java8 docs http://docs.oracle.com/javase/8/

Java SE Tools Reference for UNIX http://docs.oracle.com/javase/8/docs/technotes/tools/unix/index.html

---

jinfo http://docs.oracle.com/javase/8/docs/technotes/tools/unix/jinfo.html#BCGEBFDD

观察进程运行环境参数，包括Java System属性和JVM命令行参数
系统崩溃了？jinfo可以从core文件里面知道崩溃的Java应用程序的配置信息。

```
[root@ezsonar166 monitoring]# jinfo
Usage:
    jinfo [option] <pid>
        (to connect to running process)
    jinfo [option] <executable <core>
        (to connect to a core file)
    jinfo [option] [server_id@]<remote server IP or hostname>
        (to connect to remote debug server)

where <option> is one of:
    -flag <name>         to print the value of the named VM flag
    -flag [+|-]<name>    to enable or disable the named VM flag
    -flag <name>=<value> to set the named VM flag to the given value
    -flags               to print VM flags
    -sysprops            to print Java system properties
    <no option>          to print both of the above
    -h | -help           to print this help message
```

jinfo -option pid 查看2788的MaxPerm大小可以用 jinfo -flag MaxPermSize 2788

---

jps http://docs.oracle.com/javase/8/docs/technotes/tools/unix/jps.html#CHDGHCGB

查看所有的jvm进程，包括进程ID，进程启动的路径等等。

```
[root@ezsonar166 monitoring]# jps -help
usage: jps [-help]
       jps [-q] [-mlvV] [<hostid>]

Definitions:
    <hostid>:      <hostname>[:<port>]
```

常用参数说明：
-m 输出传递给main方法的参数，如果是内嵌的JVM则输出为null。
-l 输出应用程序主类的完整包名，或者是应用程序JAR文件的完整路径。
-v 输出传给JVM的参数。

---

jstact http://docs.oracle.com/javase/8/docs/technotes/tools/unix/jstat.html#BEHBBBDJ

jstat利用JVM内建的指令对Java应用程序的资源和性能进行实时的命令行的监控，包括了对进程的classloader，compiler，gc情况；
特别的，一个极强的监视内存的工具，可以用来监视VM内存内的各种堆和非堆的大小及其内存使用量，以及加载类的数量。

```
[root@ezsonar166 monitoring]# jstat
invalid argument count
Usage: jstat -help|-options
       jstat -<option> [-t] [-h<lines>] <vmid> [<interval> [<count>]]

Definitions:
  <option>      An option reported by the -options option
  <vmid>        Virtual Machine Identifier. A vmid takes the following form:
                     <lvmid>[@<hostname>[:<port>]]
                Where <lvmid> is the local vm identifier for the target
                Java virtual machine, typically a process id; <hostname> is
                the name of the host running the target Java virtual machine;
                and <port> is the port number for the rmiregistry on the
                target host. See the jvmstat documentation for a more complete
                description of the Virtual Machine Identifier.
  <lines>       Number of samples between header lines.
  <interval>    Sampling interval. The following forms are allowed:
                    <n>["ms"|"s"]
                Where <n> is an integer and the suffix specifies the units as
                milliseconds("ms") or seconds("s"). The default units are "ms".
  <count>       Number of samples to take before terminating.
  -J<flag>      Pass <flag> directly to the runtime system.
```

Options:
- class：统计classloader的行为
- compiler：统计hotspot just-in-time编译器的行为
- gc：统计gc行为
- gccapacity：统计堆中代的容量、空间
- gccause：垃圾收集统计，包括最近引用垃圾收集的事件，基本同gcutil，比gcutil多了两列
- gcnew：统计新生代的行为
- gcnewcapacity：统计新生代的大小和空间
- gcold：统计旧生代的行为
- gcoldcapacity：统计旧生代的大小和空间
- gcpermcapacity：统计永久代的大小和空间
- gcutil：垃圾收集统计
- printcompilation：hotspot编译方法统计
- -h n 每n个样本，显示header一次
- -t n 在第一列显示时间戳列，时间戳时从jvm启动开始计算
- <vmid> 就是进程号
- <interval> interval是监控时间间隔，单位为微妙，不提供就意味着单次输出
- <count> count是最大输出次数,不提供且监控时间间隔有值的话， 就无限打印

class:

