软件调试

软件控制着硬件，谁来控制软件。Bug层出不穷，Exception斩不断理还乱。软件缺陷难以避免，危害还很大。怎么办？调试！理解系统行为，发现bug根源，观察到每个“毛孔”，控制到每根“神经”，是你征服软件，运筹帷幄的最得力助手。



软件调试的基本过程：

一：重现bug。重现测试或其他方法发现的问题。

二：定位bug。利用各种调试工具，寻找导致软件故障的根源。这就是软件调试的核心工作。

三：探索和实现解决方案。根据寻找到的故障根源，设计解决方案。要注意避免引入新的bug。

软件调试具有难度大、难以估算完成时间和关联性强的特点。对于一些小规模程序，使用输出语句就可以帮助你进行调试。用类似System.out.println()的语句来输出所需要观察的值，但这种方法对于大规模的程序而⾔言不太现实的：

一：是因为调试过程中可能的输出值太多，难以在屏幕上找到需要的值；

二: 这会对源程序进⾏行修改，程序员必须保证在调试完成后恢复所有的代码，这非常容易引入新的bug。

因此我们需要调试器来帮我们进行调试。

调试器常见调试操作：

1. 断点（Breakpoint）

断点是调试器调试时最常用的技术之一。它的基本原理是在某一位置设置一个中断点，当cpu执行到这个位置时，便停止被调试的程序。中断到调试器中，让调试者分析和调试。

根据断点的设置位置不同，可以分为以下几种：

1. 代码断点。设置断点的位置为某一段代码的起始处。当cpu执行到此代码时产生中断。
2. 数据断点。设置断点的位置为数据的起始地址。当被调试程序访问此数据的地址时，命中断点。可以定义触发断点的方式为：读/写断点。
3. IO断点。 设置断点的位置为某一IO地址。当程序访问指定IO地址的端口时中断到调试器。

根据断点的设置方式，可以分为：软件断点和硬件断点。

软件断点是通过向指定的代码位置插入专用的断点指令和实现的。比如IA32cpu的INT3指令。机器码为0xCC。

硬件断点是通过设置cpu的调试寄存器来实现的，IA32cpu定义了8个调试寄存器。DR0-DR7。可以同时设置最多四个硬件断点。

硬件断点可以设置数据、代码和IO断点。而软件断点只能设置代码断点。

2. 调试中语句运行控制

• Step Into：跳入，进入到某个方法中。

• Step Over：单步跳过突出显示的代码行并执行同一个方法中的下一行（或者它在调用当前方法的方法中继续）。

• Step Return：单步跳出正在执行的方法。

• Resume：继续执行程序。

现在开发所用的IDE一本都会提供调试工具如Eclipse和Visual Studio，下面提供了他们的使用说明：

Eclipse：<http://www.ibm.com/developerworks/cn/linux/opensource/os-ecbug/>

VisualStudio：<http://msdn.microsoft.com/zh-cn/library/k0k771bt(v=vs.80).aspx>