**[JAVA命令大全](http://www.cnblogs.com/langtianya/p/4208135.html)**

1、java.exe:  
======================  
运行java程序,这个相信每一位用Java的人知道了。  
  
2、javac.exe:  
======================  
编译的Java程序，生成.class文件  
  
3、javaw.exe：  
======================  
功 能:跟java命令相对的，可以运行.class文件,主要用来执行图形界面的java程序运行java命令时，会出现并保持一个console窗口,程 序中的信息可以通过System.out在console内输出，而运行javaw，开始时会出现console，当主程序调用之后，console就会 消失； javaw 大多用来运行GUI程序  
  
4、javah  
======================  
功能说明：C 头文件和 Stub 文件生成器。javah 从 Java 类生成 C 头文件和 C 源文件。这些文件提供了连接胶合，使 Java 和 C 代码可进行交互。  
  
语法：

javah [ 命令选项 ] fully-qualified-classname. . .  
javah\_g [ 命令选项 ] fully-qualified-classname. . .  
  
补充说明：  
  
 javah 生成实现本地方法所需的 C 头文件和源文件。C 程序用生成的头文件和源文件在本地源代码中引用某一对象的实例变量。.h 文件含有一个 struct 定义，该定义的布局与相应类的布局平行。该 struct 中的域对应于类中的实例变量。  
  
 头文件名以及在头文件中所声明的结构名都来源于类名。如果传给 javah 的类是在某个包中，则头文件名和结构名前都要冠以该包名。下划线 (\_) 用作名称分隔符。  
  
 缺省情况下，javah 为每个在命令行中列出的类都创建一个头文件，且将该文件放在当前目录中。用 -stubs 选项创建源文件。用 -o 选项将所有列出类的结果串接成一个单一文件。  
  
命令选项  
  
-o[输出文件] 将命令行中列出的所有类的头文件或源文件串接到输出文件中。-o 或 -d 两个选项只能选择一个。  
  
-d[目录] 设置 javah 保存头文件或 stub 文件的目录。-d 或 -o 两个选项只能选择一个。  
  
-stubs 使 javah 从 Java 对象文件生成 C 声明。  
  
-verbose 指明长格式输出，并使 javah 将所生成文件的有关状态的信息输出到标准输出设备中。  
  
-help 输出 javah 用法的帮助信息。  
  
-version 输出 javah 的版本信息。  
  
-jni 使 javah 创建一输出文件，该文件包含 JNI 风格的本地方法函数原型。这是缺省输出，所以 -jni 的使用是可选的。  
  
-classpath[路径] 指定 javah 用来查询类的路径。如果设置了该选项，它将覆盖缺省值或           CLASSPATH 环境变量。目录用冒号分隔。  
  
-bootclasspath[路径] 指定加载自举类所用的路径。缺省情况下，自举类是实现核心 Java 平台的类，位于 jrelibt.jar 和 jrelibi18n.jar 中。  
  
-old 指定应当生成旧 JDK1.0 风格的头文件。  
  
-force 指定始终写输出文件。  
  
5、javap  
======================  
功能说明：Java 类文件解析器。  
  
语法： javap [ 命令选项 ] class. . .  
  
补充说明：  
  
 javap 命令用于解析类文件。其输出取决于所用的选项。若没有使用选项，javap 将输出传递给它的类的 public 域及方法。javap 将其输出到标准输出设备上。  
  
命令选项  
  
-help 输出 javap 的帮助信息。  
  
-l 输出行及局部变量表。  
  
-b 确保与 JDK 1.1 javap 的向后兼容性。  
  
-public 只显示 public 类及成员。  
  
-protected 只显示 protected 和 public 类及成员。  
  
-package 只显示包、protected 和 public 类及成员。这是缺省设置。  
  
-private 显示所有类和成员。  
  
-J[flag] 直接将 flag 传给运行时系统。  
  
-s 输出内部类型签名。  
  
-c 输出类中各方法的未解析的代码，即构成 Java 字节码的指令。  
  
-verbose 输出堆栈大小、各方法的 locals 及 args 数。  
  
-classpath[路径] 指定 javap 用来查找类的路径。如果设置了该选项，则它将覆盖缺省值或 CLASSPATH 环境变量。目录用冒号分隔。  
  
- bootclasspath[路径] 指定加载自举类所用的路径。缺省情况下，自举类是实现核心 Java 平台的类，位于 jrelib  
  
t.jar 和 jrelibi18n.jar 中。  
  
-extdirs[dirs] 覆盖搜索安装方式扩展的位置。扩展的缺省位置是 jrelibext。  
  
6.jdb.exe  
======================  
功能：Java的一个命令行调试环境，既可在本地，也可在与远程的解释器的一次对话中执行。  
  
用法：  
  
jdb于本地机器中可用如下的命令启动：  
  
C:/>jdb classname  
  
当你使用-debug选项开始一个Java例程时， 必须提供给Jdb 一个密码， 这样 Jdb才能开始运转起来。下表包含了所有jdb命令。  
  
选项 功能  
  
catch calssID 为特定异常出口而中断  
  
classes 列出当前已知的类  
  
clear classID:line 清除一个断点  
  
cont 从断点处继续执行  
  
down[n frames] 下移一个线程的堆栈  
  
dump ID[ID...] 显示所有对象信息  
  
exit(或quit) 退出调试器  
  
help(或?)　 列出所有命令  
  
ignore classID 忽略特定的异常出口  
  
list[line number] 显示源代码  
  
load classbame 载入要调试的Java类  
  
locals 在当前堆栈帧中显示所有局部变量  
  
memory 报告内存使用情况  
  
methods classID 列出一个类的成员函数集  
  
print ID[ID...] 列出对象或域  
  
resume [threadID...] 恢复线程(默认情况恢复所有线程)  
  
run class [args] 开始执行已下载的Java类  
  
step 执行当前行  
  
stop in classID:method 在一成员函数中设一断点  
  
stop at classID:line 在一行设一断点 功能:  
  
suspend[threadID...] 停止一个线程(默认情况停止所有线程)  
  
hreads threadgroup 列出线程  
  
thread threadID 设置当前线程  
  
threadgroups 列出线程组  
  
threadgroup name 设置当前线程组  
  
up [n frames] 上移一个线程堆栈  
  
use [path] 显示或改变源程序路径  
  
where [threadID] or all 使一线程的堆线置空  
  
!! 重复上一次命令  
  
-host hostname 该命令告诉Jdb到哪里去建立远程运行的Java解释器对话过程  
  
-password password 本选项告诉Jdb 用哪个密码去与远程运行的Java 对话进程相连接。 密码 password是由运行带有-debug选项的Java解释器所提供的。  
  
