

JVM笔记无特殊说明，基于jdk8&14

[零、参考资料 2](#_Toc41656663)

[一、常用JVM参数 3](#_Toc41656664)

[1. GC参数 3](#_Toc41656665)

[2. 堆参数 4](#_Toc41656666)

[3. 非堆参数 4](#_Toc41656667)

[4. 其他参数 5](#_Toc41656668)

[二、JDK分析工具 5](#_Toc41656669)

[1jps 5](#_Toc41656670)

[2jstat 6](#_Toc41656671)

[3jinfo 6](#_Toc41656672)

[4jmap 7](#_Toc41656673)

[5jhat 7](#_Toc41656674)

[6jstack 7](#_Toc41656675)

[7jstatd 8](#_Toc41656676)

[8、jcmd 8](#_Toc41656677)

[三、JDK可视化工具 9](#_Toc41656678)

[1、OQL(对象查询语言) 9](#_Toc41656679)

[1.1简介 9](#_Toc41656680)

[1.2、SELECT 9](#_Toc41656681)

[1.3内置对象 9](#_Toc41656682)

[1.4对象函数 10](#_Toc41656683)

[1.5统计集合 11](#_Toc41656684)

[2、Visual VM 12](#_Toc41656685)

[2.1启动： 12](#_Toc41656686)

[2.2dump当前JVM 13](#_Toc41656687)

[2.3插件 15](#_Toc41656688)

[2.4BTrace 16](#_Toc41656689)

[四、JVM常用执行指令 17](#_Toc41656690)

[1、java 17](#_Toc41656691)

[1.1、说明 17](#_Toc41656692)

[1.2、文档 17](#_Toc41656693)

[1.3基础指令 17](#_Toc41656694)

[2、javac 17](#_Toc41656695)

[3、jar 18](#_Toc41656696)

[4、javap 18](#_Toc41656697)

[五、LINUX命令(centos7) 19](#_Toc41656698)

[1、top 19](#_Toc41656699)

[2、ps 20](#_Toc41656700)

[3、vmstat 22](#_Toc41656701)

[4、iostat 23](#_Toc41656702)

[5、uptime 24](#_Toc41656703)

[6、dmesg 24](#_Toc41656704)

[7、mpstat 25](#_Toc41656705)

[8、sar 26](#_Toc41656706)

[9、free 26](#_Toc41656707)

[10、tcpdump 27](#_Toc41656708)

[11、netstat 28](#_Toc41656709)

[12、pidstat 30](#_Toc41656710)

[13、grep 30](#_Toc41656711)

[14、cat 32](#_Toc41656712)

[15、tail 33](#_Toc41656713)

[16、curl 33](#_Toc41656714)

[17、ping 38](#_Toc41656715)

[18、vim 39](#_Toc41656716)

[附录 41](#_Toc41656717)

# 零、参考资料

Java虚拟机规范(Java SE 8版)

Java语言规范基于Java SE 8

深入理解Java虚拟机——JVM高级特性与最佳实践

实战Java虚拟机——JVM故障诊断与性能优化

垃圾回收的算法与实现

深入理解计算机系统

Stackoverflow:

https://stackoverflow.com/questions/16549066/java-major-and-minor-garbage-collections

OpenJdk：

https://openjdk.java.net/jeps/122

Oracle官方DOC：

https://docs.oracle.com/javase/8/javase-books.htm

https://docs.oracle.com/en/java/javase/14/

https://docs.oracle.com/javase/specs/jvms/se8/jvms8.pdf

https://www.oracle.com/webfolder/technetwork/tutorials/mooc/JVM\_Troubleshooting/week1/lesson1.pdf

https://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html

http://cr.openjdk.java.net/~sundar/8022483/webrev.01/raw\_files/new/src/share/classes/com/sun/tools/hat/resources/oqlhelp.html

补充说明：

1 因为我的私服是centos7，所以Linux命令是基于centos7的linux内核

# 一、常用JVM参数

## GC参数

* 1. 打印GC基本日志

JDK8: -XX:PrintGC JDK9+: -Xlog:gc

* 1. 打印GC详细日志

JDK8: -XX:PrintGCDetails JDK9+: -Xlog:gc\*

-XX:PrintGCDetails 还会使JVM在退出前打印堆详细信息

* 1. 打印GC时堆的全面信息

JDK8: -XX:PrintHeapAtGC JDK9+:移除

* 1. 打印GC发生时间

JDK8: -XX:PrintGCTimeStamps JDK9+: -Xlog:gc\*已包含

* 1. 打印程序执行时间

-XX:+PrintGCApplicationConcurrentTime

* 1. 打印程序因STW而停顿的时间

-XX:+PrintGCApplicationStoppedTime

* 1. 打印软引用、弱引用、虚引用和Finallize队列信息

JDK8: +PrintReferenceGC JDK9+移除

* 1. 输出GC信息至文本

JDK8:-Xloggc:log/${filePath} JDK9+:-Xlog:gc:log/${filePath}

* 1. 打印类加载日志

JDK8: -XX:+TraceClassLoading JDK9+: -Xlog:class+load=info

* 1. 打印类卸载日志

JDK8: -XX:+TraceClassUnLoading JDK9+: -Xlog:class+unload=info

* 1. 显示当前的类信息柱状图

-XX:+PrintClassHistogram

* 1. 查看JVM运行显示参数

-XX:+PrintVMOptions

* 1. 打印JVM显示和隐式(默认)参数

-XX:+PrintCommandLineFlags

* 1. 打印所有系统参数

-XX:+PrintFlagsFinal

* 1. 在新生代和老年代使用串行回收器

-XX:UseSerialGC

* 1. 在新生代使用并行回收器

-XX:+UseParNewGC

* 1. 在老年代使用并行回收器

-XX:+UseParallelOldGC

* 1. 设置用于垃圾回收的线程数

-XX:ParallelGCThreads

* 1. 设置最大GC停顿时间

-XX:MaxGCPauseMillis

* 1. 设置吞吐量大小

-XX:GCTimeRatio

* 1. 打开开自适应GC策略

-XX:+UseAdaptiveSizePolicy

* 1. 使用G1

-XX:+UseG1GC

* 1. 设置停顿间隔

-XX:GCPauseIntervalMillis

* 1. 使用CMS

-XX:+UseConcMarkSweepGC

* 1. 设置CMS线程数

-XX:ParallelCMSThreads

* 1. 设置CMS触发时，老年代比例

-XX:CMSInitiatingOccupancyFraction

* 1. 设置CMS在FullGC后是否进行碎片整理

-XX:+UseCMSCompactAtFullCollection

* 1. 设置CMS进行内存压缩前，FullGC次数

-XX:CMSFullGCsBeforeCompaction

* 1. 允许对类元数据回收

-XX:+CMSClassUnloadingEnabled

* 1. 设置CMS触发时，永久区比例,仅在激活CMSClassUnloadingEnabled时有效

-XX:CMSInitiatingPermOccupancyFraction

* 1. 只有达到阈值才执行CMS GC

-XX:UseCMSInitiatingOccupancyOnly

## 堆参数

* 1. 初始堆空间

-Xms

* 1. 最大堆空间

-Xmx

* 1. 设置新生代大小

-Xmn

* 1. 设置新生代中eden区和from/to区的比例

-XX:SurvivorRatio

SurvivorRatio=eden/from=eden/to

* 1. 设置新生代与老年代的比例

-XX:NewRatio

NewRatio=老年代/新生代

* 1. 设置在内存溢出时到处整个堆信息

-XX:+HeapDumpOnOutOfMemoryError

* 1. 设置导出堆文件存放路径

-XX:HeapDumpPath

与-XX:+HeapDumpOnOutOfMemoryError配合

* 1. 设置大对象直接进入老年代的阈值

-XX:MaxTenuringThreshold

## 非堆参数

* 1. 最大直接内存

-XX:MaxDirectMemorySize

如果不设置，则默认与-Xmx相同

* 1. 每个线程的栈大小

-Xss

* 1. 打开TLAB

-XX:+UseTLAB

* 1. 打印TLAB相关信息

-XX:+PrintTLAB

JDK9+不支持

* 1. TLAB大小设置

-XX:TLABSize

* 1. 设置TLAB大小自动调整

-XX:+ResizeTLAB

* 1. 扩容时触发FullGC的初始化阈值

-XX:MetaspaceSize

## 其他参数

* 1. 设置工作模式为客户端

-client

* 1. 设置工作模式为服务端

-server

* 1. 禁用System.gc()（显示GC）

-XX:+DisableExplicitGC

* 1. 设置并发处理显示GC

-XX:+ExplicitGCInvokesConcurrent

* 1. 后台编译

-XX:+BackgroundCompilation

* 1. 开启逃逸分析

-XX:+DoEscapeAnalysis

仅在server模式下有效

* 1. 打开标量替换

-XX:+EliminateAllocation

* 1. 开启解释执行模式

-Xint

* 1. 开启编译模式

-Xcomp

* 1. 开启混合模式民，

-Xmixed

5参数JVM

# 二、JDK分析工具

## 1jps

a说明

类似Linux的ps命令，只列出当前用户的有权限查看的java程序的进程

b文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/jps.html

c命令

jps [-q] [-mlvV] [hostid]

d参数说明

|  |  |
| --- | --- |
| -q | 只输出进程ID |
| mlvV | |
| -m | 列出传递给主函数的参数 |
| -l | 主函数完整路径 |
| -v | 列出传递给jvm的参数 |
| -V | 列出通过flags文件传递给jvm的参数 |

