# 概述

Linux系统上的/proc目录是一种文件系统，即proc文件系统。与其它常见的文件系统不同的是，/proc是一种伪文件系统（也即虚拟文件系统），存储的是当前内核运行状态的一系列特殊文件，用户可以通过这些文件查看有关系统硬件及当前正在运行进程的信息。而通过了解这些信息，能够帮助我们定位疑难问题。

# 目录

## /proc/cpuinfo

## /proc/sys

ramdisk文件系统基于磁盘模拟技术，实际文件系统是ex2 ex3等。sysfs是一种基于ram文件系统和proc一样。Sysfs文件系统是一个类似于proc文件系统的特殊文件系统，用于将系统中的设备组织成层次结构，并向用户模式程序提供详细的内核数据结构信息。其实，就是在用户态可以通过对sys文件系统的访问，来看内核态的一些驱动或者设备等。

参考：

<https://www.cnblogs.com/machangwei-8/p/10354636.html>

### /proc/sys/kernel/

1、/proc/sys/kernel/ctrl-alt-del  
 该文件有一个二进制值，该值控制系统在接收到ctrl+alt+delete按键组合时如何反应。这两个值分别是：  
 0值，表示捕获ctrl+alt+delete，并将其送至 init 程序；这将允许系统可以安全地关闭和重启，就好象输入shutdown命令一样。  
 1值，表示不捕获ctrl+alt+delete，将执行非正常的关闭，就好象直接关闭电源一样。  
 缺省设置：0  
 建议设置：1，防止意外按下ctrl+alt+delete导致系统非正常重启。  
 2、proc/sys/kernel/msgmax  
 该文件指定了从一个进程发送到另一个进程的消息的最大长度（bytes）。进程间的消息传递是在内核的内存中进行的，不会交换到磁盘上，所以如果增加该值，则将增加操作系统所使用的内存数量。  
 缺省设置：8192  
 3、/proc/sys/kernel/msgmnb  
 该文件指定一个消息队列的最大长度（bytes）。  
 缺省设置：16384  
 4、/proc/sys/kernel/msgmni  
 该文件指定消息队列标识的最大数目，即系统范围内最大多少个消息队列。  
 缺省设置：16  
 5、/proc/sys/kernel/panic  
 该文件表示如果发生“内核严重错误（kernel panic）”，则内核在重新引导之前等待的时间（以秒为单位）。  
 0秒，表示在发生内核严重错误时将禁止自动重新引导。  
 缺省设置：0  
 6、proc/sys/kernel/shmall  
 该文件表示在任何给定时刻，系统上可以使用的共享内存的总量（bytes）。  
 缺省设置：2097152  
 7、/proc/sys/kernel/shmmax  
 该文件表示内核所允许的最大共享内存段的大小（bytes）。  
 缺省设置：33554432  
 建议设置：物理内存 \* 50%  
 实际可用最大共享内存段大小=shmmax \* 98%，其中大约2%用于共享内存结构。  
 可以通过设置shmmax，然后执行ipcs -l来验证。  
 8、 /proc/sys/kernel/shmmni  
 该文件表示用于整个系统的共享内存段的最大数目（个）。  
 缺省设置：4096  
 9、/proc/sys/kernel/threads-max  
 该文件表示内核所能使用的线程的最大数目。  
 缺省设置：2048  
 10、/proc/sys/kernel/sem  
 该文件用于控制内核信号量，信号量是System VIPC用于进程间通讯的方法。  
 建议设置：250 32000 100 128  
 第一列，表示每个信号集中的最大信号量数目。  
 第二列，表示系统范围内的最大信号量总数目。  
 第三列，表示每个信号发生时的最大系统操作数目。  
 第四列，表示系统范围内的最大信号集总数目。  
 所以，（第一列）\*（第四列）=（第二列）  
 以上设置，可以通过执行ipcs -l来验证。

### /proc/sys/vm

1、admin\_reserve\_kbytes

给有cap\_sys\_admin权限的用户保留的内存数量，默认值是min(free pages \* 3%, 8MB)。这些内存是为了给管理员登录和杀死进程恢复系统提供足够的内存。

2、block\_dump

如果设置的是非零值，则会启用块I/O调试。更多的关于块I/O调试的信息可以参见Documentation/laptops/laptop-mode.txt。

3、compact\_memory

只有在启用了CONFIG\_COMPACTION选项才有效。当向该文件(/proc/sys/vm/compact\_memory)写入1时，所有的内存域都会被压缩，使空闲的内存尽可能形成连续的内存块。

4、dirty\_background\_bytes

当脏页所占的内存数量超过dirty\_background\_bytes时，内核的flusher线程开始回写脏页。

注意： dirty\_background\_bytes参数和 dirty\_background\_ratio参数是相对的，只能指定其中一个。当其中一个参数文件被写入时，会立即开始计算脏页限制，并且会将另一个参数的值清零。

5、dirty\_background\_ratio

当脏页所占的百分比（相对于所有可用内存，即空闲内存页+可回收内存页）达到dirty\_background\_ratio时，内核的flusher线程开始回写脏页数据。所有可用内存不等于总的系统内存。