7、javaws.exe  
======================  
功能:Java应用程序高速缓存查看器.  
  
用法:具有图形界面,可以直接进行操作.  
  
8、jconsole.exe  
======================  
功能:  
  
J2SE监测和管理控制台---一个同JMX兼容的监测Java虚拟机的图形工具。其能够监视当地或者远程的Java虚拟机。  
  
Java监视和管理控制台,JConsole使用JVM的可扩展性Java管理扩展(JMX)工具来提供关于运行于Java平台的应用程序的性能和资源消耗的信息。  
  
在J2SE 5.0软件中，你需要启动使用-Dcom.sun.management.jmxremote选项监控的应用程序。注意：在Java SE 6软件中，不再有这一要求。当启动该应用程序时，不需要特定的命令行选项。  
  
在生产系统中的应用,JConsole启动一个在被观察的Java程序的JVM内部的JMX代理。运行另外一部分代码仅有一点极微弱的影响-但是影响很小。

　另外，尽管JConsole在监视本地应用程序的开发和快速原型开发中很有用，但在实际的应用系统中不推荐使用。理由是，JConsole本身也消耗大

量的系统资源。我们推荐的方法是用远程监控来把JConsole应用程序与被监控的系统加以隔离。因此,对于应用系统来说,以远程模式使用

JConsole更好些。对于安全的远程监控来说,可以使用安全选项。  
  
用法:具有图形界面,可直接操作,查看摘要、内存、线程、类、MBean,VM的信息。  
  
9、jps.exe  
======================  
功能:  
  
java虚拟机进程状态工具---在目标系统上列出装备有HotSpot Java的虚拟机  
  
 启动了 Java 进程时，请使用 jps 工具来通过键入 jps 标识其 JVM。它向您提供一个正在运行的 Java 进程的列表，这些进程以其 JVM 作为 ID。  
  
 jps

相当于Solaris进程工具ps。更多的信息，请参考《jps-Java Virtual Machine Process Status

Tool》。不象"pgrep java"或"ps -ef | grep

java"，jps并不使用应用程序名来查找JVM实例。因此，它查找所有的Java应用程序，包括即使没有使用java执行体的那种（例如，定制的启动

器）。另外，jps仅查找当前用户的Java进程，而不是当前系统中的所有进程。  
  
用法:  
  
usage: jps [-help]  
  
   jps [-q] [-mlvV] [<hostid>]  
  
Definitions:  
  
<hostid>:    <hostname>[:<port>]  
  
10、jstat.exe  
======================  
功能:  
  
JVM监控工具,java虚拟机统计监视工具---附加到一个装备了HotSpot的java虚拟机上来采集并且记录性能统计情况  
  
 jstat

显示一个测量（instrumented）Java HotSpot虚拟机的性能统计信息（请参考《jstat-Java Virtual

Machine Statistics Monitoring Tool》）。有关于性能计数器的更详细的信息请参考《Code

sample-jvmstat 3.0》。  
  
用法:详细用法<http://java.sun.com/j2se/1.5.0/docs/tooldocs/share/jstat.html>  
  
11、jstatd.exe  
======================  
功能:  
  
 java虚拟机的jstat守护程序---启动一个RMI服务器程序来监视各个HotSpot java虚拟机的创建和中止。并且提供了一个访问远程监视工具接入的接口。  
  
jstatd

是一个Java远程方法调用(RMI)服务器应用程序-它监控测量Java

HotSpot虚拟机的创建和终止并且提供一个接口来允许远程监控工具依附到运行于本地主机的JVM（请参考《jstatd-Virtual

Machine jstat Daemon》）。  
  
　　在应用系统中的使用  
  
　　jps及其它jvmstat实用程序都使用

极为轻量级的观察机制。由JVM分配一小部分共享内存，而性能计数器也是从这部分内存中分配的。JVM子系统基于其感兴趣的事件更新性能计数器。客户端工

具仅仅负责异步地从共享内存段中进行读取。因此，总的来说，使用jvmstat进行监控的效果是很小的。  
  
用法:详细见<http://java.sun.com/j2se/1.5.0/docs/tooldocs/share/jstatd.html>  
  
12-14、kinit.exe,klist.exe,ktab.exe  
======================  
功能:Java安全管理工具  
  
用法:详见<http://java.sun.com/j2se/1.5.0/docs/tooldocs/>#security  
  
15、native2ascii.exe  
======================  
功能说明：  
  
将含有本地编码字符（既非 Latin1 又非 Unicode 字符）的文件转换为 Unicode 编码字符的文件。  
  
语法：  
  
native2ascii [options] [inputfile [outputfile]]  
  
补充说明：  
  
 Java

编译器和其它 Java 工具只能处理含有 Latin-1 和/或 Unicode 编码（udddd 记号）字符的文件。native2ascii

将含有其它字符编码的文件转换成含 Latin-1 和/或 Unicode 编码字符的文件。若省略

outputfile，则使用标准输出设备输出。此外，如果也省略 inputfile，则使用标准输入设备输入。  
  
命令选项  
  
-reverse 执行相反的操作：将含 Latin-1 和/或 Unicode 编码字符的文件转换成含本地编码字符的文件。  
  
-encoding[encoding\_name] 指定转换过程使用的编码名称。缺省的编码从系统属性 file.encoding 中得到。  
  
16、ordb.exe  
======================  
用法：orbd <选项>  
  
其中，<选项> 包括：  
  
-port          启动 ORBD 的激活端口，缺省值为 1049 (可选)  
  