## 2jstat

a说明

查看java运行时信息

b文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/jstat.html

c命令

jstat [Options] ${pid} [ ${interval}[s|ms] [ ${count} ] ]

d参数

|  |  |
| --- | --- |
| -help | 帮助信息 |
| -class | 显示classLoader信息 |
| -compiler | 显示jit(即时编译器)相关信息 |
| -gc | 监视gc信息 |
| -gccapacity | 显示各个区域信息 |
| -gccause | 跟gcutil一样，显示垃圾回收信息，但额外显示上一次GC原因 |
| -gcnew | 新生代信息 |
| -gcnewcapacity | 新生代大小 |
| -gcold | 老年代信息和元数据区信息 |
| -gcoldcapacity | 老年代大小 |
| -gcmetacapacity | 显示元数据区大小 |
| -gcutil | 与-gc相同，显示垃圾回收信息，着重于已使用空间的百分比 |
| -printcompilation | 显示jit编译的方法信息 |
| -t | 显示程序运行时间 |
| -h | 指定周期性输出数据， |
| pid | 进程ID |
| interval | 周期时间 |
| count | 循环次数 |

## 3jinfo

a说明

查看java的参数，支持在运行时修改部分参数，这个命令在jdk8中处于实验性质

b文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/jinfo.html

c命令

jinfo [option] pid

d参数

|  |  |
| --- | --- |
| no-option | 输出全部的参数和系统属性 |
| -flag<name> | 打印指定参数 |
| -flag [+|-]<name> | 开启或关闭指定参数, Boolean |
| -flag <name>=<value> | 设置指定jvm参数值 |
| - sysprops | 输出系统属性 |

## 4jmap

ashuoming

可以生产Java程序的堆dump文件，也可以查看堆统计信息、classLoader信息、finalizer队列

b文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/jmap.html

c命令

jmap [options] pid

d参数

|  |  |
| --- | --- |
| no option |  |
| -histo[:live] | 打印每个class的实例数目,内存占用,类全名信息. VM的内部类名字开头会加上前缀”\*”. 如果live子参数加上后,只统计活的对象数量 |
| -dump:[live,] format=b, file=filename | 使用hprof二进制形式,输出jvm的heap内容到文件，live参数可选，如果指定，则只输出活对象 |
| -heap | 打印堆摘要、配置、gc算法，打印字符串的数量和大小。 |
| -clstats | 打印classLoader信息 |
| -F | 在没有pid时，强制使用-histo和-dump，同时不支持live |
| -h | 帮助信息 |
| -help | 帮助信息 |
| -Jflag | 传递参数给jmap启动的jvm |

## 5jhat

在jdk9+被移除，官方建议用visualVm代替

## 6jstack

a说明

导出线程栈信息，进行死锁检查

b文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/jstack.html

c命令

d参数

|  |  |
| --- | --- |
| -F | 强制打印栈信息 |
| -l | 长列表，打印锁的附加信息 |
| -m | 打印Java 和 native C/C++ frames的所有栈信息 |
| -h ｜-help | 打印帮助信息 |

## 7jstatd

a说明

启用远程监控，需要配置java的安全策略，并保存于jstatd.all.policy文件中

b文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/jstatd.html

c命令

jstatd J-Djava.security.policy=jstatd.all.policy [ options ]

d参数

|  |  |
| --- | --- |
| -nr | 找不到RMI注册表时，不尝试创建 |
| -p | 指定端口 |
| -n | RMI名称，默认JstatRemoteHost，如果本地有多个jstatd服务，需要保证唯一 |

## 8、jcmd

8.1说明

a将诊断命令请求发送到本地正在运行的JVM，用来导出堆、查看Java进程、导出线程信息、执行GC、还可以进行采样分析

b、执行者必须跟JVM是同一用户和用户组

8.2文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/jcmd.html

c命令

jcmd <pid | main-class> <command ... | PerfCounter.print | option>

d参数

|  |  |
| --- | --- |
| -l | 列出所有JVM |
| -h ｜-help | 列出JVM支持的命令 |
| -f filename | 从文件中读取命令 |
| PerfCounter.print | 打印目标Java进程上可用的性能计数器 |

# 三、JDK可视化工具

## 1、OQL(对象查询语言)

### 1.1简介

OQL是用于查询Java堆的类SQL查询语言。OQL允许过滤/选择从Java堆中获取的信息。虽然HAT已经支持预定义的查询，例如“显示类X的所有实例”，但OQL增加了更多的灵活性。OQL基于JavaScript表达式语言。

### 1.2、SELECT

select <JavaScript expression to select>

[ from [instanceof] <class name> <identifier>

[ where <JavaScript boolean expression to filter> ] ]

说明：

class name：java类的完全限定名，如：java.lang.String, java.util.ArrayList

instanceof：表示也查询某一个类的子类，如果不明确instanceof，则只精确查询class name指定的类

可以使用obj.field\_name语法访问Java字段，并且可以使用array [index]语法访问数组元素

例子：

1.查询长度大于等于100的字符串:

select s from java.lang.String s where s.value.length >= 100

2.查询长度大于等于256的int数组:

select a from [I a where a.length >= 256

3.查询长度大于等于256的int数组:

select a from int[] a where a.length >= 256

4.显示所有File对象的文件路径:

select file.path.value.toString() from java.io.File file

5.选取所有的ClassLoader，包括子类:

select classof(cl).name from instanceof java.lang.ClassLoader cl

6. 由给定id字符串标识的Class的实例

select o from instanceof 0x741012748 o(0x741012748是类的ID)

7.表示两位数整数的字符串:

select {instance: s, content: s.toString()} from java.lang.String s where /^\d{2}$/(s.toString())

### 

### 1.3内置对象

|  |  |  |
| --- | --- | --- |
| 对象 | 说明 | 例子 |
| heap.forEachClass(callback) | 对每一个Class对象执行一个回调操作。其中 callback 为 Javascript 函数 | heap.forEachClass(function(it) {  if (classes[it.name] != null) {  if (multipleLoadedClasses[it.name] != null) {  multipleLoadedClasses[it.name] = multipleLoadedClasses[it.name] + 1;  } else {  multipleLoadedClasses[it.name] = 1;  }  } else {  classes[it.name] = it;  }  }); |
| heap.forEachObject (callback, clazz, includeSubtypes) | clazz:指定实例对象，默认为Java.lang.Object  includeSubtypes：是否包含子类，默认true |  |
| heap.findClass(className) | 查找给定名称的Java类, 生成的Class对象具有以下属性:  name - name of the class.  superclass - Class object for super class (or null if java.lang.Object).  statics - name, value pairs for static fields of the Class.  fields - array of field objects. field object has name, signature properties.  loader - ClassLoader object that loaded this class.  signers - signers that signed this class.  protectionDomain - protection domain to which this class belongs  同时具有一下方法：  isSubclassOf - tests whether given class is direct or indirect subclass of this class or not.  isSuperclassOf - tests whether given Class is direct or indirect superclass of this class or not.  subclasses - returns array of direct and indirect subclasses.  superclasses - returns array of direct and indirect superclasses. | select heap.findClass("java.util.Vector")  select heap.findClass("java.util.Vector").superclasses() |
| head.findObject(objID) | 根据对象ID找对象 |  |
| heap.classes() | 返回堆快照中所有的类的集合 |  |
| heap.objects(clazz, [includeSubtypes], [filter]) | 返回堆快照中所有的对象的集合  clazz:指定类名称，默认java.lang.Object  includeSubtypes：是否包含子类，true  filter:过滤规则 | select heap.objects ("java.io.File",true, it.value !=null) |
| head.livepaths(obj) | 查找对象活的引用链 | select heap.livepaths(s) from java.lang.String s  输出例子：  java.lang.String#1600-> geym.zbase.ch7.heap.WebPage#57-> java.lang.Object[]#341-> java.util.Vector#11-> geym.zbase.ch7.heap.Student#3 |
| heap.roots() | 获取堆的根对象 |  |
| heap.finalizables() | 等待垃圾收集的java对象的枚举 |  |

