[

Java千百问_01基本概念（001）_什么是Java

,

点击进入_更多_Java千百问-基本概念

1、什么是Java

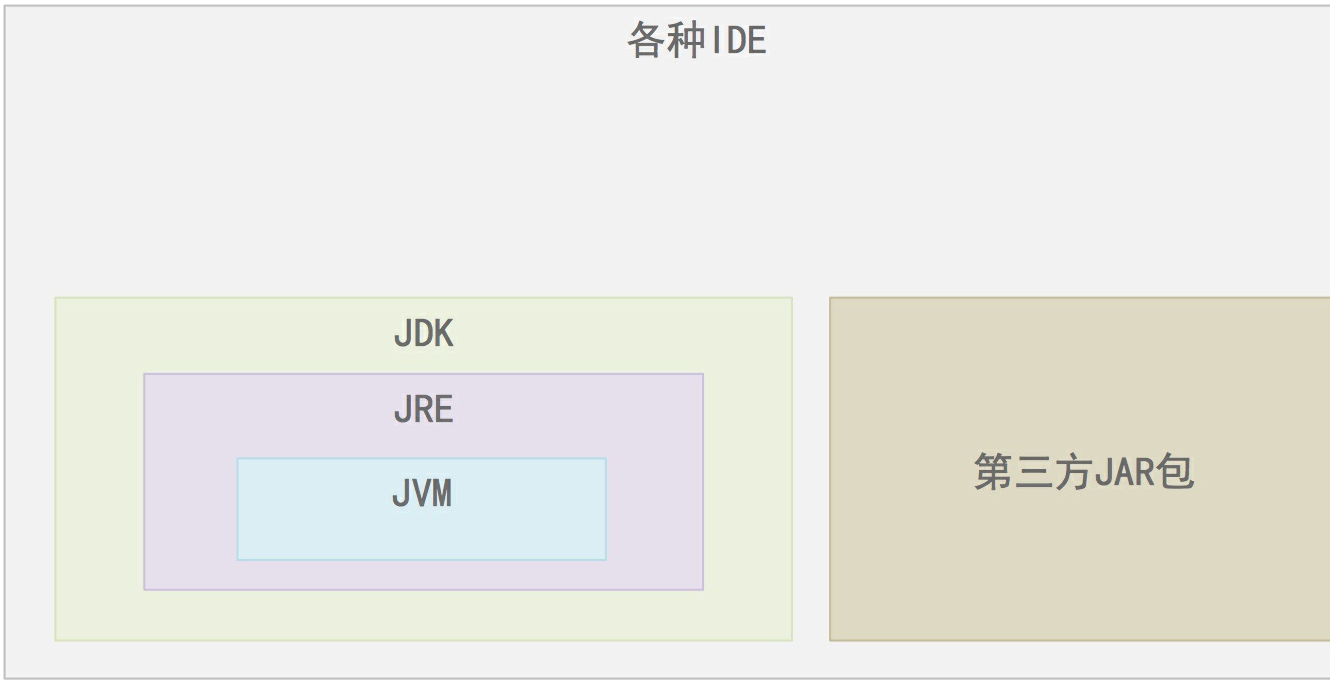
Java是一种开发语言（核心特点：跨平台，面向对象，名称由来看这里：J2EE里面的2是什么意思），对于开发者来讲，Java基本等于Jdk。

Jdk的版本介绍看这里：[Java都有那些版本][3]

开发人员一般通过IDE（Eclipse、NetBeans、JBuilder等）编写、编译Java代码（在远古没有IDE的时代，都是用文本编辑器编写，使用javac编译），在这个过程中，会使用到Jdk与第三方Jar包（Jar包即一组编译后的类打成的压缩包，可以使用解压工具解压成文件结构）。
想知道Java能用来干什么吗：[Java都能干什么][4]

[4]:

具体关系图：



其中，Jdk中包含Jre，在Jdk的安装目录下有一个名为jre的目录，里面有两个文件夹bin和lib，在这里可以认为bin里的就是Jvm，lib中则是Jvm工作所需要的类库，而Jvm和lib和起来就称为Jre。

2.什么是Jdk

即Java Development Kit，是针对Java开发人员的产品，是整个Java的核心。

想要安装Jdk看这里：[如何安装和配置Jdk][5]

[5]:

包括：Java运行环境Jre、Java工具(javac/java/jdb等)和Java基础类库(Java API，rt.jar等)。

3.什么是Jre

即Java Runtime Environment，是运行Java程序所须环境的集合，包含Jvm标准实现及Java核心类库。

与大家熟知的Jdk不同，Jre是Java运行环境，并不是一个开发环境，所以没有包含任何开发工具（如编译器和调试器），只是针对于使用Java程序的用户，只有通过它，Java的开发者才得以将自己开发的程序发布到用户手中，让用户使用。
运行Java程序一般都要求用户的电脑安装Jre；没有jre，java程序无法运行；而没有java程序，jre就没有用武之地。

包括：虚拟机Jvm，运行类库(runtime class libraries)和启动器(Java application launcher)。

4.什么是Jvm

即Java Virtual Machine，我们常说的Java虚拟机， 是整个Java实现跨平台最核心的部分，能够运行通过Java语言编写的应用程序。

所有的Java程序会首先被编译为.class的类文件，Jvm虚拟机可以执行这种编译后的类文件，也就是说class并不直接与机器的操作系统相对应，而是经过虚拟机间接与操作系统交互，由虚拟机将程序解释为目标代码（不同操作系统不同），给本地系统执行。

Jvm屏蔽了与具体操作系统平台相关的信息，使得Java程序只需生成在Java虚拟机上运行的目标代码，就可以在多种平台上不加修改地运行。
只有Jvm还不能成class的执行，因为在解释class的时候Jvm需要调用解释所需要的类库lib，即Jre中的lib类库，单独的Jvm没有任何作用。

]

[

Java千百问_01基本概念（002）_Java都有那些版本

,

点击进入_更多_Java千百问-基本概念

1、Java都有那些版本

Java最初由sun公司出品，2009年被orcale公司（即甲骨文）收购，它的版本体系分为两个纬度，纵向和横向。
纵向的版本即为我们常说的Jdk版本，通过近20年的时间，从1996年正式发布1.0版本，发展到2014年的8.0版本。
横向的版本即为我们所说的Java体系，从Java 2.0开始有所区分。

2、什么是Java体系

Java SE（J2SE，Java2 Platform Standard Edition，标准版）
Java EE（J2EE，Java 2 Platform Enterprise Edition，企业版）
Java ME（J2ME，Java 2 Platform Micro Edition，微型版）

3、什么是Java SE

以前称为J2SE。为什么叫这么奇怪的名字？请看这里：[J2SE里面的2是什么意思][2]
[2]:

它允许开发和部署在桌面、服务器、嵌入式环境和实时环境中使用的Java应用程序。
Java SE包含了支持Java Web服务开发的类，并为Java Platform，Enterprise Edition（Java EE）提供基础。

4.什么是Java EE

以前称为J2EE。

企业版本帮助开发和部署可移植、健壮、可伸缩且安全的服务器端Java应用程序。
Java EE是在Java SE的基础上构建的，它提供Web服务、组件模型、管理和通信API，可以用来实现企业级的面向服务体系结构（service-oriented architecture，SOA）和Web 2.0应用程序。

5、什么是Java ME

以前称为J2ME。

Java ME为在移动设备和嵌入式设备（比如手机、PDA、电视机顶盒和打印机）上运行的应用程序提供一个健壮且灵活的环境。
Java ME包括灵活的用户界面、健壮的安全模型、许多内置的网络协议以及对可以动态下载的连网和离线应用程序的丰富支持。

P.S.
基于Java ME规范的应用程序只需编写一次，就可以用于许多设备，而且可以利用每个设备的本机功能。

6、J2SE、J2EE、J2ME三者有什么关系

J2EE包含J2SE;
J2ME包含J2SE的核心类，但新添加了一些专有类。

]

[

Java千百问_01基本概念（003）_J2EE里面的2是什么意思

,

点击进入_更多_Java千百问-基本概念

1、J2EE里面的2是什么意思

J2SE，J2SE，J2ME中2的含义要追溯要1998年。
1998年Java 1.2版本发布，1999年发布Java 1.2的标准版，企业版，微型版三个版本，为了区分这三个版本，分别叫做Java2SE，Java2EE，Java2ME，简称J2SE，J2EE，J2ME。故，2的含义为1.2版本。
但是，这种叫法已经在2005年Java 1.6发布后取消，J2EE更名为Java EE，J2SE更名为Java SE，J2ME更名为Java ME。
所以，现在的J2EE等叫法是05年以前老一辈的叫法。想知道Java都有哪些版本吗？看这里：Java都有那些版本

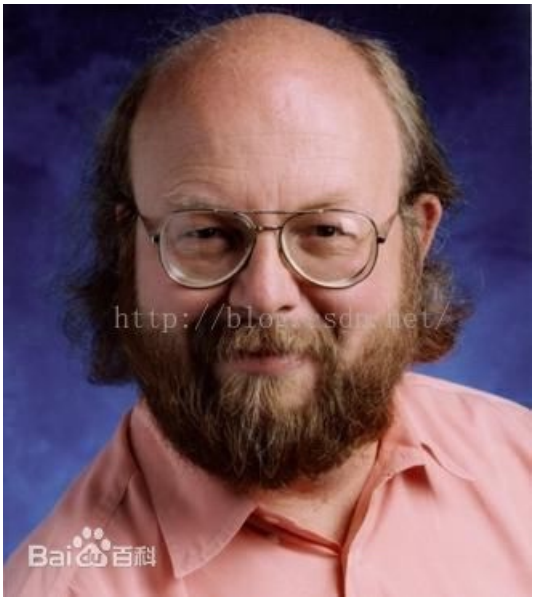
2、Java的名称是怎么来的

Java是印度尼西亚爪哇岛的英文名称，因盛产咖啡而闻名。
Java语言中的许多库类名称，很多与咖啡有关：如JavaBeans（咖啡豆）、NetBeans（网络豆）以及ObjectBeans(对象豆）等等。
sun和java的标识也正是一杯正冒着热气的咖啡：

P.S.
另一中说法是：Java源自是Java的几个主要开发人员名字的组合：**J**ames Gosling（詹姆斯·高斯林）**A**rthur**V**an Hoff（阿瑟·凡·霍夫）**A**ndy Bechtolsheim（安迪·贝克托克姆），
或“**J**ust**A**nother**V**ague**A**cronym”（只是另一个含糊的缩写）。
但比较可信的说法还是这群人出于对咖啡的喜爱，所以以Java咖啡来命名。类文件的前四个字节如果用十六进制阅读的话，分别为“CA FE BA BE”，就会拼出两个单词“CAFE BABE”（咖啡宝贝）。

3、Java是谁搞出来的

JGosling（詹姆斯·高斯林）
这个家伙被成为Java之父，是sun公司Java项目的核心成员，不过这显然不是他一个人的功劳。


http://blog.csdn.net/
Bai百科

]

[

Java千百问_01基本概念（004）_Java都能做些什么

,

点击进入_更多_Java千百问-基本概念

1、Java都能做些什么

在讨论Java能干什么之前，我们要说一下软件系统的体系结构。

你不知道Java是什么？请看这里：什么是Java

2、什么是纯C架构系统

完全脱离网络就可独立使用的软件系统（即客户端C，client），这类软件不受限于网络，只依赖于操作系统。
如：PC上的word、视频播放器等软件。

3、什么是C/S架构系统

拥有自己独立的一个或多个服务端系统（即服务端S，server），每个使用者拥有自己独立的客户端软件（即C，client），客户端与服务端通过网络进行数据的交互。

P.S.
所谓服务端，是可以通过网络访问的，接受/处理客户端数据的，为客户端提供数据的服务器+Web服务。
所谓客户端，即可以与指定服务端通过网络交互数据的纯C架构软件。
如：智能用电系统（C端：每户的智能电表；S端：电力数据管理系统）。

4、什么是B/S架构系统

拥有自己独立的一个或多个服务端系统（即S，server），每个使用者通过浏览器（即B，browser）与服务端进行数据的交互。

P.S.
如果把浏览器作为客户端的话，B/S架构即是一种特殊的C/S架构系统。
如：各大电商。

5、Java都能干什么

了解了以上概念，我们回到主题来看看Java都能干什么。
目前，Java的应用十分广泛，除了计算机底层开发，理论上其他均可以使用Java开发（当然排除那些垄断的企业，比如我们的大苹果）。

主要包括如下几个方面：

客户端软件
包括PC软件（包括windows、mac、linux等多操作系统上的软件）、软件插件、企业级应用的客户端
例如：我们熟悉的eclipse大部分都是使用java编写的，以及eclipse的各种插件
这类应用主要使用Swing、AWT或者SWT（前两者均包含在JDK中，后者是IBM的第三方库）开发。
下面是使用Swing+AWT开发的记事本截图（运行在mac系统中）：

.jpg)