-defaultdb       ORBD 文件的目录，缺省值为 "./orb.db" (可选)  
  
-serverid       ORBD 的服务器标识符，缺省值为 1 (可选)  
  
-ORBInitialPort     初始端口（必需）  
  
-ORBInitialHost     初始主机名称（必需）  
  
欢迎使用 Java IDL 服务器工具  
  
请在提示处输入命令  
  
17、idlj编译器用法：  
======================  
java com.sun.tools.corba.se.idl.toJavaPortable.Compile [选项] <idl 文件>  
  
 其中，<idl 文件> 是包含 IDL 定义的文件的名称，而[选项] 是以下所列选项的任一组合。这些选项是可选的并且可以以任意顺序显示；<idl 文件> 是必须的并且必须显示在最后。  
  
选项：  
  
-d <符号>                 等价于 IDL 文件中的以下行：  
  
                        #define <符号>  
  
-emitAll                发出所有类型，包括在 #included 文件中找到的那些类型。  
  
-f<端>                    定义要发出的绑定。   <端> 是 client、  
  
                         server、all、serverTIE 和 allTIE 中之一。   serverTIE  
  
和 allTIE  
  
                         将导致发出委托模型框架。如果未使用此  
  
                         标志，则假设为 -fclient。  
  
-i <包括路径>          默认情况下，搜索当前目录以获得其所包含的文件。  
  
                         此选项将添加其他目录。  
  
-keep                   如果要生成的文件已存在，则不  
  
                         覆盖它。默认情况下会覆盖该文件。  
  
-noWarn                抑制警告。  
  
-oldImplBase              生成与旧（1.4 版之前）JDK ORB 兼容的框架。  
  
-pkgPrefix <t> <前缀>     在文件范围内遇到类型或模块名称 <t> 时，  
  
                         将为针对 <t> 所生成的所有文件的 Java 软件包名称  
  
                         附加 <前缀>。  
  
-pkgTranslate <t> <pkg> 在遇到类型或模块名称 <t> 时，  
  
                         则在生成的 java 软件包中以 <pkg> 将其替换。请注意，  
  
                         首先进行 pkgPrefix 更改。   <t> 必须与完整的  
  
                         的软件包名称完全一致。另外，<t> 不能为  
  
                         org、org.omg 或 org.omg 的任何子包。  
  
-skeletonName <xxx%yyy> 根据模式命名框架。  
  
                         默认值为：  
  
                         适用于 POA 基类（-fserver 或 -fall）的 %POA  
  
                         适用于 oldImplBase 基类  
  
                         （-oldImplBase 和 [-fserver 或 -fall]）的 ％ImplBase。  
  
-td <目录>             将 <目录> 而非  
  
                         当前目录用作输出目录。  
  
-tieName <xxx%yyy>        根据模式命名 tie。默认值为：  
  
                         适用于 POA tie（-fserverTie 或 -fallTie）的 %POATie  
  
                         适用于 oldImplBase tie  
  
                         （-oldImplBase 和 [-fserverTie 或 -fallTie]）的 ％\_Tie。  
  
-v, -verbose              详细模式。  
  
-version                显示版本号并退出。  
  
  
  
18、servertool.exe  
======================  
可用命令：  
-------------------  
register - 注册一个可激活的服务器  
  
unregister - 取消服务器注册  
  
getserverid - 返回应用程序名称的服务器标识符  
  
list    - 列举所有已注册服务器  
  
listappnames - 列举当前定义的应用程序名称  
  
listactive - 列举当前活动的服务器  
  
locate     - 将已注册服务器定位在特定类型的端口  
  
locateperorb - 为已注册服务器的特定对象请求代理程序定位端口。  
  
orblist     - 对象请求代理程序 (orb) 名称及其映射列表  
  
shutdown - 关闭一个已注册服务器  
  
startup     - 启动一个已注册服务器  
  
help    - 取得帮助  
  
quit    - 退出此工具  
  
servertool > quit  
  
19、tnameserv.exe  
======================  
功能:提供命名服务的接口  
  
用法:详见<http://java.sun.com/j2se/1.5.0/docs/tooldocs/>#idl  
  
20、rmic  
======================  
功能说明：  
  
rmic 为远程对象生成 stub 和 skeleton。  
  
语法：  
  
rmic [ options ] package-qualified-class-name(s)  
  
补充说明：  
  
 rmic

编译器根据编译后的 Java 类（含有远程对象实现）名，为远程对象生成 stub 和 skeleton（远程对象是指实现

java.rmi.Remote 接口的对象）。在 rmic 命令中所给的类必须是经 javac 命令成功编译且是完全包限定的类。  
  
命令选项  
  
-classpath[路径] 指定 rmic 用于查询类的路径。如果设置了该选项，它将覆盖缺省值或 CLASSPATH 环境变量。目录用冒号分隔。  
  
-d[目录] 指定类层次的根目录。此选项可用来指定 stub 和 skeleton 文件的目标目录。  
  
-depend 使编译器考虑重新编译从其它类引用的类。 一般来说，它只重新编译从源代码引用的遗漏或过期的类。  
  
-g 允许生成调试表格。调试表格含有行号和局部变量的有关信息，即 Java 调试工具所使用的信息。缺省情况下，只生成行号。  
  
-J 与 -D 选项联用，它将紧跟其后的选项（ -J 与 -D 之间无空格）传给 java 解释器。  
  
-keepgenerated 为 stub 和 skeleton 文件保留所生成的 .java 源文件，并将这些源文件写到与 .class 文件相同的目录中，如果要指定目录，则使用 -d 选项。  
  