Column	Description
Loaded	被读入类的数量
Bytes	被读入的字节数（K）
Unloaded	被卸载类的数量
Bytes	被卸载的字节数（K）
Time	花费在load和unload类的时间

compiler:

Column	Description
Compiled	被执行的编译任务的数量
Failed	失败的编译任务的数量
Invalid	无效的编译任务的数量
Time	花费在执行编译任务的时间
FailedType	最近失败编译的编译类弄
FailedMethod	最近失败编译的类名和方法名

GC：

Column	Description
S0C	当前S0的容量 (KB)
S1C	当前S1的容量 (KB)
S0U	S0的使用 (KB)
S1U	S1的使用 (KB)
EC	当前eden的容量(KB)
EU	eden的使用 (KB)
OC	当前old的容量(KB)
OU	old的使用 (KB)
PC	当前perm的容量 (KB)
PU	perm的使用 (KB)
YGC	young代gc的次数
YGCT	young代gc花费的时间
FGC	full gc的次数
FGCT	full gc的时间
GCT	垃圾收集收集的总时间

gccapacity:

Column	Description
NGCMN	年轻代的最小容量 (KB)
NGCMX	年轻代的最大容量 (KB)
NGC	当前年轻代的容量 (KB)
S0C	当前S0的空间 (KB)
S1C	当前S1的空间 (KB)
EC	当前eden的空间 (KB)
OGCMN	年老代的最小容量 (KB)
OGCMX	年老代的最大容量 (KB)
OGC	当前年老代的容量 (KB)
OC	当前年老代的空间 (KB)
PGCMN	永久代的最小容量 (KB)
PGCMX	永久代的最大容量 (KB)
PGC	当前永久代的容量 (KB)
PC	当前永久代的空间 (KB)
YGC	年轻代gc的次数
FGC	

gccause:

Column	Description
LGCC	最近垃圾回收的原因
GCC	当前垃圾回收的原因

gcnew:

Column	Description
S0C	当前S0空间 (KB)
S1C	当前S1空间 (KB)
S0U	S0空间使用 (KB)
S1U	S1空间使用 (KB)
TT	Tenuring threshold
MTT	最大的tenuring threshold
DSS	希望的Survivor大小 (KB)
EC	当前eden空间 (KB)
EU	eden空间使用 (KB)
YGC	年轻代gc次数
YGCT	年轻代垃圾收集时间

gcnewcapacity:

Column	Description
NGCMN	最小的年轻代的容量 (KB)
NGCMX	最大的年轻代的容量 (KB)
NGC	当前年轻代的容量 (KB)
S0CMX	最大的S0空间 (KB)
S0C	当前S0空间 (KB)
S1CMX	最大的S1空间 (KB)
S1C	当前S1空间 (KB)
ECMX	最大eden空间 (KB)
EC	当前eden空间 (KB)
YGC	年轻代gc数量
FGC	full gc数量

gcold:

Column	Description
PC	当前perm空间 (KB)
PU	perm空间使用 (KB)
OC	当前old空间 (KB)
OU	old空间使用 (KB)
YGC	年轻代gc次数
FGC	full gc次数
FGCT	full gc时间
GCT	垃圾收集总时间

gcoldcapacity:

Column	Description
OGCMN	最小年老代容量 (KB)
OGCMX	最大年老代容量(KB)
OGC	当前年老代容量 (KB)
OC	当前年老代空间 (KB)
YGC	年轻代gc次数
FGC	full gc次数
FGCT	full gc时间
GCT	垃圾收集总时间

gcpermcapacity:

Column	Description
PGCMN	永久代最小容量 (KB)
PGCMX	永久代最大容量 (KB)
PGC	当前永久代的容量 (KB)
PC	当前永久代的空间 (KB)
YGC	年轻代gc次数
FGC	full gc次数
FGCT	full gc时间
GCT	垃圾收集总时间

gcutil:

Column	Description
S1	S1使用百分比
E	eden使用百分比
O	old使用百分比
P	perm使用百分比
YGC	年轻代gc次数
YGCT	年轻代gc时间
FGC	full gc次数
FGCT	full gc时间
GCT	垃圾收集总时间

printcompilation:

Column	Description
Compiled	被执行的编译任务的数量
Size	方法字节码的字节数
Type	编译类型
Method	编译方法的类名和方法名。类名使用"/" 代替 "." 作为空间分隔符. 方法名是给出类的方法名. 格式是一致于HotSpot –XX:+PrintComplation 选项

jstat -class 17970 1000 10 (每隔1秒监控一次，一共做10次)
jstat -gcutil 17970 1000 10 (按百分比显式)
jstat -compiler 17970 （显示VM实时编译的数量等信息）
jstat -gcold pid old对象的信息
jstat –gccapacity 可以显示，VM内存中三代（young,old,perm）对象的使用和占用大小，如：PGCMN显示的是最小perm的内存使用量，PGCMX显示的是perm的内存最大使用量，PGC是当前新生成的perm内存占用量，PC是但前perm内存占用量。其他的可以根据这个类推， OC是old内纯的占用量
jstat -gcoldcapacity pid old对象的信息及其占用量
jstat -printcompilation pid 当前VM执行的信息
jstat -printcompilation -h3 25917 1000 5 每1000毫秒打印一次，一共打印5次，还可以加上-h3每三行显示一下标题

---

jmap http://docs.oracle.com/javase/8/docs/technotes/tools/unix/jmap.html#CEGCECJB

监视进程运行中的jvm物理内存的占用情况，该进程内存内，所有对象的情况，例如产生了哪些对象，对象数量；
系统崩溃了？jmap 可以从core文件或进程中获得内存的具体匹配情况，包括Heap size, Perm size等等

```
[root@ezsonar166 monitoring]# jmap
Usage:
    jmap [option] <pid>
        (to connect to running process)
    jmap [option] <executable <core>
        (to connect to a core file)
    jmap [option] [server_id@]<remote server IP or hostname>
        (to connect to remote debug server)

where <option> is one of:
    <none>               to print same info as Solaris pmap
    -heap                to print java heap summary
    -histo[:live]        to print histogram of java object heap; if the "live"
                         suboption is specified, only count live objects
    -clstats             to print class loader statistics
    -finalizerinfo       to print information on objects awaiting finalization
    -dump:<dump-options> to dump java heap in hprof binary format
                         dump-options:
                           live         dump only live objects; if not specified,
                                        all objects in the heap are dumped.
                           format=b     binary format
                           file=<file>  dump heap to <file>
                         Example: jmap -dump:live,format=b,file=heap.bin <pid>
    -F                   force. Use with -dump:<dump-options> <pid> or -histo
                         to force a heap dump or histogram when <pid> does not
                         respond. The "live" suboption is not supported
                         in this mode.
    -h | -help           to print this help message
    -J<flag>             to pass <flag> directly to the runtime system
```

option参数如下：
- -dump:format=b,file=<filename> pid # dump堆到文件,format指定输出格式，live指明是活着的对象,file指定文件名
- -finalizerinfo # 打印等待回收对象的信息
- -heap # 打印heap的概要信息，GC使用的算法，heap的配置及wise heap的使用情况,可以用此来判断内存目前的使用情况以及垃圾回收情况
- -histo[:live] # 打印堆的对象统计，包括对象数、内存大小等等（因为在dump:live前会进行full gc，因此不加live的堆大小要大于加live堆的大小 ）
- -permstat # 打印classload类装载器和 jvm heap长久层的信息. 包含包括每个装载器的名字，活跃，地址，父装载器，和其总共加载的类大小。另外,内部String的数量和占用内存数也会打印出来.
- -F # 强制，强迫.在pid没有相应的时候使用-dump或者-histo参数. 在这个模式下,live子参数无效.
- -J # 传递参数给jmap启动的jvm. ，如：-J-Xms256m