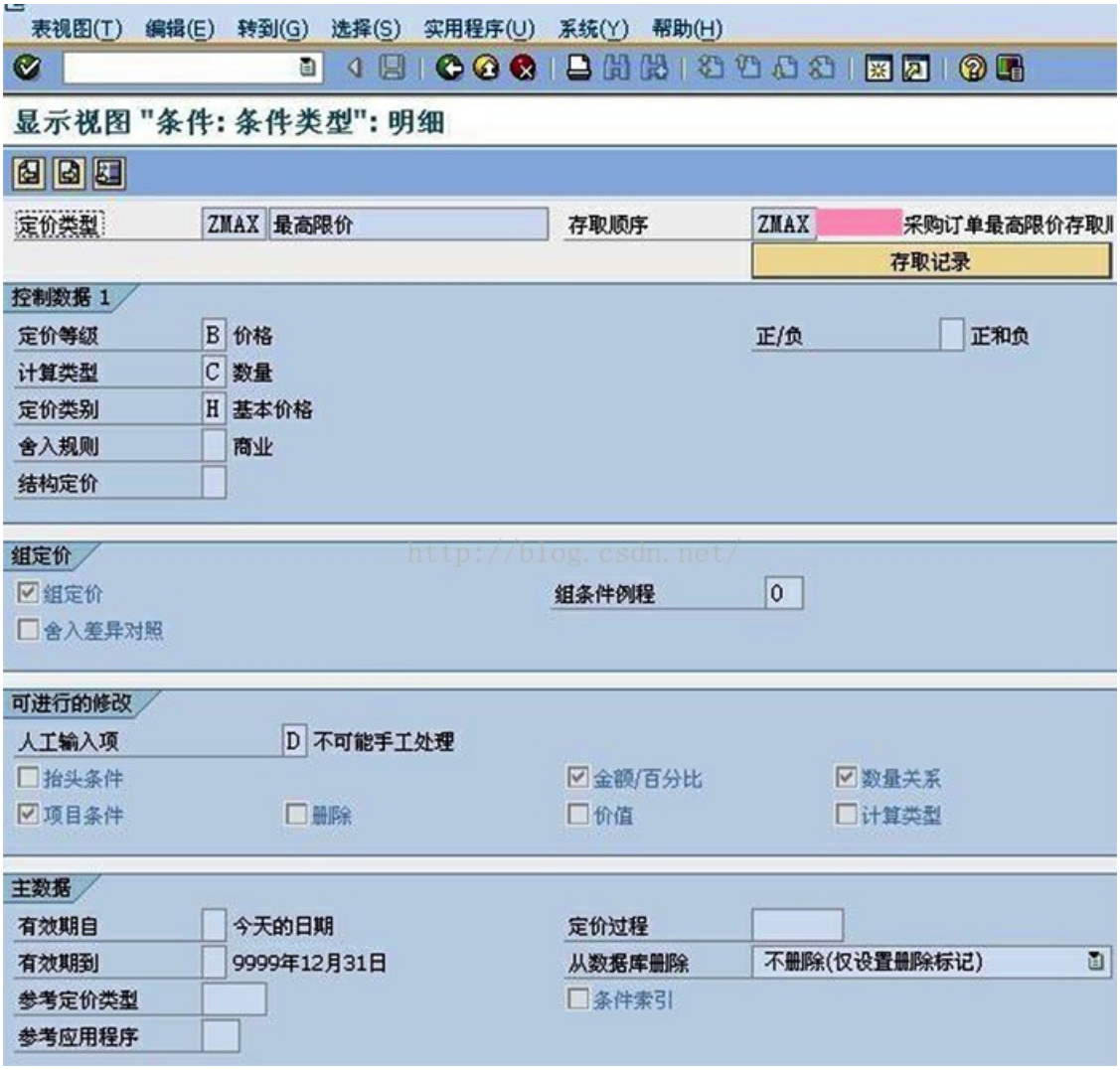
企业级应用

C/S架构系统的服务端、B/S架构系统均可以使用Java进行开发。
例如：B/S架构：ERP系统、C/S架构：医院管理系统（每个医生的pc上都会安装对应客户端）。

P.S.

大部分医院采用C/S架构系统，主要是因为医学的特殊性，例如需要展示x光片、B超视频以及其他浏览器无法很好支持的特殊功能。当然，这种趋势在不久的将来可能会改变。

下面是SAP公司开发的ERP系统截图：



web应用
纯B/S架构系统，面对人群不是企业而是个人，现在很大一部分互联网企业的平台均是使用Java开发。

例如：个大电商、论坛、O2O服务平台。

这个就不举具体的例子了，避免做广告＝。＝。

手机应用
我们熟悉的手机android系统便是由Java开发的，android系统的应用软件，绝大部分都是由Java开发。

P.S.
当然也有例外，目前有一部部分android游戏便不是Java实现，而是通过框架使用C++或者其他语言开发的，常见的有cocos2dx。

例如：android系统的大部分软件应用。

这里也不举例了。

]

[

Java千百问_01基本概念（005）_如何安装和配置Jdk

,

点击进入_更多_Java千百问

如何安装和配置Jdk

想要安装java，也就是我们所知的Jdk，需要先去官网下载。

在oracle的Java官网可以下载到任意版本的Jdk，我们可以下载最新版本的安装包（也可以下载绿色版，不过需要自行配置环境变量）。

安装之后，我们就可以使用Java开发程序了。

这里要说的是，如果自行配置环境变量，针对不同的操作系统配置方式差异较大。

1.windows如何配置Java环境变量

第一步

右击"我的电脑"→属性→高级系统设置→高级→环境变量→系统变量→新建变量（变量名称：JAVA_HOME，路径：C:\Java\jdk1.8.0，路径自行修改）。

第二步

然后点击"系统变量"→找到"Path"变量→编辑

在变量值最后输入 %JAVA_HOME%\bin;%JAVA_HOME%\jre\bin;

第三步

系统变量→新建CLASSPATH变量

变量值填写.;%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar（注意最前面有一点）

2.mac/linux如何配置Java环境变量

配置方法多种多样，这里只介绍最为常用，最为简单的办法。

第一步

打开终端，执行（显示隐藏文件：defaults write com.apple.finder AppleShowAllFiles Yes && killall Finder; 隐藏：defaults write com.apple.finder AppleShowAllFiles No && killall Finder）：

```
open ~/.bash_profile
```

系统会打开编辑器，可以添加环境变量。添加如下内容（其中路径自行修改）：

```
JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_40.jdk/Contents/Home

PATH=$PATH:$JAVA_HOME/bin:

export PATH
```

```
export JAVA_HOME
```

第二步

保存之后，再执行：

```
source ~/.bash_profile
```

即可使配置生效

3.如何判断Jdk是否正常安装

打开终端/cmd，输入：

```
java -version
```

如果返回对应Jdk版本号，即说明安装成功，可以正常使用了。

点击进入ooppookid的博客

]