-nowarn 关闭警告。如果使用该选项，则编译器不输出任何警告信息。  
  
-show 显示 rmic 编译器的 GUI（图形用户界面）。输入一个或多个包限定类名（以空格分隔），并按回车键或"显示"按钮，创建 stub 和 skeleton。  
  
-vcompat （缺省值）创建与 JDK 1.1 和 1.2 stub 协议版本都兼容的 stub 和 skeleton。  
  
-verbose 使编译器和链接器输出关于正在编译哪些类和正在加载哪些类文件的信息。  
  
-v1.1 创建 JDK 1.1 stub 协议版本的 stub 和 skeleton。  
  
-v1.2 只创建 JDK 1.2 stub 协议版本的 stub。  
  
21、rmid  
======================  
功能说明：  
  
rmid 启动激活系统守护进程，以便能够在 Java 虚拟机上注册和激活对象。  
  
语法：  
  
rmid [-port port] [-log dir]  
  
补充说明：  
  
 rmid 工具启动激活系统守护进程。必须先启动激活系统守护进程，才能向激活系统注册可被激活的对象或在 Java 虚拟机上激活可被激活的对象。  
  
命令选项  
  
-C<某些命令行选项> 指定一个选项，在创建每个 rmid 的子守护进程（激活组）时，该选项以命令行参数的形式传给该子守护进程。  
  
-log[目录] 指定目录的名称，激活系统守护进程在该目录中写入其数据库及相关信息。缺省状态下，将在执行 rmid 命令的目录中创建一个 log 目录。  
  
-port[端口] 指定 rmid 的注册服务程序所使用的端口。激活系统守护进程将 ActivationSystem 与该注册服务程序中的名称java.rmi.activation.ActivationSystem 捆绑在一起。  
  
-stop 停止 -port 选项所指定端口上的当前 rmid 调用。若未指定端口，则将停止在端口 1098 上运行的 rmid。  
  
22、rmiregistry  
======================  
功能说明：rmiregistry 命令可在当前主机的指定端口上启动远程对象注册服务程序。  
  
语法：  
  
rmiregistry [port]  
  
补充说明：  
  
 rmiregistry

命令在当前主机的指定 port 上创建并启动远程对象注册服务程序。如果省略 port，则注册服务程序将在 1099

端口上启动。rmiregistry 命令不产生任何输出而且一般在后台运行。远程对象注册服务程序是自举命名服务。主机上的 RMI

服务器将利用它将远程对象绑定到名字上。客户机即可查询远程对象并进行远程方法调用。注册服务程序一般用于定位应用程序需调用其方法的第一个远程对象。该

对象反过来对各应用程序提供相应的支持，用于查找其它对象。java.rmi.registry.LocateRegistry

类的方法可用于在某台主机或主机和端口上获取注册服务程序操作。java.rmi.Naming 类的基于 URL

的方法将对注册服务程序进行操作，并可用于查询远程对象、将简单（字符串）名称绑定到远程对象、将新名称重新绑定到远程对象（覆盖旧绑定）、取消远程对象

的绑定以及列出绑定在注册服务程序上的 URL。  
  
23、pack200.exe  
======================  
功

能:Transforms a JAR file into a compressed pack200 file using the Java

gzip compressor. The compressed packed files are highly compressed JARs,

which can be directly deployed, saving bandwidth and reducing download

time.  
  
用法:详见<http://java.sun.com/j2se/1.5.0/docs/guide/deployment/deployment-guide/pack200.html>  
  
Usage:   pack200 [-opt... | --option=value]... x.pack[.gz] y.jar  
  
Packing Options  
  
   -g, --no-gzip                output a plain \*.pack file with no zipping  
  
   --gzip                          (default) post-process the pack output with gzip  
  
   -G, --strip-debug             remove debugging attributes while packing  
  
   -O, --no-keep-file-order        do not transmit file ordering information  
  
   --keep-file-order             (default) preserve input file ordering  
  
   -S{N}, --segment-limit={N}    output segment limit (default N=1Mb)  
  
   -E{N}, --effort={N}          packing effort (default N=5)  
  
   -H{h}, --deflate-hint={h}    transmit deflate hint: true, false, or keep (default)  
  
   -m{V}, --modification-time={V}   transmit modtimes: latest or keep (default)  
  
   -P{F}, --pass-file={F}       transmit the given input element(s) uncompressed  
  
   -U{a}, --unknown-attribute={a}   unknown attribute action: error, strip, or pass (default)  
  
   -C{N}={L}, --class-attribute={N}={L}   (user-defined attribute)  
  
   -F{N}={L}, --field-attribute={N}={L}   (user-defined attribute)  
  
   -M{N}={L}, --method-attribute={N}={L} (user-defined attribute)  
  
   -D{N}={L}, --code-attribute={N}={L} (user-defined attribute)  
  
   -f{F}, --config-file={F}        read file F for Pack200.Packer properties  
  
   -v, --verbose                increase program verbosity  
  
   -q, --quiet                   set verbosity to lowest level  
  
   -l{F}, --log-file={F}           output to the given log file, or '-' for System.out  
  
   -?, -h, --help                print this message  
  
   -V, --version                print program version  
  
   -J{X}                         pass option X to underlying Java VM  
  
Notes:  
  
   The -P, -C, -F, -M, and -D options accumulate.  
  
   Example attribute definition:   -C SourceFile=RUH .  
  
   Config. file properties are defined by the Pack200 API.  
  
   For meaning of -S, -E, -H-, -m, -U values, see Pack200 API.  
  
   Layout definitions (like RUH) are defined by JSR 200.  
  