### 1.4对象函数

|  |  |  |
| --- | --- | --- |
| 函数 | 说明 | 例子 |
| classof(objname) | 返回指定对象的类  返回对象有以下属性：  name - name of the class.  superclass - Class object for super class (or null if java.lang.Object).  statics - name, value pairs for static fields of the Class.  fields - array of field objects. Field objects have name, signature properties.  loader - ClassLoader object that loaded this class.  signers - signers that signed this class.  protectionDomain - protection domain to which this class belongs.  有以下方法：  isSubclassOf() - tests whether given class is direct or indirect subclass of this class or not.  isSuperclassOf() - tests whether given Class is direct or indirect superclass of this class or not.  subclasses() - returns array of direct and indirect subclasses.  superclasses() - returns array of direct and indirect superclasses. | select classof(o).name from instanceof java.lang.ref.Reference o |
| allocTrace(objName) | This returns allocation site trace of a given Java object if available.  返回对象的属性有：  className - name of the Java class whose method is running in the frame.  methodName - name of the Java method running in the frame.  methodSignature - signature of the Java method running in the frame.  sourceFileName - name of source file of the Java class running in the frame.  lineNumber - source line number within the method. |  |
| objectid(objName) | 获取对象的ID | select objectid(o) from java.lang.Object o |
| forEachReferrer(callback,obj) | 遍历指定对象的引用者 |  |
| reachables(obj) | 获取指定对象的可达对象集合 | select reachables(p) from java.util.Properties p |
| referrers(obj) | 返回指定对象的引用者集合 | select count(referrers(o)) from java.lang.Object o |
| referees(obj) | 返回指定对象的直接引用者集合 |  |
| refers(firstobj,secondObj) | 判断firstobj是否引用了secondObj |  |
| root(obj) | 判断是否根对象，如果是则返回描述，否则返回null |  |
| sizeof(obj) | 对象大小，但不包含其引用对象 |  |
| rsizeof(obj) | 对象大小，包含其引用对象,即堆深。其不仅与对象有关，还有当前对象的数据内容有关 |  |
| toHtml(obj) | 将对象转为html显示 | select "<b>" + toHtml(o) + "</b>" from java.lang.Object o |
|  |  |  |
|  |  |  |

### 1.5统计集合

|  |  |  |
| --- | --- | --- |
| 函数 | 说明 | 例子 |
| contains(objSet, booleanExpression) | 判断当前集合是否包含指定表达式的对象，  内置对象有：  it -> currently visited element  index -> index of the current element  array -> array/enumeration that is being iterated | select count(heap.classes(), "/java.io./.test(it.name)") |
| count(objSet, booleanExpression) | 当前集合包含指定表达式的对象的总数，  内置对象有：  it -> currently visited element  index -> index of the current element  array -> array/enumeration that is being iterated |  |
| filter(objSet, booleanExpression) | 获得当前集合包含指定表达式的对象的子集，  内置对象有：  it -> currently visited element  index -> index of the current element  array -> array/enumeration that is being iterated |  |
| length(objSet) | 返回集合中元素数量 |  |
| concat(objSet1,ObjSet2) | 链接两个集合 |  |
| map(objSet,transerRule) | 将集合中元素按特定规则转换  内置对象：  it -> currently visited element  index -> index of the current element  array -> array/enumeration that is being iterated  result -> result array/enumeration | select map(heap.findClass("java.io.File").statics, "index + = + toHtml(it)") |
| max(setObj,[express]) | 获得集合中最大的元素，默认进行数值比较  内置对象：  lhs -> left side element for comparison  rhs -> right side element for comparison | select max(map(heap.objects(java.lang.String, false), it.value.length))  select max(heap.objects(java.lang.String), lhs.value.length > rhs.value.length) |
| min(setObj,[express]) | 最小值，其他同max |  |
| sort(setObj,[express]) | 排序，其他同max |  |
| top(set,expression,num) | 获取集合中，指定规则的头几个对象  内置对象：  lhs -> left side element for comparison  rhs -> right side element for comparison |  |
| sum(setObj,[express]) | 计算集合的累计值，默认进行数值计算 | select sum(map(reachables(p), sizeof(it))) from java.util.Properties p  select sum(reachables(p), sizeof(it)) from java.util.Properties p |
| unique(objSet) | 返回无重复的对象集合 |  |

## 2、Visual VM

### 2.1启动：

A screenshot of a social media post

Description automatically generated

### 2.2dump当前JVM

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

### 2.3插件

A screenshot of a computer

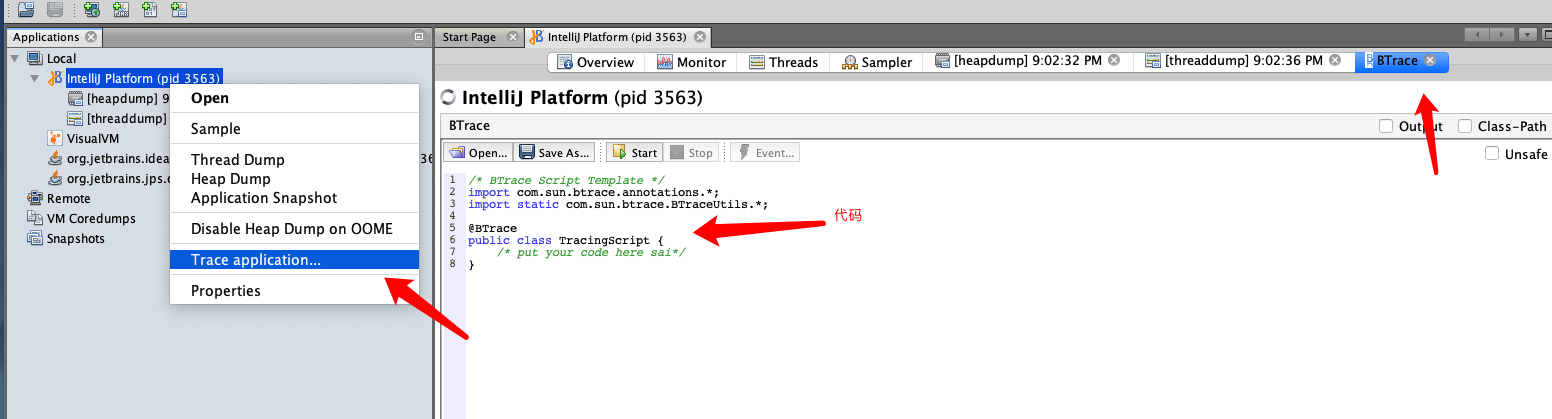
Description automatically generated

A screenshot of a social media post

Description automatically generated

### 2.4BTrace

通过字节码注入，动态监控系统的运行情况。它可以跟踪指定的方法调用、构造函数调用和系统内存等信息。在Visual VM中安装插件BTrace后，右击Java程序打开Trece application



例子脚本

@TLS

private static long startTime = 0;

@OnMethod(clazz="/.+/", //监控任意类

method="/slowMethod/") //监控slowMethod方法

public static void startMethod(){

startTime = timeMillis();

}

@OnMethod(clazz="/.+/",method="/slowMethod/",location=@Location(Kind.RETURN))//方法返回时触发

public static void endMethod(){

long time = timeMillis() - startTime;

println(strcat("execute time(nanos): ", str(time)));

}

## 3、JConsole

3.1、说明

3.2、文档

https://docs.oracle.com/en/java/javase/14/management/using-jconsole.html#GUID-77416B38-7F15-4E35-B3D1-34BFD88350B5

## 4、Mission Control

## 5、JHSDB

# 四、JVM常用执行指令

## 1、java

### 1.1、说明

### 1.2、文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/java.html

### 1.3基础指令

a、执行一个main类

java [options] mainclass [args ...]

b、执行一个带main方法的jar包

java [options] -jar jarfile [args ...]

c、执行一个模块里的main类

java [options] -m module[/mainclass] [args ...]

java [options] --module module[/mainclass] [args ...]

d、执行一个源文件，即非class二进制文件，如：hello.java

java [options] source-file [args ...]