使用jmap进行 heap dump的例子： jmap -dump:format=b,file=<filename> <pid> 观察运行中的jvm物理内存的占用情况
打印内存统计图：jmap -histo:live <pid>

---

jstack http://docs.oracle.com/javase/8/docs/technotes/tools/unix/jstack.html#BABGJDIF

观察jvm中当前所有线程的运行情况和线程当前状态。
系统崩溃了？如果java程序崩溃生成core文件，jstack工具可以用来获得core文件的java stack和native stack的信息，从而可以轻松地知道java程序是如何崩溃和在程序何处发生问题。
系统hung住了？jstack工具还可以附属到正在运行的java程序中，看到当时运行的java程序的java stack和native stack的信息, 如果现在运行的java程序呈现hung的状态，jstack是非常有用的。

```
[root@ezsonar166 monitoring]# jstack
Usage:
    jstack [-l] <pid>
        (to connect to running process)
    jstack -F [-m] [-l] <pid>
        (to connect to a hung process)
    jstack [-m] [-l] <executable> <core>
        (to connect to a core file)
    jstack [-m] [-l] [server_id@]<remote server IP or hostname>
        (to connect to a remote debug server)

Options:
    -F  to force a thread dump. Use when jstack <pid> does not respond (process is hung)
    -m  to print both java and native frames (mixed mode)
    -l  long listing. Prints additional information about locks
    -h or -help to print this help message
```

Options:
- -F 当’jstack [-l] pid’没有相应的时候强制打印栈信息
- -l 长列表. 打印关于锁的附加信息,例如属于java.util.concurrent的ownable synchronizers列表.
- -m 打印java和native c/c++框架的所有栈信息.

当程序出现死锁的时候，使用命令：`jstack pid > jstack.log`，然后在`jstack.log`文件中，搜索关键字“`BLOCKED`”，定位到引起死锁的地方

---

jhat http://docs.oracle.com/javase/8/docs/technotes/tools/unix/jhat.html#CIHHJAGE

用于对JAVA heap进行离线分析的工具，他可以对不同虚拟机中导出的heap信息文件进行分析，如LINUX上导出的文件可以拿到WINDOWS上进行分析，可以查找诸如内存方面的问题。
命令格式：jhat dumpfile(jmap生成的文件)

```
[root@ezsonar166 monitoring]# jhat -h
Usage:  jhat [-stack <bool>] [-refs <bool>] [-port <port>] [-baseline <file>] [-debug <int>] [-version] [-h|-help] <file>

-J<flag>          Pass <flag> directly to the runtime system. For
  example, -J-mx512m to use a maximum heap size of 512MB
-stack false:     Turn off tracking object allocation call stack.
-refs false:      Turn off tracking of references to objects
-port <port>:     Set the port for the HTTP server.  Defaults to 7000
-exclude <file>:  Specify a file that lists data members that should
  be excluded from the reachableFrom query.
-baseline <file>: Specify a baseline object dump.  Objects in
  both heap dumps with the same ID and same class will
  be marked as not being "new".
-debug <int>:     Set debug level.
    0:  No debug output
    1:  Debug hprof file parsing
    2:  Debug hprof file parsing, no server
-version          Report version number
-h|-help          Print this help and exit
<file>            The file to read

For a dump file that contains multiple heap dumps,
you may specify which dump in the file
by appending "#<number>" to the file name, i.e. "foo.hprof#3".

All boolean options default to "true"
```