Repacking mode updates the JAR file with a pack/unpack cycle:  
  
pack200 [-r|--repack] [-opt | --option=value]... [repackedy.jar] y.jar  
  
24、unpack200.exe  
======================  
功能:Transforms a packed file produced by pack200 into a JAR file  
  
用法:  
  
Usage:   unpack200 [-opt... | --option=value]... x.pack[.gz] y.jar  
  
Unpacking Options  
  
   -H{h}, --deflate-hint={h}     override transmitted deflate hint: true, false,or keep (default)  
  
   -r, --remove-pack-file        remove input file after unpacking  
  
   -v, --verbose                 increase program verbosity  
  
   -q, --quiet                set verbosity to lowest level  
  
   -l{F}, --log-file={F}       output to the given log file, or '-' for standard output (default)  
  
   -?, -h, --help             print this message  
  
   -V, --version                 print program version  
  
   -J{X}                      Java VM argument (ignored)  
  
25、packager.exe  
======================  
功能:对象包装程序.  
  
用法:具有图形界面与操作菜单.  
  
26、policytool.exe  
======================  
功能:生成规则文件的工具  
  
用法:有图形界面与操作菜单.

27、serialver

======================

功能说明：serialver 命令返回 serialVersionUID。

语法：serialver [ 命令选项 ]补充说明：serialver 以适于复制到演变类的形式返回一个或多个类的 serialVersionUID。不带参数调用时，它输出用法行。

命令选项

-show 显示一个简单的用户界面。输入完整的类名并按回车键或"显示"按钮可显示 serialVersionUID。

28、jarsigner

======================

功能说明：为 Java 归档 (JAR) 文件产生签名，并校验已签名的 JAR 文件的签名。

语法：jarsigner [ 命令选项 ] jar-file aliasjarsigner -verify [ 命令选项 ] jar-file

补充说明：jarsigner 工具用于两个目的：

1:为 Java 归档 (JAR) 文件签名

2:校验已签名的 JAR

文件的签名和完整性命令选项-keystore[url] 指定密钥仓库的 URL。缺省值是用户的宿主目录中的 .keystore

文件，它由系统属性"user.home"决定。-storetype[storetype]

指定要被实例化的密钥仓库类型。默认的密钥仓库类型是安全属性文件中 "keystore.type" 属性值所指定的那个类型，由

java.security.KeyStore 中的静态方法 getDefaultType 返回。-storepass[password]

指定访问密钥仓库所需的口令。这仅在签名（不是校验）JAR 文件时需要。在这种情况下，如果命令行中没有提供 -storepass

选项，用户将被提示输入口令。-keypass[password] 指定用于保护密钥仓库项（由命令行中指定的别名标出）的私钥的口令。使用

jarsigner 为 JAR 文件签名时需要该口令。如果命令行中没有提供口令，且所需的口令与密钥仓库的口令不同，则将提示用户输入它。

-sigfile[file] 指定用于生成 .SF 和 .DSA 文件的基本文件名。

-signedjar[file] 指定用于已签名的 JAR 文件的名称。

-verify 如果它出现在命令行中，则指定的 JAR 文件将被校验，而不是签名。如果校验成功，将显示"jar verified"。如果试图校验未签名的 JAR 文件，或校验被不支持的算法（例如未安装 RSA 提供者时使用的 RSA）签名的 JAR 文件，则将有如下显示： "jar is unsigned. (signatures missing or not parsable)" 。

-certs 如果它与 -verify 和 -verbose 选项一起出现在命令行中，则输出将包括 JAR 文件的每个签名人的证书信息。-verbose 如果它出现在命令行中，则代表"verbose"模式，它使 jarsigner 在 JAR 签名或校验过程中输出额外信息。

-internalsf 过去，JAR 文件被签名时产生的 .DSA（签名块）文件包含一个同时产生的 .SF 文件（签名文件）的完整编码副本。这种做法已被更改。为了减小输出 JAR 文件的整个大小，缺省情况下 .DSA 文件不再包含 .SF 文件的副本。但是如果 -internalsf 出现在命令行中，将采用旧的做法。该选项主要在测试时有用；实际上不应使用它，因为这样将消除有用的优化。

-sectionsonly 如果它出现在命令行中，则 JAR 文件被签名时生成的 .SF 文件（签名文件）将不包括含有整个清单文件的散列的头。它仅包含 与 JAR 中每个单独的源文件相关的信息和散列。该选项主要在测试时有用；实际上不应使用它，因为这样将消除有用的优化。-J[javaoption] 将指定的 javaoption 串直接传递到 Java 解释器。(（jarsigner 实际上是解释器的一个 "wrapper"）。该选项不应含有任何空格。它有助于调整执行环境或内存使用。要获得可用的解释器选项的清单，可在命令行键入 java -h 或 java -X。

29、keytool.exe

======================

功能说明：管理由私钥和认证相关公钥的 X.509 证书链组成的密钥仓库（数据库）。还管理来自可信任实体的证书。

语法：keytool [ 命令 ]

补充说明：

keytool 是个密钥和证书管理工具。它使用户能够管理自己的公钥/私钥对及相关证书，用于（通过数字签名）自我认证（用户向别的用户/服务认证自己）或数据完整性以及认证服务。它还允许用户储存他们的通信对等者的公钥（以证书形式）。

30、appletviewer

======================

功能说明：Java applet 浏览器。 appletviewer 命令可在脱离万维网浏览器环境的情况下运行 applet。

语法：

appletviewer [ threads flag ] [ 命令选项 ] urls ...