ps:这个语句等同于，javac hello.java & java hello

### 1.4参数

ps：如无特殊说明，非windows指的是**Oracle Solaris, Linux, and macOS**平台

|  |  |
| --- | --- |
| classname | 运行的class文件 |
| args | main方法入参 |
| options | |
| -version[:release] |  |
| -jar | 执行一个jar包 |
| -m  --module | 执行一个模块，jdk9开始有模块概念 |
| **标准 Options，适用于所有虚拟机实现** | |
| -agentlib:libname[=options] | 加载本地代理库 |
| -agentpath:pathname[=options] | 加载本地代理库，机制跟agentlib相同，但需要库的绝对路径 |
| --class-path classpath  -classpath  -cp | 指定class库，用（;）分割多个  等同环境变量(CLASSPATH) |
| --disable-@files | 禁用某个文件 |
| --enable-preview | 是否可以用预览功能 |
| --module-path modulepath...  -p modulepath | 模块路径，用（;）分割多个 |
| --upgrade-module-path modulepath... | 替换指定模块 |
| --add-modules ${module} | 指定根模块，module可以是ALL-DEFAULT, ALL-SYSTEM, ALL-MODULE-PATH. |
| --list-modules | 仅列出可见模块并退出 |
| -d module\_name  --describe-module module\_name | 仅显示指定模块的描述并退出 |
| --dry-run | 仅启动VM但不运行main方法，用于测试option参数是否正确 |
| --validate-modules | 仅验证模块冲突和错误，并退出 |
| -Dproperty=value | 设置一个系统属性值，值如果有空格需要双引号包裹 |
| -disableassertions[:[packagename]...|:classname]  -da[:[packagename]...|:classname] | 禁用断言，默认范围为所有包和类  可以通过指定包和类。如果参数是:...，则针对无名包禁用断言 |
| -disablesystemassertions  -dsa | 禁用所有系统类的断言 |
| -enableassertions[:[packagename]...|:classname]  -ea[:[packagename]...|:classname] | 启用断言，默认范围为所有包和类  可以通过指定包和类。如果参数是:...，则针对无名包启用断言 |
| -enablesystemassertions  -esa | 启用所有系统类的断言 |
| -javaagent:jarpath[=options] | 指定代理包 |
| --show-version | 启动时打印版本号 |
| -showversion | 抛错时打印版本号 |
| --show-module-resolution | 启动期间显示模块间关系 |
| -splash:imagepath | 指定一个splash图片 |
| -verbose:class | 显示加载的类的信息 |
| -verbose:gc | 显示GC事件信息 |
| -verbose:jni | 显示调用native方法或接口行为的信息 |
| -verbose:module | 显示使用到的模块信息 |
| --version | 打印版本信息并退出 |
| -version | 打印版本信息并退出 |
| -X | 打印扩展option的帮助信息 |
| --help-extra | 打印扩展option的帮助信息 |
| @argfile | 从文件中读取option配置 |
| **扩展Option，仅适用于HotSpot** | |
| -Xbatch | 禁用后台编译  等同于-XX:-BackgroundCompilation |
| -Xbootclasspath/a:directories|zip|JAR-files | 指定引导类路径，多个时：  windows用;分割  非windows用:分割 |
| -Xcheck:jni | 在调用jni之前会检查调用参数和运行时环境数据，  也会检查挂起异常，  会降低性能 |
| -Xdebug | 什么都不做，向后兼容 |
| -Xdiag | 显示附加的诊断信息 |
| -Xint | 仅用解释模式执行  禁止编译native代码  解释执行所有字节  JIT编译的优势无法体现 |
| -Xinternalversion | 显示比-version更多的信息 |
| -Xlog:option | 启用统一日志组件  详见[链接](https://docs.oracle.com/en/java/javase/14/docs/specs/man/java.html#enable-logging-with-the-jvm-unified-logging-framework) |
| -Xmixed | 用解释模式执行字节码，不影响native code |
| -Xmn size | 设置年轻代初始和最大堆容量，  单位：k，m，g  可以分别用-XX:NewSize设置初始大小，-XX:MaxNewSize设置最大大小 |
| -Xms size | 设置最小&初始堆大小  值必须1024的整数倍，大于1MB  可以分别用-XX:MinHeapSize设置最小堆大小，-XX:InitialHeapSize设置堆初始大小  如果不设置此值，默认初始化堆大小为年轻代和年老代的和 |
| -Xmx size | 指定堆的最大值  值必须1024的整数倍，大于2MB  默认值决定于运行时系统配置  等同于-XX:MaxHeapSize. |
| -Xnoclassgc | 禁用gc |
| -Xrs | 减少JVM感知系统信号  solaris,linux,macos中，减少SIGINT, SIGTERM, SIGHUP, and SIGQUIT信号干扰，同时无法导出SIGQUIT 线程信息  windows中减少CTRL\_C\_EVENT, CTRL\_CLOSE\_EVENT, CTRL\_LOGOFF\_EVENT, or CTRL\_SHUTDOWN\_EVENT干扰，同时无法用ctrl+break导出线程 |
| -Xshare:mode | 设置class data sharing(CDS)  mode取值：  auto：默认  on，强制使用CDS，否则抛错；仅用于测试，不能用于生产环境  off：关闭CDS |
| -XshowSettings | 显示所有设置 |
| -XshowSettings:category | 显示指定类型设置  category取值：  all：默认值  locale：本地设置  properties：系统设置  vm：jvm设置  system：适用于linux，显示主机系统或容器配置 |
| -Xss size | 设置线程本地栈大小  windows默认值以来虚拟内存  非windows：1024KB  等同于-XX:ThreadStackSize |
| --add-reads module=target-module | 无条件让指定模块读取目标模块(Updates module to read the target-module, regardless of the module declaration) |
| --add-exports module/package=target-module | 无条件将模块导出包到目标模块的 |
| --add-opens module/package=target-module | 无条件将模块的包开放给目标模块 |
| --illegal-access=parameter | 非法访问数据：  permit：默认行为。允许对封装类型进行非法访问。当第一次尝试通过反射进行非法访问时会生成一个警告  warn：与permit一样，但每次非法访问尝试时都会产生错误  debug：同时显示非法访问尝试的堆栈跟踪。  deny：不允许非法的访问尝试。这将是未来的默认行为 |
| --limit-modules module[,module...] | 指定可见模块范围 |
| --patch-module module=file(;file)\* | 重写指定模块 |
| --source version | 设置数据源版本 |
| **进阶option** | |
| -XX:+UnlockDiagnosticVMOptions | 启用诊断类配置，默认是禁用 |
| -XX:+UnlockExperimentalVMOptions | 启用实验性质的功能 |
| **HotSpot进阶配置** | |
| -XX:ActiveProcessorCount=x | 修改jvm可用cpu核数，受-XX:-UseContainerSupport配置影响 |
| -XX:AllocateHeapAt=path | 指定对象堆分配位置，  代码、元数据、线程本地栈仍在内存中，不影响 |
| -XX:-CompactStrings | 禁用压缩字符串功能，启用压缩功能时，在内部仅用单字节表示并用ISO-8859-1 / Latin-1编码表示每个字符(When this option is enabled, Java Strings containing only single-byte characters are internally represented and stored as single-byte-per-character Strings using ISO-8859-1 / Latin-1 encoding)。  禁用压缩后，使用utf-16编码(jdk8是utf-8) |
| -XX:ErrorFile=filename | 指定错误数据输出文件  默认是当前目录，名为hs\_err\_pidpid.log文件 |
| -XX:+ExtensiveErrorReports | 输出更详细错误报告到ErrorFile，包括敏感信息，默认禁用 |
| -XX:FlightRecorderOptions=parameter=value  -XX:FlightRecorderOptions:parameter=value | 配置jfr参数，商业工具，具体参数值参考文档 |
| -XX:LargePageSizeInBytes=size | 设置大页容量，对内存密集型程序有利，必须是2的倍数 |
| -XX:MaxDirectMemorySize=size | 配置直接内存大小，默认或设置为0，JVM自动分配 |
| -XX:-MaxFDLimit | 禁用打开文件描述符限制改为硬件限制(Disables the attempt to set the soft limit for the number of open file descriptors to the hard limit)，只有在macos下才需要禁用。 |
| -XX:NativeMemoryTracking=mode | 指定跟踪本地内存模式  mode取值：  off，默认值，不跟踪本地内存使用  summary，只跟踪基本内存使用  detail，跟踪内存使用详情 |
| -XX:ObjectAlignmentInBytes=alignment | 对象大小对齐，取值8-256，2的倍数；默认8byte； |
| -XX:OnError=string | 在抛错时，执行指定命令串，多命令用分号分隔  比如非windows：  -XX:OnError="gcore %p;gdb -p %p" |
| -XX:OnOutOfMemoryError=string | 在OOM时执行指定命令 |
| -XX:+PrintCommandLineFlags | 打印JVM显示和隐式(默认)参数 |
| -XX:+PreserveFramePointer | 使用RBP寄存器存储frame pointer |
| -XX:+PrintNMTStatistics | 当设置-XX:NativeMemoryTracking，在JVM退出时收集打印本地内存跟踪数据 |
| -XX:SharedArchiveFile=path | 指定CDS文件路径 |
| -XX:SharedArchiveConfigFile=shared\_config\_file | 将分享数据加入到指定的配置文件中 |
| -XX:SharedClassListFile=file\_name | 批量加载存储在指定文件里的CDS列表 |
| -XX:+ShowCodeDetailsInExceptionMessages | 显示详细错误信息，for NullPointerException |
| -XX:+ShowMessageBoxOnError | 当虚拟机出错时，弹出一个提示窗，并阻塞退出操作，用于分析出错原因 |
| -XX:StartFlightRecording=parameter=value | 启用jfr，商业工具，具体参数值参考文档 |
| -XX:ThreadStackSize=size | 设置线程栈大小，同-Xss |
| -XX:-UseBiasedLocking | 禁用偏向锁 |
| -XX:-UseCompressedOops | 禁用压缩指针，默认启用压缩指针，仅在64bit的JVM有效 |
| -XX:-UseContainerSupport | 禁用容器支持，仅在64bit的JVM有效 |
| -XX:+UseHugeTLBFS | 仅用于LINUX，等同于  -XX:+UseLargePage |
| -XX:+UseLargePages | 启用大型页 |
| -XX:+UseTransparentHugePages | 仅用于Linux，启用动态分配large pages |
| -XX:+AllowUserSignalHandlers | 允许用户处理应用消息 |
| -XX:VMOptionsFile=filename | 指定VM配置文件 |
| **基于HotSpot的JIT相关高级参数** | |
| -XX:AllocateInstancePrefetchLines=lines | 设置实例分配前预留缓存行数，默认1 |
| -XX:AllocatePrefetchDistance=size | 设置对象预分配空间，默认值-1 |
| -XX:AllocatePrefetchInstr=instruction | 在分配指针前预取指令数，默认0 |
| -XX:AllocatePrefetchLines=lines | 设置分配对象后预留缓存行，如果最后个对象非数组，默认1，否则是3 |
| -XX:AllocatePrefetchStepSize=size | 设置顺序指令每步骤预留空间大小，默认16b |