[

Java千百问_01基本概念（006）_线程和进程有什么区别

,

点击进入_更多_Java千百问

1、进程是什么

进程（process）是具有一定独立功能的程序，操作系统利用进程把工作划分为一些功能单元。

进程是进行资源分配和调度的一个独立单位。它还拥有一个私有的虚拟地址空间，该空间仅能被它所包含的线程访问。

一个应用程序（application）是由一个或多个相互协作的进程组成的。例如，Visual Studio开发环境就是利用一个进程编辑源文件，并利用另一个进程完成编译工作的应用程序。

2、线程是什么

线程（thread）是进程中所包含的一个或多个执行单元。它只能归属于一个进程并且只能访问该进程所拥有的资源。它进程中执行运算的最小单位，是进程中的一个实体，是被进程独立调度和分派的基本单位。

线程自己不拥有系统资源，只拥有一点在运行中必不可少的资源（计数器、寄存器和栈），但它可与同属一个进程的其它线程共享进程所拥有的全部资源。一个线程可以创建和撤消另一个线程，同一进程中的多个线程之间可以并发执行。

当操作系统创建一个进程后，该进程会自动申请一个名为主线程（首要线程）的线程。主线程将执行运行时宿主，而运行时宿主会负责载入CLR（公共语言运行库）。

3、线程和进程有什么关系以及区别？

首先，进程和进程如同列车和车厢，没有可比性，但是他们有一定的相关性：

1. 一个线程只能属于一个进程，而一个进程可以有多个线程，但至少有一个线程。
2. 资源分配给进程，同一进程的所有线程共享该进程的所有资源。
3. 虚拟机分给线程，即真正在虚拟机上运行的是线程。
4. 线程在执行过程中，需要协作同步。不同进程的线程间要利用消息通信的办法实现同步。

如果非要比较进程与线程的区别，可以从以下几个方面来看：

1. 调度
   线程作为调度和分配的基本单位，进程作为拥有资源的基本单位

2. 并发性
   不仅进程之间可以并发执行，同一个进程的多个线程之间也可并发执行

3. 拥有资源
   进程是拥有资源的一个独立单位，线程不拥有系统资源，但可以访问隶属于进程的资源.

4. 系统开销
   在创建或撤消进程时，由于系统都要为之分配和回收资源，导致系统的开销明显大于创建或撤消线程时的开销。

]

[

Java千百问_01基本概念（007）_线程的状态有哪些

,

点击进入_更多_Java千百问

1、线程的状态有哪些

在java中java.lang.Thread类有一个变量threadStatus，标示了该线程的当前状态，它是一个int类型，但是对应的get方法返回值是一个枚举（Thread的内部类），源码如下：

```
public enum State {
        /**
         * Thread state for a thread which has not yet started.
         */
        NEW,

        /**
         * Thread state for a runnable thread.  A thread in the runnable
         * state is executing in the Java virtual machine but it may
         * be waiting for other resources from the operating system
         * such as processor.
         */
        RUNNABLE,

        /**
         * Thread state for a thread blocked waiting for a monitor lock.
         * A thread in the blocked state is waiting for a monitor lock
         * to enter a synchronized block/method or
         * reenter a synchronized block/method after calling
         * {@link Object#wait() Object.wait}.
         */
        BLOCKED,

        /**
         * Thread state for a waiting thread.
         * A thread is in the waiting state due to calling one of the
         * following methods:
         * <ul>
         *   <li>{@link Object#wait() Object.wait} with no timeout</li>
         *   <li>{@link #join() Thread.join} with no timeout</li>
         *   <li>{@link LockSupport#park() LockSupport.park}</li>
         * </ul>
         *
         * <p>A thread in the waiting state is waiting for another thread to
         * perform a particular action.
         *
         * For example, a thread that has called <tt>Object.wait()</tt>
         * on an object is waiting for another thread to call
         * <tt>Object.notify()</tt> or <tt>Object.notifyAll()</tt> on
         * that object. A thread that has called <tt>Thread.join()</tt>
         * is waiting for a specified thread to terminate.
         */
        WAITING,

        /**
         * Thread state for a waiting thread with a specified waiting time.
         * A thread is in the timed waiting state due to calling one of
         * the following methods with a specified positive waiting time:
         * <ul>
         *   <li>{@link #sleep Thread.sleep}</li>
         *   <li>{@link Object#wait(long) Object.wait} with timeout</li>
         *   <li>{@link #join(long) Thread.join} with timeout</li>
         *   <li>{@link LockSupport#parkNanos LockSupport.parkNanos}</li>
         *   <li>{@link LockSupport#parkUntil LockSupport.parkUntil}</li>
```

```
         * </ul>
         */
        TIMED_WAITING,

        /**
         * Thread state for a terminated thread.
         * The thread has completed execution.
         */
        TERMINATED;
    }
```

可以看到，线程包含6个可见状态：NEW、
RUNNABLE、BLOCKED、WAITING、TIMED_WAITING、TERMINATED。
我们还可以通过JavaVisualVM的线程监控可以看到，它会监控线程的运行（对应RUNNABLE）、休眠（对应NEW）、等待（WAITING+TIMED_WAITING）、驻留（BLOCKED）、监视状态、已完成（TERMINATED）六种状态。

了解监控JVM看这里：如何监控jvm的运行情况

对于State枚举，源码注释写的很清楚，这里我们具体看一下：

NEW
A thread that has not yet started is in this state.
一个还没有开始的线程就处于这种状态。

表明线程尚未开始。

RUNNABLE
A thread executing in the Java virtual machine is in this state.
一个在JVM中执行的线程处于这种状态。

可运行线程。一个线程在Java虚拟机处于可运行状态，即它在等待其他资源或操作系统的处理，一旦获取到CPU资源进行了执行，则进入人们所说的子状态RUNNING状态（这个状态JVM并不关心，我们也不是特别关注，一般的JVM监控工具都不会统计这种状态）。

BLOCKED
A thread that is blocked waiting for a monitor lock is in this state.
一个被阻塞的线程就处于这种状态，它正在等待监视器锁。
出于某种原因，比如等待用户输入等而让出当前的CPU给其他的线程执行时，会进入BLOCKED状态。
BLOCKED状态的线程会一直等待监视器锁，而后执行synchronized代码块/方法。或者在调用Object.wait()后，执行synchronized代码块/方法。

WAITING
A thread that is waiting indefinitely for another thread to perform a particular action is in this state.
一个线程正在无限期地等待另一个线程来唤醒是在这种状态下。

通过以下方法进入WAITING状态:

1. 调用Object.wait()且没有超时
2. 调用Thread.join()且没有超时
3. 调用LockSupport.park(Object)

一个线程处于WAITING状态需要由于另一个线程激活。例如，一个线程执行Object.wait()后会等待另一个线程调用对象Object.notify()或Object.notifyAll()。

