# 什么是

作为一个Java程序员，我们每天都在写Java代码，我们写的代码都是在一个叫做Java虚拟机的东西上执行的。但是如果要问什么是虚拟机，恐怕很多人就会模棱两可了。

虚拟机是一种**抽象化的**[**计算机**](http://baike.baidu.com/view/3314.htm)，通过**在实际的计算机上仿真模拟各种计算机功能来实现的**。Java虚拟机**有自己完善的**[**硬体**](http://baike.baidu.com/view/2871257.htm)**架构，如**[**处理器**](http://baike.baidu.com/view/50152.htm)**、**[**堆栈**](http://baike.baidu.com/view/93201.htm)**、**[**寄存器**](http://baike.baidu.com/view/6159.htm)等，还**具有相应的**[**指令**](http://baike.baidu.com/view/178461.htm)**系统**。**JVM屏蔽了与具体**[**操作系统**](http://baike.baidu.com/view/880.htm)**平台相关的信息，使得Java**[**程序**](http://baike.baidu.com/view/17674.htm)**只需生成在Java虚拟机上运行的目标代码（**[**字节码**](http://baike.baidu.com/view/560330.htm)**），就可以在多种平台上不加修改地运行。**

这种解释应该算是正确的，但是只描述了**虚拟机的外部行为和功能**，并没有针对内部原理做出说明。一般情况下我们不需要知道虚拟机的运行原理，只要专注写java代码就可以了，这也正是虚拟机之所以存在的原因--**屏蔽底层操作系统平台的不同并且减少基于原生语言开发的复杂性，使java这门语言能够跨各种平台**（只要虚拟机厂商在特定平台上实现了虚拟机），并且简单易用。这些都是虚拟机的外部特性，但是从这些信息来解释虚拟机，未免太笼统了，无法让我们知道**内部原理**。

## 从进程的角度解释JVM

让我们尝试从操作系统的层面来理解虚拟机。我们知道，虚拟机是运行在操作系统之中的，那么什么东西才能在操作系统中运行呢？当然是进程，因为进程是操作系统中的执行单位。可以这样理解，当它在运行的时候，它就是一个操作系统中的进程实例，当它没有在运行时（作为可执行文件存放于文件系统中），可以把它叫做程序。

对命令行比较熟悉的同学，都知道其实一个命令对应一个可执行的二进制文件，当敲下这个命令并且回车后，就会创建一个进程，加载对应的可执行文件到进程的地址空间中，并且执行其中的指令。下面对比C语言和Java语言的HelloWorld程序来说明问题。

首先编写C语言版的HelloWorld程序。

**[cpp]** [view plain](https://blog.csdn.net/zhangjg_blog/article/details/20380971) [copy](https://blog.csdn.net/zhangjg_blog/article/details/20380971)

1. #include <stdio.h>
2. #include <stdlib.h>
4. **int** main(**void**) {
5. printf("hello world\n");
6. **return** 0;
7. }

编译C语言版的HelloWorld程序：

**[plain]** [view plain](https://blog.csdn.net/zhangjg_blog/article/details/20380971) [copy](https://blog.csdn.net/zhangjg_blog/article/details/20380971)

1. gcc HelloWorld.c -o HelloWorld

运行C语言版的HelloWorld程序：

**[plain]** [view plain](https://blog.csdn.net/zhangjg_blog/article/details/20380971) [copy](https://blog.csdn.net/zhangjg_blog/article/details/20380971)

1. zhangjg@linux:/deve/workspace/HelloWorld/src$ ./HelloWorld
2. hello world

gcc编译器编译后的文件直接就是可被操作系统识别的二进制可执行文件，当我们在命令行中敲下 ./HelloWorld这条命令的时候， 直接创建一个进程， 并且将可执行文件加载到进程的地址空间中， 执行文件中的指令。

作为对比， 我们看一下Java版HelloWord程序的编译和执行形式。

# JVM内存区域

我们在编写程序时，经常会遇到OOM（out of Memory）以及内存泄漏等问题。为了避免出现这些问题，我们首先必须对JVM的内存划分有个具体的认识。JVM将内存主要划分为：方法区、虚拟机栈、本地方法栈、堆、程序计数器。JVM运行时数据区如下：   