补充说明：

appletviewer 命令连接到 url 所指向的文档或资源上，并在其自身的窗口中显示文档引用的每个 applet。

注意：

如果 url 所指向的文档不引用任何带有

OBJECT、EMBED 或 APPLET 标记的 applet，那么 appletviewer 就不做任何事情。命令选项-debug 在

Java 调试器 jdb 中启动 appletviewer，使您可以调试文档中的 applet。

-encoding[编码名称] 指定输入 HTML 文件的编码名称。

-J[javaoption] 将 javaoption 字符串作为单个参数传给运行 appletviewer 的 Java 解释器。参数不能含有空格。由多重参数组成的字符串，其中的每个参数都必须以前缀 -J 开头，该前缀以后将被除去。这在调整编译器的执行环境或内存使用时将很有用。

31、extcheck

======================

功能说明：extcheck 检测目标 jar 文件与当前安装方式扩展 jar 文件间的版本冲突。

语法：extcheck [ -verbose ] targetfile.jar

补充说明：extcheck 实用程序检查指定 Jar 文件的标题和版本与 JDK TM 软件中所安装的扩展是否有冲突。在安装某个扩展前，可以用该实用程序查看是否已安装了该扩展的相同版本或更高的版本。extcheck 实用程序将 targetfile.jar 文件清单的 specification-title 和 specification-version 头与当前安装在扩展目录下所有 Jar 文件的相对应的头进行比较（缺省扩展目录为 jre/lib/ext）。extcheck 实用程序比较版本号的方式与 java.lang.Package.isCompatibleWith 方法相同。若未检测到冲突，则返回代码为 0。如果扩展目录中任何一个 jar 文件的清单有相同的 specification-title 和相同的或更新的 specification-version 号，则返回非零错误代码。如果 targetfile.jar 的清单中没有 specification-title 或 specification-version 属性，则同样返回非零错误代码。命令选项-verbose 对扩展目录中的 Jar 文件进行检查时，列出文件。此外，还报告目标 jar 文件的清单属性及所有冲突的 jar 文件。

32、jar.exe

======================

功能说明：Java归档工具

语法：jar [ 命令选项 ] [manifest] destination input-file [input-files]

补充说明：jar工具是个java应用程序，可将多个文件合并为单个JAR归档文件。jar是个多用途的存档及压缩工具，它基于ZIP和ZLIB压 缩格式。然而，设计jar的主要目的是便于将java applet或应用程序打包成单个归档文件。将applet或应用程序的组件(.class 文件、图像和声音)合并成单个归档文件时，可以用java代理(如浏览器)在一次HTTP事务处理过程中对它们进行下载，而不是对每个组件都要求一个新连 接。这大大缩短了下载时间。jar还能压缩文件，从而进一步提高了下载速度。此外，它允许applet的作者对文件中的各个项进行签名，因而可认证其来 源。jar工具的语法基本上与tar命令的语法相同。

命令选项

-c 在标准输出上创建新归档或空归档。

-t 在标准输出上列出内容表。

-x[file] 从标准输入提取所有文件，或只提取指定的文件。如果省略了file，则提取所有文件；否则只提取指定文件。

-f 第二个参数指定要处理的jar文件。在-c(创建)情形中，第二个参数指的是要创建的jar文件的名称(不是在标准输出上)。在-t(表(或-x(抽取)这两种情形中，第二个参数指定要列出或抽取的jar文件。

-v 在标准错误输出设备上生成长格式的输出结果。

-m 包括指定的现有清单文件中的清单信息。

用法举例："jar cmf myManifestFile myJarFile \*.class"-0 只储存，不进行 ZIP 压缩。-M 不创建项目的清单文件。-u 通过添加文件或更改清单来更新现有的 JAR 文件。

例如："jar -uf foo.jar foo.class"将文件 foo.class 添加到现有的JAR文件foo.jar中，而"jar umf manifest foo.jar"则用manifest中的信息更新foo.jar的清单。-C 在执行 jar 命令期间更改目录。

例如："jar -uf foo.jar -C classes \*"将classes目录内的所有文件加到foo.jar中，但不添加类目录本身。

程序示例

1:将当前目录下所有CLASS文件打包成新的JAR文件：jar cf file.jar \*.class

2:显示一个JAR文件中的文件列表jar tf file.jar

3:将当前目录下的所有文件增加到一个已经存在的JAR文件中jar cvf file.jar \*

33、javadoc

======================

功能说明Java API文档生成器从Java源文件生成API文档HTML页。

语法：javadoc [ 命令选项 ] [ 包名 ] [ 源文件名 ] [ @files ]

其中[ 包名 ]为用空格分隔的一系列包的名字，包名不允许使用通配符，如（\*）。

[ 源文件名 ]为用空格分隔的一系列的源文件名，源文件名可包括路径和通配符，如（\*）。

[ @files ]是以任何次序包含包名和源文件的一个或多个文件。

补充说明

Javadoc解析Java源文件中的声明和文档注释，并产生相应的HTML页缺省），描述公有类、保护类、内部类、接口、构造函数、方法和域。在 实现时，Javadoc要求且依赖于java编译器完成其工作。Javadoc调用部分 javac编译声明部分，忽略成员实现。它建立类的内容丰富的内部表示，包括类层次和"使用"关系，然后从中生成HTML。Javadoc还从源代码的文 档注释中获得用户提供的文档。当Javadoc建立其内部文档结构时，它将加载所有引用的类。由于这一点，Javadoc必须能查找到所有引用的类，包括 引导类、扩展类和用户类。

命令选项

-overview i>path/filename 指定javadoc应该从path/filename所指定的"源"文件中获取概述文档，并将它放到概述页中（overview- summary.html）。其中path/filename 是相对于-sourcepath的相对路径名。

-public 只显示公有类及成员。-protected 只显示受保护的和公有的类及成员。这是缺省状态。

-package 只显示包、受保护的和公有的类及成员。

-private 显示所有类和成员。

-help 显示联机帮助，它将列出这些javadoc和doclet命令行选项。

-doclet class 指定启动用于生成文档的docle 的类文件。该doclet定义了输出的内容和格式。如果未使用-doclet选项，则javadoc使用标准doclet生成缺省HTML格式。该类必须 包含start(Root)法。该启动类的路径由 -docletpath选项定义。-docletpath classpathlist 指定doclet类文件的路径，该类文件用-doclet选项指定。如果doclet已位于搜索路径中，则没有必要使用该选项。-1.1 生成具有用Javadoc 1.1生成的文档的外观和功能的文档。也就是说，页的背景为灰色，用图像做页眉，使用bullet列表而不是表格，具有单层目的目录结构，不包含继承 API，不使?\*\*\* TML框架，并且不支持内部类。该选项还自动将索引分割成每个字母一个文件。如果想要这种外观，则该选项比javadoc 1.1优越之处等于修正了一些错误。