一个线程调用了另一个线程的Thread.join()方法，则在另一个线程执行后才会继续执行（join方法可以指定延迟执行时间）。

TIMED_WAITING
A thread that is waiting for another thread to perform an action for up to a specified waiting time is in this state.
一个线程在一个时间阈值内等待另一个线程来唤醒就处于这种状态，达到阈值则回到RUNNABLE状态。

通过以下方法进入TIMED_WAITING状态:

1. 调用Thread.sleep()
2. 调用Object.wait()超过时间阈值
3. 调用Thread.join()超过时间阈值
4. 调用LockSupport.parkNanos(Object，long)
5. 调用LockSupport.parkUntil(Object，long)

TERMINATED
A thread that has exited is in this state.
一个线程退出就处于这种状态。

线程执行完成就会变为这个状态。

]

[

Java千百问_01基本概念（008）_jar是什么

,

点击进入_更多_Java千百问

1、jar是什么

JAR（Java Archive）是一个独立于平台的文件格式。可以包含多个文件，若干Java applet及其必要的组件（.class文件、图片和声音等）可以被打包在一个JAR文件中。

了解JavaApplet看这里：JavaApplet是什么

JAR文件格式以各种操作系统均能够压缩/解压的zip文件格式为基础。与zip不同的是，jar文件不仅仅用于压缩和发布，而且还用于封装库、组件和插件以及部署程序，并可被像编译器和JVM这样的工具直接使用。

在JAR中可以包含特殊的文件，例如：manifests和部署描述符，用来指示工具如何处理特定的jar。

使用jar的好处在于能够提高网络传输的速度，并且JAR还支持压缩，降低了文件的大小，进一步提高了传输效率。

此外，它是完全可扩展的，对应的API在java.util.jar包中。了解java.util.jar包看这里：[java.lang.ref包有什么用][2]

2、jar和zip有什么区别

jar文件格式提供了许多优势和功能，其中很多是传统的压缩格式如zip或者rar所没有的。它们之间的区别和联系主要包括：

1. 安全性
   jar文件可以加上数字化签名。这样一来，能够识别签名的工具就可以有选择地为用户授予软件安全特权，这是zip做不到的，数字签名还可以检测代码是否被篡改过。

2. 减少下载时间
   如果一个applet捆绑到一个jar中，那么浏览器就可以在一个HTTP请求中下载这个applet的类文件和相关的资源，而不是对每一个文件打开一个新连接。

3. 压缩
   同zip一样，jar格式也允许您压缩文件以提高存储效率。

4. 平台扩展
   Java扩展框架（Java Extensions Framework）提供了向Java核心平台添加功能的方法，这些扩展是用jar打包的。

5. 包密封性
   存储在jar文件中的包可以进行密封，以增强版本一致性和安全性。密封一个jar包意味着jar包中的所有类都必须在同一jar文件中。

6. 包版本控制
   一个jar可以包含有关它文件的数据，如厂商和版本信息等。

7. 可移植性
   处理jar的机制是Java平台核心API的标准部分，跨平台可以毫无问题的使用。

]

[

Java千百问_01基本概念（009）_CLASSPATH是什么

,

点击进入_更多_Java千百问

1、CLASSPATH是什么

CLASSPATH是Java解释器中用来指定搜索包路径的集合，Java解释器是这样工作的：

1. Java解释器找到环境变量CLASSPATH（将Java或者具有Java解释能力的工具，如浏览器，安装到机器中时，通过操作系统环境变量进行设定）。
2. CLASSPATH包含了一个或多个目录，它们作为一种特殊的“根路径”使用，从这里展开对.class文件的搜索。从那个根开始，解释器会寻找包名，并将每个点号（句点）替换成一个斜杠，从而生成从CLASSPATH根开始的每一个路径名（所以package foo.bar.baz 会变成foo\bar\baz 或者foo/bar/baz；具体是正斜杠还是反斜杠由操作系统决定）。
3. 将这些路径名连接到一起，成为CLASSPATH内的各个条目（入口）。以后搜索.class文件时，就可从这些地方开始查找与准备创建的类名对应的名字。
4. 当然Java解释器也会搜索一些标准目录（Java解释器所在的目录）。

通过上面的解释，我们可以了解到CLASSPATH的作用。简单来说，就是告诉java我们所使用的class在什么地方：

当你写下improt java.util.*时，解释器面对import关键字时，就知道你要引入java.util这个package中的类。但是解释器如何知道你把这个package放在哪里？所以你首先得告诉编译器这个package的所在位置，就是通过设置CLASSPATH，如果java.util这个package在c:\jdk\目录下，你得把c:\jdk\这个路径设置到CLASSPATH中去。当编译器面对import java.util.*这个语句时，它先会查找CLASSPATH所指定的目录，并检视子目录java\util是否存在，然后找出名称吻合的已编译文件（.class文件）。

2、如何设置CLASSPATH

不同的环境设置CLASSPATH也不尽相同：

1. 通过windows系统中系统环境变量设置CLASSPATH，这种设置是针对运行在这个系统上的java应用程序。通过“系统属性->高级->环境变量->系统变量”，新建“classpath=;%JAVA_HOME%\lib\tools.jar;%JAVA_HOME%\lib\dt.jar;”，其中%JAVA_HOME%是指引用JAVA_HOME环境变量，一般指定为jdk的目录：“JAVA_HOME=D:\Program Files\Java\jdk1.6.0_10”
2. 通过eclipse设置项目的CLASSPATH，项目右键->Properties->Java Build Path，这种设置只针对指定的项目生效（会在项目中生成.classpath文件）。

]

[

Java千百问_01基本概念（010）_Solaris操作系统是什么

,

点击进入_更多_Java千百问

1、Solaris操作系统是什么

Solaris是Sun公司研发的计算机操作系统。它是UNIX操作系统的衍生版本之一。 目前Solaris属于混合开源软件。2005年6月14日，Sun公司将正在开发中的Solaris 11的源代码开放，这版本就是OpenSolaris。

Sun的操作系统最初叫做SunOS，由于Sun的创始人之一Bill Joy来自于U.C.Berkeley，因此SunOS主要是基于BSD UNIX。从SunOS 5.0开始，SUN的操作系统开始转向System V Release 4，并且有了新的名字叫做Solaris 2.0。Solaris 2.6以后，SUN删除了版本号中的"2"，因此，SunOS 2.10就叫做Solaris 10。

从版本10开始，Solaris修改了其许可证，使产品能免费应用于任何系统。Solaris的早期版本后来又被重新命名为Solaris 1.x。所以"SunOS"这个词被用做专指Solaris操作系统的内核，因此Solaris被认为是由SunOS，图形化的桌面计算环境，以及它网络增强部分组成。

2009年，SUN公司被Oracle收购，Solaris和OpenSolaris一并归Oracle所有，OpenSolaris项目已经终结，其基金会解散。一个替代的项目OpenIndiana成立。该产品又恢复了私有性质，遵循一份限制许可证。