| -XX:AllocatePrefetchStyle=style | 产生预取指令方式  0:不发生预取指令  1:默认配置，在每次分配后，执行预取指令  2:用TLAB watermark pointer决定预取指令  3：每个缓存行产生一条预取指令 |
| -XX:+BackgroundCompilation | 启用后台编译 |
| -XX:CICompilerCount=threads | 设置编译线程数量  默认值是CPU数和内存大小 |
| -XX:+UseDynamicNumberOfCompilerThreads | 动态创建编译线程 |
| -XX:CompileCommand=command,method[,option] | 在执行指定方法时，执行指定命令 |
| -XX:CompileCommandFile=filename | 批量设置在执行指定方法时，执行指定命令 |
| -XX:CompilerDirectivesFile=file | 在程序启动是将指定指令加载到指令栈  需要配合  -XX:UnlockDiagnosticVMOptions |
| -XX:+CompilerDirectivesPrint | 当项目启动或指令加载时，打印指令栈 |
| -XX:CompileOnly=methods | 只编译指定的方法，多个用都好隔开 |
| -XX:CompileThresholdScaling=scale | 指定方法首次编译方式 |
| -XX:+DoEscapeAnalysis | 启用逃逸分析 |
| -XX:InitialCodeCacheSize=size | 设置初始缓存大小 |
| -XX:+Inline | 启用方法内联 |
| -XX:InlineSmallCode=size | 设置内联代码量 |
| -XX:+LogCompilation | 记录编译行为日志 |
| -XX:FreqInlineSize=size | 设置hot方法内联空间 |
| -XX:MaxInlineSize=size | 设置cold方法内联空间 |
| -XX:MaxTrivialSize=size | 设置trivial方法内联空间 |
| -XX:MaxNodeLimit=nodes | 设置单方法编译时节点数 |
| -XX:NonNMethodCodeHeapSize=size | 设置代码段包含非代码容量 |
| -XX:NonProfiledCodeHeapSize=size | 设置代码段包含无关方法大小  仅在设置-XX:SegmentedCodeCache时生效 |
| -XX:+OptimizeStringConcat | 优化string拼接 |
| -XX:+PrintAssembly | 打印汇编码 |
| -XX:ProfiledCodeHeapSize=size | 设置代码段包含profiled methods大小，  仅在设置-XX:SegmentedCodeCache时生效 |
| -XX:+PrintCompilation | 打印所有方法编译时日志 |
| -XX:+PrintInlining | 打印内联日志，可以查看哪些方法内联处理 |
| -XX:ReservedCodeCacheSize=size | 预留代码缓存，不能比  -XX:InitialCodeCacheSize小 |
| -XX:RTMAbortRatio=abort\_ratio | 指定RTM(Restricted Transactional Memory)中止比例 |
| -XX:RTMRetryCount=number\_of\_retries | 指定RTM 锁定代码重试次数 |
| -XX:+SegmentedCodeCache | 允许代码段缓存 |
| -XX:StartAggressiveSweepingAt=percent | 指定执行强制移除未使用代码的剩余空间阈值比例 |
| -XX:-TieredCompilation | 禁用分层编译 |
| -XX:UseSSE=version | 指定sse指令版本，仅适用86 |
| -XX:UseAVX=version | 指定AVX指令仅适用86 |
| -XX:+UseAES | 使用基于硬件的编译器内部AES |
| -XX:+UseAESIntrinsics | 使用编译器内部AES |
| -XX:+UseAESCTRIntrinsics | 使用编译器内部AES/CTR |
| -XX:+UseGHASHIntrinsics | 使用编译器内部GHASH，需要同时设置  -XX:+UnlockDiagnosticVMOptions |
| -XX:+UseBASE64Intrinsics | 使用编译器内部BASE64  -XX:+UnlockDiagnosticVMOptions |
| -XX:+UseAdler32Intrinsics | 使用编译器内部Adler32  -XX:+UnlockDiagnosticVMOptions |
| -XX:+UseCRC32Intrinsics | 同上 |
| -XX:+UseCRC32CIntrinsics | 同上 |
| -XX:+UseSHA | 同上 |
| -XX:+UseSHA1Intrinsics | 同上 |
| -XX:+UseSHA256Intrinsics | 同上 |
| -XX:+UseSHA512Intrinsics | 同上 |
| -XX:+UseMathExactIntrinsics | 同上 |
| -XX:+UseMultiplyToLenIntrinsic | 同上 |
| -XX:+UseSquareToLenIntrinsic | 同上 |
| -XX:+UseMulAddIntrinsic | 同上 |
| -XX:+UseMontgomeryMultiplyIntrinsic | 同上 |
| -XX:+UseMontgomerySquareIntrinsic | 同上 |
| -XX:+UseCMoveUnconditionally | 无条件生成CMove指令 |
| -XX:+UseCodeCacheFlushing | 关闭编译时，清空代码缓存 |
| -XX:+UseCondCardMark | 启用更新卡表前检查卡是否被标志，默认是关闭。在并发操作时，能提高性能 |
| -XX:+UseCountedLoopSafepoints | 将安全点保存在统计循环(counted loop)中 |
| -XX:LoopStripMiningIter=number\_of\_iterations | Controls the number of iterations in the inner strip mined loop |
| -XX:LoopStripMiningIterShortLoop=number\_of\_iterations | Controls loop strip mining optimization |
| -XX:+UseFMA | 使用编译器内部基于硬件的FMA指令 |
| -XX:+UseRTMDeopt | 自动调节RTM锁定 |
| -XX:+UseRTMLocking | 用RTM为重量级锁锁定代码 |
| -XX:+UseSuperWord | 向量化scalar操作 |
| **可用性相关参数** | |
| -XX:+DisableAttachMechanism | 禁用分析工具，如jcmd, jstack, jmap, jinfo等 |
| -XX:+ExtendedDTraceProbes | 启用动态追踪，仅非Win系统 |
| -XX:+HeapDumpOnOutOfMemoryError | 在OOM时，导出堆信息 |
| -XX:HeapDumpPath=path | 指定堆导出文件存储路径，默认为当前工作目录，文件名默认为java\_pid<pid>.hprof |
| -XX:LogFile=path | 设置虚拟机日志路径，默认为当前工作目录，默认文件名为hotspot.log. |
| -XX:+PrintClassHistogram | 打印类直方图，当收到指定系统事件时：  win：control+c  非win：control+break |
| -XX:+PrintConcurrentLocks | 打印锁信息，当收到同上系统事件时 |
| -XX:+PrintFlagsRanges | 打印测试信息 |
| -XX:+PerfDataSaveToFile | 当应用退出时，保存状态二进制信息到指定文件 |
| -XX:+UsePerfData | 启用perfdata功能 |
| **高级HotSpot参数** | |
| -XX:+AggressiveHeap | 根据CPU和RAM自动优化heap |
| -XX:+AlwaysPreTouch | 在OS请求之后，分配内存给应用之前，VM移动每一页(Requests the VM to touch every page on the Java heap after requesting it from the operating system and before handing memory out to the application.) |
| -XX:ConcGCThreads=threads | 并发GC线程数，默认是 |
| -XX:+DisableExplicitGC | 禁用System.gc() |
| -XX:+ExplicitGCInvokesConcurrent | System.gc()立即触发GC，仅在设置-XX:+UseG1GC时有效 |
| -XX:G1AdaptiveIHOPNumInitialSamples=number | 设置采集标本周期数， |
| -XX:G1HeapRegionSize=size | 设置G1堆的域大小，必须是2的倍数，区间1M-32M |
| -XX:G1HeapWastePercent=percent | 浪费堆内存比例，即当可回收比例小于此值时，不执行混合GC周期，默认5 |
| -XX:G1MaxNewSizePercent=percent | G1年轻代比例，默认60 |
| -XX:G1MixedGCCountTarget=number | 混合GCtarget数，默认8 |
| -XX:G1MixedGCLiveThresholdPercent=percent | 混合GC时，老年代占用比例，默认85 |
| -XX:G1NewSizePercent=percent | 新生代最小比例，默认5 |
| -XX:G1OldCSetRegionThresholdPercent=percent | 每个混合GC周期收集老年代比例，默认10 |
| -XX:G1ReservePercent=percent | 预留堆比例，默认10 |
| -XX:+G1UseAdaptiveIHOP | 使用自适应老年代占用空间 |
| -XX:InitialHeapSize=size | 堆初始化，值要么0，要么1024的倍数并大于1MB。当赋值0时，大小为新老两代空间之和 |
| -XX:InitialRAMPercentage=percent | 初始堆占用内存比例，默认值为1.5625 |
| -XX:InitialSurvivorRatio=ratio | Survivor区初始比例，默认值8，在设置-XX:+UseParallelGC 或-XX:+UseParallelOldGC时有效 |
| -XX:InitiatingHeapOccupancyPercent=percent | G1收集器老年代初始占用比例，默认45 |
| -XX:MaxGCPauseMillis=time | G1最大暂停时间，单位毫秒，默认值200 |
| -XX:MaxHeapSize=size | 最大堆，必须是1024的倍数且大于2M |
| -XX:MaxHeapFreeRatio=percent | 允许堆空闲比例，如果堆空闲大于此比例，则缩小堆，默认值是70 |
| -XX:MaxMetaspaceSize=size | 最大元数据空间，默认情况下无上限 |
| -XX:MaxNewSize=size | 新生代最大值 |
| -XX:MaxRAM=size | 最大物理内存占用空间，默认值是最大物理内存和128GB取最小值 |
| -XX:MaxRAMPercentage=percent | 最大物理内存占用 |
| -XX:MinRAMPercentage=percent | 最小物理内存占用 |
| -XX:MaxTenuringThreshold=threshold | 年轻代到老年代的阈值，最大/默认都是15 |
| -XX:MetaspaceSize=size | 设置元数据空间大小 |
| -XX:MinHeapFreeRatio=percent | 设置堆最小空闲比例，低于此比例，堆会扩大 |
| -XX:MinHeapSize=size | 设置堆最小值，值必须是0或者1024倍数且大于1MB，如果值为0，值与初始大小一样 |
| -XX:NewRatio=ratio | 新老年代比例，默认值为2 |
| -XX:NewSize=size | 年轻代初始大小，等同于-Xmn |
| -XX:ParallelGCThreads=threads | STW工作线程并发数 |
| -XX:+ParallelRefProcEnabled | 启用并发引用处理，默认禁用 |
| -XX:+PrintAdaptiveSizePolicy | 启用打印自适应大小信息，默认禁用 |
| -XX:+ScavengeBeforeFullGC | 启用每次FullGC前先尝试回收年轻代 |
| -XX:SoftRefLRUPolicyMSPerMB=time | 设置软引用在最后次被引用后存活时间，单位毫秒， |
| -XX:-ShrinkHeapInSteps | 禁用逐步缩小堆空间，每次都立即缩小到指定大小 |
| -XX:StringDeduplicationAgeThreshold=threshold | 设置String对象去重生效年龄，默认值是3 |
| -XX:SurvivorRatio=ratio | eden和survivor空间比例，默认是8 |
| -XX:TargetSurvivorRatio=percent | 设置年轻代GC后survivor空间比例，默认是50 |
| -XX:TLABSize=size | TLAB大小 |
| -XX:+UseAdaptiveSizePolicy | 开启自适应策略(Ergonomics) |
| -XX:+UseG1GC | 启用G1GC |
| -XX:+UseGCOverheadLimit | 启用GC消耗时间限制，如果GC时间消耗占用了98%且少于2%堆释放，则OOM |
| -XX:+UseNUMA | 使用NUMA（nonuniform memory architecture），默认关闭，仅在parallelGC下有效 |
| -XX:+UseParallelGC | 使用parallel scavenge GC |
| -XX:+UseSerialGC | 使用Serial GC |
| -XX:+UseSHM | 允许JVM使用共享存储设置大页，仅用于Linux |
| -XX:+UseStringDeduplication | 启用String对象去重，仅限G1有效 |
| -XX:+UseTLAB | 启用TLAB |
| -XX:+UseZGC | 启用ZGC |