-sourcepath sourcepathlist当将包名传递到javadoc命令中时，指定定位源文件（.java）的搜索路径。注意只有当用 javadoc命令指定包名时才能使用sourcepath选项 -- 它将不会查找传递到javadoc命令中的.java文件。如果省略-sourcepath，则javadoc使用类路径查找源文件。- classpath classpathlist 指定javadoc将在其中查找引用类的路径 -- 引用类是指带文档的类加上它们引用的任何类。Javadoc将搜索指定路径的所有子目录。classpathlist可以包括多个路径，彼此用逗号分隔。

-bootclasspath classpathlist 指定自举类所在路径。它们名义上是Java平台类。这个bootclasspath是Javadoc将用来查找源文件和类文件的搜索路径的一部分。在 classpathlist中用冒号（:）分隔目录。-extdirs dirlist 指定扩展类所在的目录。它们是任何使用Java扩展机制的类。这个 extdirs是Javadoc将用来查找源文件和在文件的搜索路径的一部分。在dirlist中用冒号（:）分隔目录。-verbose 在javadoc运行时提供更详细的信息。不使用verbose选项时，将显示加载源文件、生成文档（每个源文件一条信息）和排序的信息。verbose 选项导致打印额外的信息，指定解析每个java源文件的毫秒数。-locale language\_country\_variant 指定javadoc在生成文档时使用的环境。

-encoding name 指定源文件编码名，例如EUCJIS/SJIS。如果未指定该选项，则使用平台缺省转换器。-J[flag] 将flag直接传递给运行javadoc的运行时系统java。注意在J和flag之间不能有空格。标准 Doclet 提供的选项-d directory 指定javadoc保存生成的HTML件的目的目录。省略该选项将导致把文件保存到当前目录中。其中directory可以是绝对路径或相对当前工作目录 的相对路径。-use 对每个带文档类和包包括一个"用法"页。该页描述使用给定类或包的任何 API 的包、类、方法、构造函数和域。对于给定类 C，使用类 C 的任何东西将包括 C 的子类、声明为 C 的域、返回 C 的方法以及具有 C 类型参数的方法和构造函数。-version 在生成文档中包括 @version 文本。缺省地将省略该文本。-author 在生成文档中包括 @author 文本。-splitindex 将索引文件按字母分割成多个文件，每个字母一个文件，再加上一个包含所有以非字母字符开头的索引项的文件。

-windowtitle[title] 指定放入 HTML <title> 标记中的标题。它将出现在窗口标题栏中和为该页创建的任何浏览器书签（最喜爱的位置）中。该标题不应该包含任何 HTML 标记，因为浏览器将不能正确解释它们。在 title 中的任何内部引号必须转义。如果省略 -windowtitle，则 Javadoc 对该选项使用 -doctitle 的值。-doctitle[title] 指定放置在靠近概述概览文件顶部的标题。该标题将作为一级标题，居中地直接放在导航栏下面。title 可包含 html 标记和空格，但是如果这样，则必须用引号将它括起。在 title 中的任何内部引号必须转义。-title[title] 该选项不再存在。它仅存在于 Javadoc 1.2 的 Beta 版中。它已重命名为 -doctitle。重命名该选项是为了更清楚地表示它定义文档标题而不是窗口标题。-header[header] 指定放置在每个输出文件顶部的页眉文本。该页眉将放在上部导航栏的右边。header 可包含 HTML 标记和空格，但是如果这 样则必须用引号将它括起。在 header 中的任何内部引号必须转义。

-footer[footer] 指定放置在每个输出文件底部的脚注文本。脚本将放置在下部导航栏的右边。footer 可包含 html 标记和空格，但是如果这样，则必须用引号将它括起。在 footer 中的任何内部引号必须转义。-bottom[text] 指定放置在每个输出文件底部的文本。该文本将放置在页底，位于下部导航栏的下面。其中 text 可包含 HTML 标记和空格，但是如果这样，则必须用引号将它括起。在 text 中的任何内部引号必须转义。

-link[docURL] 创建链接指向已用 javadoc-生成的外部引用类的文档。参数 docURL是想要链接到的 javadoc-生成的外部文档的 URL。该位置可以是相对的或绝对的 URL。

-linkoffline[docURL][packagelistURL] 该选项为外部引用类名字创建指向文档的链接。- group[groupheading]packagepattern:packagepattern:... 将概述页上的包分成指定的组，每组一个表格。用不同的 -group 选项指定每个组。各组按命令行中指定的次序出现在页面上。组内的包按字母排序。对于给定 -group 选项，与 packagepattern 表达式列表匹配的包出现在标题为 groupheading 的表格中。-nodeprecated 防止在文档中生成任何不鼓励使用的 API。它执行-nodeprecatedlist 所做的事情，并且它不在文档其余部分生成任何不鼓励使用的 API。当编写代码并不想被不鼓励使用的代码分心时，这是非常有用的。

-nodeprecatedlist 防止在生成文件中包含不鼓励使用的 API 列表（deprecated-list.html）并防止在导航栏中包含该页的链接。（但是，javadoc 继续在文档其余部分生成不鼓励使用的 API。） 如果源代码未包含不鼓励使用的 API，并且想要导航栏更干净，则它是非常有用的。

-notree 在生成文档中忽略类/接口层次。缺省地，将产生该层次。-noindex 在生成文档中忽略索引。缺省地，将产生索引。-nohelp 在输出的每页顶部和底部的导航栏中忽略"帮助"链接。