2014年5月6日， Oracle发布了Oracle Solaris 11.2，这是一款基于云的操作系统。新版本Solaris从操作系统产品上升到全面成熟的云平台。其中Solaris与OpenStack的集成，将改善虚拟化以及管理其他Hypervisor的能力。另外，Solaris 11.2还将与Oracle数据库完全集成。Solaris 11中提供的企业云基础架构能力之外，Solaris 11.2还为OpenStack提供了构建基于Solaris的云基础架构的方法。在Solaris 11.2基础包中，包含完整的OpenStack发行版。

Solaris支持多种系统架构：SPARC, x86 and x64。在版本2.5.1的时候，Solaris曾经一度被移植到PowerPC架构，但是后来又在这一版本正式发布时被删去。与Linux相比，Solaris可以更有效地支持对称多处理器、即SMP架构。Sun同时宣布将在Solaris 10的后续版本中提供Linux运行环境。允许Linux二进制程序直接在Solaris x86和x64系统上运行。

2、Solaris的安全特性有那些

Solaris具有以下安全特性：

1. 基于标准的密码架构(Standards-based Cryptographic Framework)
2. 综合性防火墙(Integrated Firewall)
3. 拥有安全执行的认证(Verification of Secure Execution)
4. 基础稽查与报告工具(BART：Basic Audit and Reporting Tools)
5. 提供仅有最小特权的安全性服务(Services Secured With Least Privileges)
6. 灵活的企业认证(Flexible Enterprise Authentication)
7. 安全的数据中心整合(Secure Data Center Consolidation)
8. 中央托管的用户权限管理(URM:Centrally Managed User Rights Management)
9. 最小化的安装选项(Minimized Install Option)
10. 精细过程的权限管理(Fine grained Process Rights Management)

]

[

Java千百问_01基本概念（011）_JavaApplet是什么

,

点击进入_更多_Java千百问

1、JavaApplet是什么

Applet是采用Java编程语言编写的小应用程序，该程序可以包含在HTML（标准通用标记语言的一个应用）页中，与在页中包含图像的方式大致相同。Applet不需要main()方法，由Web浏览器中内嵌的Java虚拟机调用执行。

在Java Applet中，可以实现图形绘制、字体和颜色控制、动画和声音的插入、人机交互及网络交流等功能。Applet还可以使用抽象窗口工具箱（Abstract Window Toolkit， AWT）的窗口环境开发工具。 AWT利用用户计算机的GUI元素，可以建立标准的图形用户界面，如窗口、按钮、滚动条等等。

2、applet如何运行

使用Applet编写的一些小应用程序，都是直接嵌入到网络页面中，由支持Java的浏览器解释执行，并能够产生特殊效果。

在含有Applet网页的HTML文件代码中，会带有applet（HTML5中使用object标签））标签。当支持Java的网络浏览器遇到这个标签时，就将下载相应的小应用程序代码并在本地计算机上执行该Applet。HTML文件中关于Applet的信息至少应包含以下三点：字节码文件名(编译后的Java文件，以.class为后缀)

、字节码文件的地址、在网页上显示Applet的方式。

它可以大大提高Web页面的交互能力和动态执行能力。包含Applet的网页被称为Java-powered页，可以称其为Java支持的网页。由于Applet是在用户的计算机上执行的，所以它的执行速度不受网络带宽的限制，用户可以更好地欣赏网页上Applet产生的多媒体效果。

3、Applet的安全限制是什么

因为applet是从远端服务器上下载并且在本地执行，所以安全性就显得格外重要。我们通过限制applet在沙箱（applet的运行环境）中运行，保证了对本地系统而言applet是安全的。applet在沙箱中运行时，要注意以下几点：

1. 不能运行任何本地可执行程序。
2. 除了存放下载的applet的服务器外，applet不能和其它主机进行通信。
3. 不能对本地文件系统进行读写。

4、Applet的生命周期是什么

1. applet初始化init()
   当浏览器加载applet，进行初始化的时候调用该方法。

2. 开始执行start()
   在init()方法之后调用。当用户从其它页面转到包含applet的页面时，该方法也被调用。

3. 停止stop()
   在用户离开包含applet的页面时被调用。

4. 销毁destroy()
   当applet不再被使用，或浏览器退出的时候，该方法被调用。

了解如何编写applet看这里：[如何编写applet程序][2]

]

[

Java千百问_01基本概念（012）_Socket是什么

,

点击进入_更多_Java千百问

1、什么是Socket

Socket也称作“套接字”。网络上的两个应用程序通过一个双向的通讯连接实现数据的交换，这个双向链路的一端称为一个Socket，所以Socket都是成对出现的。Socket通常用来实现客户方和服务方的连接，一个Socket由一个IP地址和一个端口号唯一确定。

Socket是TCP/IP协议的一个十分流行的解决方案，是支持TCP/IP协议的网络通信的基本操作单元。Socket所支持的协议种类不只TCP/IP一种，只不过在Java环境下，Socket编程主要是指基于TCP/IP协议的网络编程。

了解TCP/IP协议看这里：[TCP/IP协议是什么][2]

Socket正如其英文原意那样，像一个插座，网络上某一个主机运行了多个应用，同时提供多种服务。每种服务都打开一个Socket，并绑定到一个端口上，不同的端口对应于不同的服务。这台主机就犹如布满各种插座的房间，每个插座有一个编号，不同的插座提供不同的服务：有的提供220伏交流电、有的提供宽带网络、有的提供有线电视节目。客户软件将插头插到不同编号的插座，就可以得到不同的服务。

2、Socket如何通讯

Socket步骤如下：Server端监听某个端口是否有连接请求，Client端向Server端发出连接请求，如果Server端监听到了Client端的请求，则向Client端发回接受消息。这样一个连接就建立起来了。Server端和Client端都可以通过Send，Write等方法与对方通信。

从上面的描述可以看出，Socket之间的连接过程可以分为三个步骤：

1. 服务器监听
   服务器端的Socket，它并不定位具体客户端的Socket，而是处于等待连接的状态，实时监控网络状态。

2. 客户端请求
   由客户端的Socket提出连接请求，要连接的目标是服务器端的Socket。为此，客户端的Socket必须首先描述它要连接的Socket，指出服务器端Socket的地址和端口号，然后就向服务器端Socket提出连接请求。

3. 连接确认
   当服务器端Socket监听到客户端Socket的连接请求时，它会响应客户端Socket的请求，建立一个新的线程，把服务器端Socket的描述发给客户端。一旦客户端确认了此描述，连接就建立好了。而服务器端套接字继续处于监听状态，继续接收其他客户端套接字的连接请求。