## 2、javac

### 2.1文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/javac.html

### 2.2基础指令

javac [options] [sourcefiles-or-classnames]

### 2.3主要参数

|  |  |
| --- | --- |
| **Option** | |
| @filename | 从指定文件加载参数 |
| -A*key*[=value] | 配置注释处理器的参数，不被javac直接解释，如果有多个key/val，用点(.)分割 |
| --add-modules module | 指定附加模块，多个用逗号隔开 |
| --boot-class-path path  -bootclasspath path | 指定重写的引导类，仅适用与jdk9之前 |
| --class-path path  -classpath path  -cp path | 指定类库，会覆盖环境变量CLASSPATH |
| -d directory | 指定class文件存放根目录，如果不指定，则会存到源码文件所在目录 |
| -deprecation | 显示每个deprecated成员和类 |
| --enable-preview | 启用预览功能 |
| -encoding encoding | 指定编码 |
| -endorseddirs directories | 仅适用于jdk9之前 |
| -extdirs directories | 扩展包目录  多个目录：win用分号；分割，非win用冒号分割 |
| -g | 生成所有debug信息，默认情况下仅生成源文件名和行数 |
| -g: [lines, vars, source,none] | 生成指定的debug信息  lines：行号  vars：本地变量  source：源码文件  none:不输出 |
| -h directory | 指定头文件存放目录 |
| --help  -help  -? | 显示标准参数 |
| --help-extra  -X | 显示扩展参数 |
| -implicit:[none , class] | 是否生成隐式引用的类文件  none：不生成  class：自动生成 |
| -J*option* | 传送java命令的参数 |
| --limit-modules module | 限制可见模块，多个用逗号隔开 |
| --module module-name  -m module-name | 覆盖编译指定模块源文件 |
| --module-path path or -p path | 指定模块库路径 |
| --module-source-path module-source-path | 指定模块源码文件 |
| --module-version version | 指定模块版本 |
| -nowarn | 禁止警告信息 |
| -parameters | 生成方法参数元数据，用于可以通过反射获取方法参数原名 |
| -proc:[none,only] | 控制是否完成注解和编译  none：编译时不处理注解  only：只完成注解处理 |
| -processor class1 | 指定注解处理器，多个用逗号分开 |
| --processor-module-path path | 指定注解处理器所在模块 |
| --processor-path path  -processorpath path | 指定注解处理器所在目录 |
| -profile profile | 指定配置文件，仅适用于jdk9之前 |
| --release release |  |
| -s directory |  |
| --source release or -source release |  |
| --source-path path or -sourcepath path |  |
| --system jdk | none |  |
| --target release or -target release |  |
| --upgrade-module-path path |  |
| -verbose |  |
| --version or -version |  |
| -Werror |  |
| --add-exports module/package=other-module( |  |
| --add-reads module=other-module( |  |
| --default-module-for-created-files module-name |  |
| -Djava.endorsed.dirs=dirs |  |
| -Djava.ext.dirs=dirs |  |
| --doclint-format [html4|html5] |  |
| --patch-module module=path |  |
| -Xbootclasspath:path |  |
| -Xdiags:[compact |  |
| -Xdoclint |  |
| -Xlint |  |
| -Xlint:[-]key( |  |
| -Xmaxerrs number |  |
| -Xmaxwarns number |  |
| -Xpkginfo:[always |  |
| -Xplugin:name args |  |
| -Xprefer:[source |  |
| -Xprint |  |
| -XprintProcessorInfo |  |
| -XprintRounds |  |
| -Xstdout filename |  |
| **环境变量** | |
| CLASSPATH |  |
| JDK\_JAVAC\_OPTIONS |  |

## 3、jar

3.1、说明

创建Jar包，基于ZIP或ZLIB压缩格式

3.2、文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/jar.html

3.3、基础指令

jar [OPTION ...] [ [--release VERSION] [-C dir] files] ...

3.4、主要参数

|  |  |
| --- | --- |
| -c  --create | 创建归档 |
| -i=FILE  --generate-index=FILE | 产生索引文件 |
| -t  --list | 列出打包内容 |
| -u  --update | 更新覆盖已存在的包 |
| -x  --extract | 解压 |
| -d  --describe-module | 打印模块描述或名称 |
| -c DIR | 改变指定的目录，必须在命令行末尾 |
| -f=FILE  --file=FILE | 指定包名 |
| --release VERSION | 指定发布版本号，必须大于等于9的整数 |
| -v or --verbose | 打印全部信息 |
| 创建更新时参数 |  |
| -e=CLASSNAME  --main-class=CLASSNAME | 指定启动 |
| -m=FILE  --manifest=FILE | 指定清单信息文件 |
| -M  --no-manifest | 不创建清单文件 |
| --module-version=VERSION | 指定模块版本 |
| --hash-modules=PATTERN | 根据pattern计算哈希 |
| -p  --module-path | 指定生产哈希所依赖的模块 |
| @file | 从指定文件中加载参数 |
| -0  --no-compress | 生成不压缩的jar包 |

## 4、javap

4.1说明

a、反编译二进制文件(class)

b、无options时，反编译protected和public字段和方法

c、

4.2文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/javap.html

4.2基础指令

javap [options] classes...