-nonavbar 防止产生导航栏、页眉和脚注，否则它们将出现在生成页的顶部和底部。它对"bottom"选项没有影响。当只对内容感兴趣并且没有必要导航时，例如仅将文 件转换成 PostScript 或 PDF 以进行打印，-nonavbar 选项是非常有用的。

-helpfile[path/filename] 指定顶部和底部导航栏中"帮助"链接所链接到的替代帮助文件 path/filename 的路径。不使用该选项时，Javadoc 自动创建帮助文件 help-doc.html，它在 Javadoc 中硬编码。该选项使得可覆盖这种缺省情况。其中 filename 可以是任何名字，不局限于 help-doc.html -- Javadoc 将相应调整导航栏中的链接。

-stylesheetfile[path/filename] 指定替代 HTML 样式表单文件的路径。不使用该选项时，Javadoc 将自动创建样式表单文件 stylesheet.css，它在 Javadoc 中硬编码。该选项使得可覆盖这种缺省情况。其中 filename 可以是任何名字，不局限于 stylesheet.css。

-docencoding[name] 指定输出 HTML 文件的编码方式。总结： 在新发布的JDK1.5/J2SE 5带了一组新的试验工具。 性能监测工具jconsole ,jps,jstat,jstatd。同时一些新的故障发现和修改工具也可以下载了。这些工具能够显示内存表，在线程中生成堆栈跟踪，以及给一个进程或者 核心打印配置信息。

监测，管理工具

jconsole：J2SE监测和管理控制台---一个同JMX兼容的监测Java虚拟机的图形工具。其能够监视当地或者远程的Java虚拟机。 jps:java虚拟机进程状态工具---在目标系统上列出装备有HotSpot Java的虚拟机 jstat:java虚拟机统计监视工具---附加到一个装备了HotSpot的java虚拟机上来采集并且记录性能统计情况 jstatd：java虚拟机的jstat守护程序---启动一个RMI服务器程序来监视各个HotSpot java虚拟机的创建和中止。并且提供了一个访问远程监视工具接入的接口。

故障工具

jinfo：java的配置信息。打印出一个给定的进程或者核心文件或者远程debug服务器的配置信息。

jsadebugd：Java实用Agent调试守护进程----连接一个进程或者一个核心文件，使得其运行像一个调试服务器。

jmap：Java内存映射----打印共享对象内存映射或者一个给定的进程或者核心对象或者远程调试服务器上堆内存的细节

jstack：Java堆栈跟踪---打印出给定的进程或者核心文件或者远程调试服务器的线程的堆栈跟踪信息 另外在JDK里面，很多常用的标准的工具： 如基本工具(javac,java,javadoc,apt,appletviewer,jar,jdb,javah,javap,extcheck)     安全工具(keytool,jarsigner,policytool,kinit,klist,ktab)

国际化工具(native2ascii)

远程方法调用(RMI)工具(rmic,rmiregistry,rmid,serialver)

Java IDL和RMI-IIOP工具（tnameserv,idlj,orbd,servertool）

Java部署工具(pack200,unpack200)

Java插件工具(htmlconverter)

Java Web Start tools(javaws)

Java SE

6支持postmortem可观察性工具-它能够从挂起的Java进程或Java核心复制中获得信息。这些工具（除了jhat外）都使用Solaris

libproc库来依附到和读取被观察的程序。在观察期间，目标程序被挂起。当Java进程被挂起或当从一个Java进程中发生一个核心复制时，可以使用

这些工具。在任何可能的情况下，请考虑使用gcore来捕获系统的核心复制的一个快照并且使用任何下列工具"离线"分析核心复制。

(一) jinfo

jinfo打印一个给定的Java进程或核心文件或一个远程调试服务器的Java配置信息。配置信息包括Java系统属性和JVM命令行标志(更多信息，请参考《jinfo-Configuration Info》)。

(二) jmap

jmap：如果这个工具不使用任何选项（除了pid或core选项）运行，那么它显示类似于Solaris的pmap工具所输出的信息。这个工具支持针对Java堆可观察性的若干其它选项。

在Java SE 6平台中,新加入了一个-dump选项。这样可以使jmap能够把Java堆信息复制到一个文件中，然后我们可以使用新的jhat命令（见下面一节）来分析它。

jmap -dump选项并不使用Solaris libproc来实现实时处理；而是，它运行当前正运行的JVM中的一小段代码，由此来实现堆复制。既然这种堆复制代码运行于JVM内部，那么其速度是比 较快的。堆复制的效果大致相当于实现一次"完全的GC"（对整个堆的垃圾收集），再加上把该堆的内容写入到文件中。实现堆复制的另外一种可能的思路是使用 gcore来进行核心复制并且运行"jmap -dump"（这与以"离线"方式运行的核心复制形成对照）。

(三) jstack

jstack等价于Solaris的pstack工具。jstack打印所有的Java线程的堆栈跟踪信息（可选地包括本机帧信息），请参考《jstack-堆栈跟踪》。关于锁和死锁的信息也可以被打印，请参考java.util.concurrent locks。

(四) jsadebugd

jsadebugd依附到一个Java进程或核心文件并且担当一个调试服务器的作用。远程客户，例如jstack、jmap和jinfo，都能够通过Java RMI依附到该服务器。

(五) jhat

jhat是一个Java堆复制浏览器。这个工具分析Java堆复制文件（例如，由上面的 "jmap -dump"所产生的）。Jhat启动一个允许堆中的对象在web浏览器中进行分析的web服务器。这个工具并不是想用于应用系统中而是用于"离线"分 析。"jhat工具是平HIDDEN立的"，其意思是，它可以被用来观察在任何平台上所产生的堆复制。例如，我们有可能在Linux系统上使用jhat来 观察一个在Solaris OS上所产生的堆复制。