();
  Code:
   0:   aload_0
   1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   4:   return

public static void main(java.lang.String[]);
  Code:
   0:   iconst_2
   1:   istore_1
   2:   iconst_2
   3:   istore_2
   4:   return

}
```

Code:

0: iconst_2 //把2放到栈顶
1: istore_1 //把栈顶的值放到局部变量1中，即i中
2: iconst_2 //把2放到栈顶
3: istore_2 //把栈顶的值放到局部变量2中，即j中(i 和 j同时指向2)
4: return

虽然这里说i和j同时指向2，但这里不等于说i和j指向同一块地址（java是不允许程序员直接修改堆栈中的数据的，所以就不要想着，我是不是可以修改栈中的2，那样岂不是i和j的值都会变化。另：在编译器内部，遇到j=2；时，它就会重新搜索栈中是否有2的字面值，如果没有，重新开辟地址存放2的值；如果已经有了，则直接将j指向这个地址。因此,就算j另被赋值为其他值，如j=4,j值的改变不会影响到i的值。 ）

再来一个例子：

```
1  /**
2   * For Test Javap
3   *
4   * @author YanPeng
5   */
6  public class TestJavap {
7
8    /**
9     * @param args
10    */
11   public static void main(String[] args) {
12     int i = 2;
13     int j = i;
14   }
15 }
16
```

还是javap -c

```
D:¥Yanpeng¥workspace¥HelloWorld¥src>javap -c TestJavap
Compiled from "TestJavap.java"
public class TestJavap extends java.lang.Object{
public TestJavap();
  Code:
   0:   aload_0
   1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   4:   return

public static void main(java.lang.String[]);
  Code:
   0:   iconst_2
   1:   istore_1
   2:   iload_1
   3:   istore_2
   4:   return

}
```

Code:
0: iconst_2 **//把2放到栈顶**
1: istore_1 **//把栈顶的值放到局部变量1中，即i中**
2: iload_1 **//把i的值放到栈顶，也就是说此时栈顶的值是2**
3: istore_2 **//把栈顶的值放到局部变量2中，即j中**
4: return

看到这里是不是有点明确了？

分类: Java相关知识

yanpeng
关注 - 0
粉丝 - 7
+加关注

« 上一篇：C#面试基础问题
» 下一篇：javap 学习日记~2

posted @ 2009-06-09 16:08 yanpeng 阅读(1689) 评论(3) 编辑 收藏

评论列表

#1楼 2012-06-22 18:14 拌面

Code:
0: iconst_2 //把2放到栈顶
1: istore_1 //把栈顶的值放到局部变量1中，即i中
2: iconst_2 //把2放到栈顶
3: istore_2 //把栈顶的值放到局部变量2中，即j中(i 和 j同时指向2)
4: return

额。有点理解不了。。看得懂IA32，看不懂这个！把栈顶的值放到局部变量1中，即i中。。。这里的局部变量是什么样的存在？

支持(0) 反对(0)

#2楼[楼主] 2012-06-22 23:23 yanpeng

@ 拌面
java把内存分成两种，一种堆内存，一种叫做栈内存。
一般 new出来的对象是放在堆内存中的
栈内存与对内存相比速度会快。仅次于寄存器，一般只放一些基础类型。比如int、char、long等

“额。有点理解不了。。看得懂IA32，看不懂这个！把栈顶的值放到局部变量1中，即i中。。。这里的局部变量是什么样的存在？”
我的理解是 这个局部变量是放在栈内存中的一个能唯一标识 i的一个东西，java每生成一个对象都是这样的，先在内存中划分一定空间定义对象，然后再分一个空间指向刚才定义的对象。

以上浅见，有可能有出入的地方，欢迎一起探讨。

支持(0) 反对(0)

#3楼 2012-06-23 12:46 拌面

堆栈概念我知道的！就是有点理解不了：把栈顶的值放到局部变量1中，即i中 这些概念！ X86体系的，栈就是编译时候确定的，堆就是调用malloc之类的函数，即分配器来分配的

但是暂时有点理解不了java 字节码（还没怎么看！刚接触~），再看看！ 有问题时要向兄弟你请教的哦~

支持(0) 反对(0)