对于一个功能齐全的Socket，其工作过程需要包含以下四个步骤：

1. 创建Socket
2. 打开连接到Socket的输入/出流
3. 对Socket进行读/写操作
4. 关闭Socket

Socket服务端分为单线程Socket、多线程Socket两种。顾名思义，单线程Socket是指同时只能与一个客户端连接；多线程Socket是指能够同时与多个客户端连接。

了解如何编写单线程Socket程序看这里：[如何编写单线程Socket程序][3]
了解如何编写多线程Socket程序看这里：[如何编写多线程Socket程序][4]

]

[

Java千百问_01基本概念（013）_Socket、SocketChannel有什么区别

,

点击进入_更多_Java千百问

1、Socket、SocketChannel有什么区别

了解Socket看这里：Socket是什么
Socket、SocketChannel二者的实质都是一样的，都是为了实现客户端与服务器端的连接而存在的，但是在使用上，却有很大的区别。具体如下：

所属包不同
Socket在java.net包中，而SocketChannel在java.nio包中。

异步方式不同
从包的不同，我们大体可以推断出他们主要的区别：Socket是阻塞连接（当然我们可以自己实现非阻塞），SocketChannel可以设置非阻塞连接。
使用ServerSocket、Socket类时，服务端Socket往往要为每一个客户端Socket分配一个线程，而每一个线程都有可能处于长时间的阻塞状态中。过多的线程也会影响服务器的性能（可以使用线程池优化，具体看这里：如何编写多线程Socket程序）。而使用SocketChannel、ServerSocketChannel类可以非阻塞通信，这样使得服务器端只需要一个线程就能处理所有客户端socket的请求。

了解阻塞、非阻塞看这里：[阻塞、非阻塞有什么区别][3]

性能不同
一般来说使用SocketChannel会有更好的性能。其实，Socket实际应该比SocketChannel更高效，不过由于使用者设计等原因，效率反而比直接使用SocketChannel低。

使用方式不同
Socket、ServerSocket类可以传入不同参数直接实例化对象并绑定ip和端口，如：

```
Socket socket = new Socket("127.0.0.1", "8000");
ServerSocket serverSocket = new ServerSocket("8000");
```

而SocketChannel、ServerSocketChannel类需要借助Selector类控制，如：

```
Selector selector = Selector.open();
ServerSocketChannel serverChannel = ServerSocketChannel.open();
serverChannel.configureBlocking(false); // 设置为非阻塞方式,如果为true 那么就为传统的阻塞方式
serverChannel.socket().bind(new InetSocketAddress(port)); // 绑定IP 及 端口
serverChannel.register(selector, SelectionKey.OP_ACCEPT); // 注册 OP_ACCEPT事件
new ServerThread().start(); // 开启一个线程 处理所有请求
```

2、SocketChannel方式有什么核心类

下面是SocketChannel方式需要用到的几个核心类：

- ServerSocketChannel
  ServerSocket的替代类, 支持阻塞通信与非阻塞通信。
- SocketChannel
  Socket的替代类, 支持阻塞通信与非阻塞通信。
- Selector

为ServerSocketChannel监控接收客户端连接就绪事件, 为SocketChannel监控连接服务器读就绪和写就绪事件。

- SelectionKey
  代表ServerSocketChannel及SocketChannel向Selector注册事件的句柄。当一个SelectionKey对象位于Selector对象的selected-keys集合中时，就表示与这个SelectionKey对象相关的事件发生了。
  在SelectionKey类中有几个静态常量：
  SelectionKey.OP_ACCEPT，客户端连接就绪事件，等于监听serversocket.accept()，返回一个socket。
  SelectionKey.OP_CONNECT，准备连接服务器就绪，跟上面类似，只不过是对于socket的 相当于监听了socket.connect()。
  SelectionKey.OP_READ，读就绪事件, 表示输入流中已经有了可读数据, 可以执行读操作。
  SelectionKey.OP_WRITE，写就绪事件, 表示可以执行写操作。
  了解如何开发ServerSocketChannel看这里：[如何编写非阻塞SocketChannel程序][4]

]

[

Java千百问_01基本概念（014）_同步、异步有什么区别

,

点击进入_更多_Java千百问

1、同步、异步有什么区别

在进行网络编程时，我们通常会看到同步、异步、阻塞、非阻塞四种调用方式以及他们的组合。
了解阻塞、非阻塞看这里：阻塞、非阻塞有什么区别
其中同步方式、异步方式主要是由客户端（client）控制的，具体如下：

同步（Sync）

所谓同步，就是发出一个功能调用时，在没有得到结果之前，该调用就不返回或继续执行后续操作。
根据这个定义，Java中所有方法都是同步调用，应为必须要等到结果后才会继续执行。我们在说同步、异步的时候，一般而言是特指那些需要其他端协作或者需要一定时间完成的任务。
简单来说，同步就是必须一件一件事做，等前一件做完了才能做下一件事。

例如：B/S模式中的表单提交，具体过程是：客户端提交请求->等待服务器处理->处理完毕返回，在这个过程中客户端（浏览器）不能做其他事。

异步（Async）

异步与同步相对，当一个异步过程调用发出后，调用者在没有得到结果之前，就可以继续执行后续操作。当这个调用完成后，一般通过状态、通知和回调来通知调用者。对于异步调用，调用的返回并不受调用者控制。
对于通知调用者的三种方式，具体如下：

1. 状态
   即监听被调用者的状态（轮询），调用者需要每隔一定时间检查一次，效率会很低。

2. 通知
   当被调用者执行完成后，发出通知告知调用者，无需消耗太多性能。

3. 回调
   与通知类似，当被调用者执行完成后，会调用调用者提供的回调函数。

例如：B/S模式中的ajax请求，具体过程是：客户端发出ajax请求->服务端处理->处理完毕执行客户端回调，在客户端（浏览器）发出请求后，仍然可以做其他的事。

总结来说，同步和异步的区别：请求发出后，是否需要等待结果，才能继续执行其他操作。

]

[

Java千百问_01基本概念（015）_阻塞、非阻塞有什么区别

,

点击进入_更多_Java千百问

1、阻塞、非阻塞有什么区别

在进行网络编程时，我们通常会看到同步、异步、阻塞、非阻塞四种调用方式以及他们的组合。
了解同步、异步看这里：同步、异步有什么区别
其中阻塞方式、 非阻塞方式主要是针对服务端（server）的，具体如下：