4.3主要参数

|  |  |
| --- | --- |
| classes | 指定一个class文件，比如：  path/to/MyClass.class  jar:file:///path/to/MyJar.jar!/mypkg/MyClass.class  java.lang.Object |
| options | |
| - verbose  -v | 打印附加信息 |
| -l | 打印本地变量表 |
| -public | 仅打印public 方法和字段 |
| -protected | 仅打印protected和public方法和字段 |
| -private  -p | 打印所有方法和字段，包括private |
| -c | 反编译代码 |
| -s | 打印内部类 |
| -sysinfo | 展示系统信息 |
| -constants | 显示常量 |
| --module ${module}  -m ${module} | 指定类所在模块 |
| --module-path | 模块路径 |
| --system | 指定系统模块路径 |
| --class-path path  -classpath path  -cp path | 指定类路径，会覆盖CLASSPATH环境变量 |
| -bootclasspath | 指定启动类 |
| --multi-release | 指定jdk版本 |
| -J-${option} | 附带java参数，option参考java指令 |

# 五、LINUX命令(centos7)

## 1、top

a、说明

实时显示系统各个进程的资源占用情况

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -d ${time} | 间隔时间，单位秒 |
| -n ${max} | 抓取多少次 |
| -u  -U | 指定用户,或按u，输入指定用户名 |
| -p ${pid} | 指定pid |
| -o ${field} | 指定排序字段 |
| -w |  |
| -H | 从线程维度查看，或按h |
| -c | 显示命令完整路径,或按c |

c、显示参数说明



|  |  |  |
| --- | --- | --- |
| 第一部分 | | |
| 位置 | 参数 | 说明 |
| top | days | 系统已运行的时间 |
|  | users | 登陆用户数 |
|  | load average | 平均负载，1分钟，5分钟，15分钟 |
| Task | total | 进程总数 |
|  | running | 运行总数 |
|  | sleeping | 睡眠总数 |
|  | stopped | 停止总数 |
|  | zombie | 僵尸进程数(父进程已经退出,而该进程dead之后没有进程接受,就成为僵尸进程.(zombie)进程) |
| %Cpu | us | 用户占用率 |
|  | sy | 内核占用率 |
|  | ni | 用户改变过优先级的进程占用率 |
|  | id | 空闲CPU占用率 |
|  | wa | 等待中的CUP比例 |
|  | hi | 硬件中断请求占用率 |
|  | si | 软件中断请求占用率 |
|  | st | 被偷走百分比 |
| KiB Mem | total | 物理总量 |
|  | free | 空闲 |
|  | used | 使用 |
|  | buff/cache | 作为内核缓存的总量 |
| Kib Swap | total | 虚拟内存总量 |
|  | free | 空闲虚拟内存 |
|  | used | 虚拟内存使用量 |
|  | avail Mem | 可以用总量 |
| 第二部分 | | |
| head | PID | 进程号 |
|  | USER | 进程所有者用户 |
|  | PR | 优先级 |
|  | NI | nice值，负值表示高优先级，正值表示低优先级 |
|  | VIRT | 进程使用虚拟内存总量 |
|  | RES | 进程使用的、未被换出的物理内存大小 |
|  | SHR | 共享内存大小 |
|  | S | 进程状态。  D=不可中断的睡眠状态  R=运行  S=睡眠  T=跟踪/停止  Z=僵尸进程 |
|  | %CPU | CPU占比 |
|  | %MEM | 物理内存占比 |
|  | TIME+ | 使用CPU时间，单位 1/100秒 |
|  | COMMEND | 命令名/命令行 |

## 2、ps

a、说明

查看进程状态

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -A  -e | 显示所有进程 |
| -a | 显示除session leader进程之外的进程， |
| a | 显示所有用户的进程 |
| -d |  |
| -N  --deselect |  |
| r |  |
| T |  |
| x |  |
| -C |  |
| -G <GID>  --GROUP <GID> |  |
| -g <group>  --group <group> |  |
| -p <PID>  p <PID>  --pid <PID> |  |
| --ppid <PID> |  |
| -q <PID>  q  --quick-pid <PID> |  |
| -s  --sid <session> |  |
| -t <tty>  t <tty>  --tty <tty> |  |
| -u, U, --user <UID> |  |
| -U, --User <UID> |  |
| -F |  |
| -f |  |
| f  --forest |  |
| -H |  |
| -j |  |
| j |  |
| -l |  |
| l |  |
| -M  Z |  |
| -O <format> |  |
| O |  |
| -o <format>  o <format>  --format <format> |  |
| s |  |
| u |  |
| v |  |
| X |  |
| -y |  |
| --context |  |
| --headers |  |
| --no-headers |  |
| --cols <num>  --columns <num>  width <num> |  |
| --rows <num>  --lines <num> |  |
| H |  |
| -L |  |
| -m  m |  |
| -T |  |
| -c |  |
| c |  |
| e |  |
| k  --sort |  |
| L |  |
| n |  |
| S  --cumulative |  |
| -y |  |
| -V  V  --version |  |
| -w  w |  |

c、显示参数说明

## 3、vmstat

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -a  --active |  |
| -f  --forks |  |
| -m  --slabs |  |
| -n  --one-header |  |
| -s  --stats |  |
| -d  --disk |  |
| -D  --disk-sum |  |
| -p <dev>  --partition <dev> |  |
| -S <char>  --unit <char> |  |
| -w  --wide |  |
| -t  --timestamp |  |
| -V  --version |  |

c、显示参数说明

## 4、iostat

a、说明

通过iostat方便查看CPU、网卡、tty设备、磁盘、CD-ROM 等等设备的活动情况, 负载信息。它的特点是汇报磁盘活动统计情况，同时也会汇报出CPU使用情况。同vmstat一样，iostat也有一个弱点，就是它不能对某个进程进行深入分析，仅对系统的整体情况进行分析

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -c |  |
| -d |  |
| -h |  |
| -k |  |
| -m |  |
| -N |  |
| -t |  |
| -V |  |
| -x |  |
| -y |  |
| -z |  |
| -j { ID | LABEL | PATH | UUID | ... } |  |
| -T |  |
| -g <group> |  |
| -p |  |

c、显示参数说明

## 5、uptime

a、说明

打印系统总共运行了多长时间和系统的平均负载

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -p  --pretty |  |
| -s  --since |  |
| -V  -vesion |  |

c、显示参数说明

## 6、dmesg

a、说明

显示内核缓冲区系统控制信息的工具, 有时候屏幕上的启动信息一闪而过，我们无法查看到具体信息，又或者服务器在电信机房，更看不到开机启动信息。这时候linux提供了dmesg这条命令

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -C  --clear |  |
| -c  --read-clear |  |
| -D  --console-off |  |
| -d  --show-delta |  |
| -e  --reltime |  |
| -E  --console-on |  |
| -F <file>  --file <file> |  |
| -f <list>  --facility <list> |  |
| -H  --human |  |
| -k  --kernel |  |
| -L  --color |  |
| -l <list>  --level <list> |  |
| -n <level>  --console-level <level> |  |
| -P  --nopager |  |
| -r  --raw |  |
| -S  --syslog |  |
| -s  --buffer-size <size> |  |
| -T  --ctime |  |
| -t  --notime |  |
| -u  --userspace |  |
| -w  --follow |  |
| -x  --decode |  |
| -V  --version |  |

c、显示参数说明

## 7、mpstat

a、说明

实时系统监控工具。其报告与CPU的一些统计信息，这些信息存放在/proc/stat文件中。在多CPUs系统里，其不但能查看所有CPU的平均状况信息，而且能够查看特定CPU的信息。mpstat最大的特点是：可以查看多核心cpu中每个计算核心的统计数据

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -A |  |
| -u |  |
| -V |  |
| -I |  |
| -P <cpu> |  |

c、显示参数说明

## 8、sar

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -A |  |
| -B |  |
| -b |  |
| -C |  |
| -d |  |
| -F |  |
| -H |  |
| -h |  |
| -p |  |
| -q |  |
| -R |  |
| -r |  |
| -s |  |
| -t |  |
| -u |  |
| -V |  |
| -v |  |
| -W |  |
| -w |  |
| -y |  |
| -I |  |
| -P |  |
| -m |  |
| -n |  |
| -j |  |
| -f |  |
| -o |  |
| -i |  |
| -s <hh:mm:ss> |  |
| -e <hh:mm:ss> |  |

c、显示参数说明

## 9、free

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -b  -bytes |  |
| -k  --kilo |  |
| -g  --giga |  |
| --tera |  |
| --peta |  |
| -h  --human |  |
| --si |  |
| -l  --lohi |  |
| -t  -total |  |
| -s N  --seconds N |  |
| -c N  -count N |  |
| -w  --width |  |

c、显示参数说明

## 10、tcpdump

a、说明

根据使用者的定义对网络上的数据包进行截获的包分析工具

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -B <size> |  |
| -c <count> |  |
| -C <file\_size> |  |
| -E |  |
| -F <file> |  |
| -G <seconds> |  |
| -i <interface> |  |
| -j <tstamptype> |  |
| -M |  |
| --number |  |
| -Q  -P |  |
| -r <file> |  |
| -s |  |
| --time-stamp-precision |  |
| --immediate-mode |  |
| -T |  |
| -V <file> |  |
| -w <file> |  |
| -W |  |
| -y |  |
| -z |  |
| -Z |  |

c、显示参数说明

## 11、netstat

a、说明

显示各种网络相关信息，如网络连接，路由表，接口状态 (Interface Statistics)，masquerade 连接，多播成员 (Multicast Memberships) 等等

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -r  --routec |  |
| -I  --interfaces=<Iface> |  |
| -i  --interfaces |  |
| -g  --groups |  |
| -s  --statistics |  |
| -M  --masquerade |  |
|  |  |
| -v  --verbose |  |
| -W  --wide |  |
| -n  --numeric |  |
| --numeric-hosts |  |
| --numeric-ports |  |
| --numeric-users |  |
| -N  --symbolic |  |
| -e  --extend |  |
| -p  --programs |  |
| -o  --timers |  |
| -c  --continuous |  |
| -l  --listening |  |
| -a  --all |  |
| -F  --fib |  |
| -C  --cache |  |
| -Z  --context |  |

c、显示参数说明

## 12、pidstat

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -d |  |
| -h |  |
| -I |  |
| -l |  |
| -r |  |
| -s |  |
| -t |  |
| -U |  |
| -u |  |
| -V |  |
| -w |  |
| -C |  |
| -p |  |
| -T |  |

c、显示参数说明

## 13、grep

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -E  --extended-regexp |  |
| -F  --fixed-strings |  |
| -G  --basic-regexp |  |
| -P  --perl-regexp |  |
| -e  --regexp=PATTERN |  |
| -f  --file=FILE |  |
| -i  --ignore-case |  |
| -w  --word-regexp |  |
| -x  --line-regexp |  |
| -z  --null-data |  |
| -s  --no-messages |  |
| -v  --invert-match |  |
| -m  --max-count=NUM |  |
| -b  --byte-offset |  |
| -n  --line-number |  |
| --line-buffered |  |
| -H  --with-filename |  |
| -h  --no-filename |  |
| --label=label |  |
| -o  --only-matching |  |
| -q  --quiet  --silent |  |
| --binary-files=TYPE |  |
| -a  --text |  |
| -I |  |
| -d  --directories=ACTION |  |
| -D  --devices=ACTION |  |
| -r  --recursive |  |
| -R  --dereference-recursive |  |
| --include=FILE\_PATTERN |  |
| --exclude=FILE\_PATTERN |  |
| --exclude-from=FILE |  |
| --exclude-dir=PATTERN |  |
| -L  --files-without-match |  |
| -l  --files-with-matches |  |
| -c  --count |  |
| -T  --initial-tab |  |
| -Z  --null |  |
| -B  --before-context=NUM |  |
| -A  --after-context=NUM |  |
| -C  --context=NUM |  |
| -NUM |  |
| --group-separator=SEP use SEP as a group separator |  |
| --no-group-separator |  |
| --color[=WHEN] |  |
| --colour[=WHEN] |  |
| -U  --binary |  |
| -u  --unix-byte-offsets |  |

c、显示参数说明

## 14、cat

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -A  --show-all |  |
| -b  --number-nonblank |  |
| -e |  |
| -E  --show-ends |  |
| -n  --number |  |
| -s  --squeeze-blank |  |
| -t |  |
| -T  --show-tabs |  |
| -u |  |
| -v  --show-nonprinting |  |

## 15、tail

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -c  --bytes |  |
| -f  --follow |  |
| -F |  |
| -n  --lines |  |
| --max-unchanged-stats |  |
| --pid |  |
| -q  --quiet  --silent |  |
| --retry |  |
| -s  --sleep-interval |  |
| -v --verbose |  |

## 16、curl

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| --anyauth |  |
| -a --append |  |
| --basic |  |
| --cacert |  |
| --capath |  |
| -E  --cert |  |
| --cert-type |  |
| --ciphers |  |
| --compressed |  |
| -K  --config |  |
| --connect-timeout |  |
| -C  --continue-at |  |
| -b  --cookie |  |
| -c  --cookie-jar |  |
| --create-dirs |  |
| --crlf |  |
| --crlfile |  |
| -d  --data |  |
| --data-ascii |  |
| --data-binary |  |
| --data-urlencode |  |
| --delegation |  |
| --digest |  |
| --disable-eprt |  |
| --disable-epsv |  |
| -D  --dump-header |  |
| --egd-file |  |
| --engine |  |
| -f  --fail |  |
| -F  --form |  |
| --form-string |  |
| --ftp-account |  |
| --ftp-alternative-to-user |  |
| --ftp-create-dirs |  |
| --ftp-method |  |
| --ftp-pasv |  |
| -P  --ftp-port |  |
| --ftp-skip-pasv-ip |  |
| --ftp-pret |  |
| --ftp-ssl-ccc |  |
| --ftp-ssl-ccc-mode |  |
| --ftp-ssl-control |  |
| -G  --get |  |
| -g  --globoff |  |
| -H  --header |  |
| -I  --head |  |
| -h  --help |  |
| --hostpubmd5 |  |
| -0  --http1.0 |  |
| --ignore-content-length |  |
| -i  --include |  |
| -k  --insecure |  |
| --interface |  |
| -4  --ipv4 |  |
| -6  --ipv6 |  |
| -j  --junk-session-cookies |  |
| --keepalive-time |  |
| --key |  |
| --key-type |  |
| --krb |  |
| --libcurl |  |
| --limit-rate |  |
| -l  --list-only |  |
| --local-port |  |
| -L  --location |  |
| --location-trusted |  |
| -M  --manual |  |
| --mail-from |  |
| --mail-rcpt |  |
| --mail-auth |  |
| --max-filesize |  |
| --max-redirs |  |
| -m  --max-time |  |
| --metalink |  |
| --negotiate |  |
| -n  --netrc |  |
| --netrc-optional |  |
| --netrc-file |  |
| -N  --no-buffer |  |
| --no-keepalive |  |
| --no-sessionid |  |
| --noproxy |  |
| --ntlm |  |
| -o  --output |  |
| --pass |  |
| --post301 |  |
| --post302 |  |
| --post303 |  |
| -#  --progress-bar |  |
| --proto |  |
| --proto-redir |  |
| -x  --proxy |  |
| --proxy-anyauth |  |
| --proxy-basic |  |
| --proxy-digest |  |
| --proxy-negotiate |  |
| --proxy-ntlm |  |
| -U  --proxy-user |  |
| --proxy1.0 |  |
| -p  --proxytunnel |  |
| --pubkey |  |
| -Q  --quote |  |
| --random-file |  |
| -r  --range |  |
| --raw |  |
| -e  --referer |  |
| -J  --remote-header-name |  |
| -O  --remote-name |  |
| --remote-name-all |  |
| -R  --remote-time |  |
| -X  --request |  |
| --resolve |  |
| --retry |  |
| --retry-delay |  |
| --retry-max-time |  |
| -S  --show-error |  |
| -s  --silent |  |
| --socks4 |  |
| --socks4a |  |
| --socks5 |  |
| --socks5-basic |  |
| --socks5-gssapi |  |
| --socks5-hostname |  |
| --socks5-gssapi-service |  |
| --socks5-gssapi-nec |  |
| -Y  --speed-limit |  |
| -y  --speed-time |  |
| --ssl |  |
| --ssl-reqd |  |
| -2  --sslv2 |  |
| -3  --sslv3 |  |
| --ssl-allow-beast |  |
| --stderr |  |
| --tcp-nodelay |  |
| -t  --telnet-option |  |
| --tftp-blksize |  |
| -z  --time-cond |  |
| -1  --tlsv1 |  |
| --tlsv1.0 |  |
| --tlsv1.1 |  |
| --tlsv1.2 |  |
| --tlsv1.3 |  |
| --tls-max |  |
| --trace |  |
| --trace-ascii |  |
| --trace-time |  |
| --tr-encoding |  |
| -T  --upload-file |  |
| --url |  |
| -B  --use-ascii |  |
| -u  --user |  |
| --tlsuser |  |
| --tlspassword |  |
| --tlsauthtype |  |
| --unix-socket |  |
| -A  --user-agent |  |
| -v  --verbose |  |
| -V  --version |  |
| -w  --write-out |  |
| --xattr |  |
| -q |  |

## 17、ping

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -6 |  |
| -c |  |
| -i |  |
| -I |  |
| -m |  |
| -M |  |
| -l |  |
| -p |  |
| -Q |  |
| -s |  |
| -S |  |
| -t |  |
| -T |  |
| -w |  |
| -W |  |

c、显示结果说明

## 18、vim

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -v |  |
| -e |  |
| -E |  |
| -s |  |
| -d |  |
| -y |  |
| -R |  |
| -Z |  |
| -m |  |
| -M |  |
| -b |  |
| -l |  |
| -C |  |
| -N |  |
| -V[N][fname] |  |
| -D |  |
| -n |  |
| -r |  |
| -r |  |
| -L |  |
| -A |  |
| -H |  |
| -F |  |
| -T |  |
| -u |  |
| --noplugin |  |
| -p[N] |  |
| -o[N] |  |
| -O[N] |  |
| --cmd |  |
| -c |  |
| -S |  |
| -s |  |
| -w |  |
| -W |  |
| -x |  |
| --startuptime |  |
| -i |  |

e、编辑时快捷键

|  |  |
| --- | --- |
| 快捷键 | 说明 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# 附录

a堆分配参数示意图