阻塞（Block）

阻塞调用是指调用结果返回之前，当前线程会被挂起。挂起即线程进入非可执行状态，在这个状态下，cpu不会给线程分配时间片，即线程暂停运行。

了解线程的状态看这里：线程的状态有哪些
阻塞调用会让线程一直进行等待，当调用没有执行完就有另一次请求，这是会开启一个新的线程来执行，会占用更多的线程资源。

由于阻塞线程的大部分时间都浪费在等待请求上了，所以并不能处理过多的请求。

非阻塞（Unblock）

非阻塞是相对阻塞来说的，是指调用不会阻塞当前线程，而会立刻返回。

非阻塞是通过轮询不断去询问数据是否准备好了，如果准备好了主动获取数据。在这期间线程没有挂起。

非阻塞处理连接的线程数和请求数没有联系，也就是说很多个请求可以通过相对较少的线程进行处理。

总结来说， 阻塞和非阻塞的区别：请求发出后，没有数据到达时，是否立刻返回。

]

[

Java千百问_01基本概念（016）_32位和64位计算机有什么区别

,

点击进入_更多_Java千百问

1、32位和64位计算机有什么区别

我们通常说的32位、64位计算机是指计算机的CPU位数。当然很早还有8位、16位的CPU，以Intel的80x86系列来说，8位的8080，16位的8086、8088、80186、80286，而32位的CPU最早始于80386，64位就是大家熟悉的EM64T技术以及AMD的x86-64。当然不同的厂商间同位数的CPU内部有很大的区别，但是它们的核心都是一样：CPU处理能力为64位。

这个位数指的是CPU的通用寄存器（GPRs，General-Purpose Registers，寄存器可以简单理解为一个可以暂存指令、数据和地址的空间，CPU运算时的结果都会暂时放在这里）的指令集、寻址能力。

一般来说，相比较32位的CPU来说，64位CPU最为明显的变化就是寄存器和指令指针升级到64位、内存寻址能力提高到64位，还有其他变化例如增加了8个64位的通用寄存器。更高位数的CPU，可以进行更大范围的整数运算，同时可以支持更大的内存。具体如下：

- 从运算来说，32位处理器一次只能处理32位，也就是4个字节的数据，而64位处理器一次就能处理64位，即8个字节的数据。
- 从内存来说，传统32位处理器的寻址空间最大不足4G（理论上有2^32个物理地址），形成了运行效率的瓶颈。而64位的处理器在理论上则可以将近达到1700万个TB（2^64个，大到惊人）。

2、CPU位数大小有什么影响

一个简单的例子可以说明CPU位数的影响，对于16位CPU，指令集只能操作16bit数据和16bit地址。不同CPU的寄存器、指令集不同，要区别对待，这里以8086来说明。

将16bit数据放入寄存器中

例如：

```
MOV AX,1234H          ;向寄存器 AX 传入数据 1234H
MOV AH,56H              ;向寄存器 AX 的高 8 位寄存器 AH 中传入数据 56H
MOV AL,78H    ;向寄存器 AX 的低 8 位寄存器 AL 中传入数据 78H
```

这里要说明的一点，第一句我们向AX寄存器（累加寄存器）中存放了一个16bit的数1234H，但实际是AX由AH、AL两个寄存器组成，所以可以直接操作AH、AL这两个8位寄存器。
如果我们想在一个寄存器中存入一个超过16bit的数，在16位CPU下是不可能的。如果想处理16bit的数，只能借助其他寄存器，分段处理。

获取16bit内存地址中的数据

例如：

```
MOV  BX,1000H
MOV DS, BX  ;向DS段寄存器传入1000H，由于8086不支持直接将数据传入段寄存器，所以只能借助其他寄存器传值。
MOV AX,[1234H]     ;将内存地址1000H:1234H中的值读到AX寄存器中
```

这里要说明的一点，8086的物理地址支持每次传20位的地址，但是由于16位CPU的指令集只能支持16bit，最大的寻址空间理论值为2^16（64K），为了能够支持2^20个地址（1M），所以需要将分为段地址和偏移地址，表现形式如1000H:1234H。

了解CPU物理地址形成看这里：[CPU物理内存地址如何形成][2]

]

[

Java千百问_01基本概念（017）_内存物理地址在CPU中如何形成

,

点击进入_更多_Java千百问

1、什么是内存的物理地址

我们通过8086CPU来说明内存地址是如何形成的。

首先我们要了解物理地址，当CPU需要访问一个内存单元时，需要给出内存单元的地址，而每一个内存单元在物理内存空间中都有一个唯一的地址，即可以通过这个地址定位到内存单元，而这个地址即为物理地址。

CPU通过地址总线将一个内存单元的物理地址送入存储器，而后CPU便可以通过这个物理地址来访问这个物理地址所指向的内存单元了。

2、内存物理地址在CPU中如何形成

首先，我们知道8086CPU的地址总线是20根，即每次都可以传输20位的地址，从而寻址能力有2^20，也就是1MB的大小。但是，8086CPU的寄存器只有16位，也就是在8086CPU的内部，处理、传输、暂存的地址都只能是16位，即8086CPU不能完整的保存下一个物理地址（物理地址为20位）。

如果以最简单的方式（即直接用16位寄存器来保存物理地址）的话，寻址能力只有2^16，也就是64KB大小，难道8086CPU只能支持64KB大小的内存吗？
当然不是，8086CPU在这里采取了一定的措施，使其寻址能力达到1MB。

8086CPU在内部把两个16位的地址进行合成，从而形成一个20位的物理地址，8086CPU的寻址能力便可以达到1MB。
具体是如何将两个16位的地址合成为一个20位的物理地址的呢？
当CPU在访问内存时，其会使用一个16位的基地址，然后再使用一个16位的偏移地址，通过将两个地址传入8086CPU的地址加法器中进行合成，即可以构造出20位的物理地址。

至于合成的方式如下：
将段地址左移4位，形成基地址，然后将基地址和偏移地址相加便形成了20位的物理地址。如下图：

3、内存段是什么

其实在物理内存中是没有内存段这一概念的，内存段的概念来自于CPU中的段寄存器。

我们将若干个地址连续的内存单元看做是一个段，通过将一个段地址左移4位形成基地址，然后通过这个基地址来定位这个段的起始地址，最后再通过偏移地址便可以精确定位到段中的内存单元了。

由于内存段的起始地址是一个段地址左移4位，所以内存段的起始地址肯定是16的倍数。而且一个内存段内部的内存单元只能通过偏移地址来定位，而偏移地址为16位，所以一个段的长度也就是2^16也就是64KB的大小。

在编程时，可以将一段内存定义成为一个段，分为数据段，代码段，栈段这三种类型的段。具体如下：

1. 数据段
   存放我们所需要使用数据的内存段（当然段起始地址肯定是16的倍数，并且段长度<=64KB）。

2. 代码段
   存放代码（也就是指令）的内存段。

3. 栈段

我们将一段内存当做栈来使用就称为栈段。

一个简单的例子：

```
MOV BX, 1000H   ;
MOV DS, BX    ;向DS段寄存器传入1000H段地址。
MOV AX, [1234H];将内存地址1000H:1234H（即11234H）中的值读到AX寄存器中。
```

]