6、dirty\_bytes

当脏页所占的内存数量达到dirty\_bytes时，执行磁盘写操作的进程自己开始回写脏数据。

注意：dirty\_bytes参数和dirty\_ratio参数是相对的，只能指定其中一个。当其中一个参数文件被写入时，会立即开始计算脏页限制，并且会将另一个参数的值清零

7、dirty\_expire\_centisecs

脏数据的过期时间，超过该时间后内核的flusher线程被唤醒时会将脏数据回写到磁盘上，单位是百分之一秒。

8、dirty\_ratio

当脏页所占的百分比（相对于所有可用内存，即空闲内存页+可回收内存页）达到dirty\_ratio时，执行磁盘写操作的进程会自己开始回写脏数据。所有可用内存不等于总的系统内存。

9、dirty\_writeback\_centisecs

内核的flusher线程会周期性地唤醒，然后将老的脏数据写到磁盘上。 dirty\_writeback\_centisecs定义了唤醒的间隔，单位是百分之一秒。如果设置为0，则禁止周期性地唤醒回写线程。

10、drop\_caches

向/proc/sys/vm/drop\_caches文件中写入数值可以使内核释放page cache，dentries和inodes缓存所占的内存。

只释放pagecache：

echo 1 >  /proc/sys/vm/drop\_caches

只释放dentries和inodes缓存：

echo 2 >  /proc/sys/vm/drop\_caches

释放pagecache、dentries和inodes缓存：

echo 3 >  /proc/sys/vm/drop\_caches

这个操作不是破坏性操作，脏的对象（比如脏页）不会被释放，因此要首先运行sync命令。

11、hugepages\_treat\_as\_movable

这个参数用来控制是否可以从ZONE\_MOVABLE内存域中分配大页面。如果设置为非零，大页面可以从ZONE\_MOVABLE内存域分配。ZONE\_MOVABLE内存域只有在指定了kernelcore启动参数的情况下才会创建，如果没有指定kernelcore启动参数， hugepages\_treat\_as\_movable参数则没有效果。

大页面迁移在某些情况下是可用的，这取决于系统架构和大页面的大小。如果大页面支持迁移，从ZONE\_MOVABLE内存域分配总是启用的，会忽略 hugepages\_treat\_as\_movable参数的值。换句话说， hugepages\_treat\_as\_movable参数只对不支持迁移的大页面有效。

假设在你的系统中大页面不能迁移，这个参数的一个用例是用户可以通过启用从ZONE\_MOVABLE内存域分配大页面的特性，让大页面池扩展性更好。这是因为在ZONE\_MOVABLE内存域，页面的回收、迁移和压缩工作的更好，也更容易获得连续的内存块。注意，为不能迁移的大页面使用ZONE\_MOVABLE会损害内存热移除等特性，因此需要用户自己来权衡。

12、lowmem\_reserve\_ratio

在有高端内存的机器上，从低端内存域给应用层进程分配内存是很危险的，因为这些内存可以通过mlock()系统锁定，或者变成不可用的swap空间。在有大量高端内存的机器上，缺少可以回收的低端内存是致命的。因此如果可以使用高端内存，Linux页面分配器不会使用低端内存。这意味着，内核会保护一定数量的低端内存，避免被用户空间锁定。

这个参数同样可以适用于16M的ISA DMA区域，如果可以使用低端或高端内存，则不会使用该区域。

lowmem\_reserve\_ratio参数决定了内核保护这些低端内存域的强度。预留的内存值和lowmem\_reserve\_ratio数组中的值是倒数关系，如果值是256，则代表1/256，即为0.39%的zone内存大小。如果想要预留更多页，应该设更小一点的值。更多的信息参考 [这里](http://kernel.taobao.org/index.php/Kernel_Documents/mm_sysctl" \t "https://blog.csdn.net/chongyang198999/article/details/_blank) 。

13、max\_map\_count

进程中内存映射区域的最大数量。在调用malloc，直接调用mmap和mprotect和加载共享库时会产生 内存映射区域 。虽然大多数程序需要的内存映射区域不超过1000个，但是特定的程序，特别是malloc调试器，可能需要很多，例如每次分配都会产生一到两个内存映射区域。默认值是65536。

14、memory\_failure\_early\_kill

该参数控制在某个内核无法处理的内存错误发生（由硬件检测到）时，如何来杀死进程。在某些情况下（例如内存页在磁盘上仍然有一个有效的副本），内核会透明地处理错误，不会影响任何程序。但是如果没有任何其他的副本，内核会杀死进程避免内存错误的影响扩大。

1：一旦内存错误被检测到，杀死所有和被损坏的、不能恢复的内存页相关的进程。注意，这里不支持几种类型的页面，例如内核内部分配的数据或者swap缓存，但是对用户层的大多数页面都支持。

0：从所有相关的进程中取消对被损坏页面的映射，只杀死试图访问该页面的进程。

杀死进程是通过发送SIGBUS信号，进程可以通过捕捉这个信号来处理。

只有在支持先进的机器检查处理机制的架构或平台上才能使用该特性，依赖于硬件的能力。

程序可以调用prctl系统调用的PR\_MCE\_KILL命令来重置这个配置。

15、memory\_failure\_recovery

启用内存错误恢复（如果平台支持的话）

1：尝试恢复

0：当发生内存错误时panic

16、min\_free\_kbytes

这个参数用来指定强制Linux VM保留的内存区域的最小值，单位是kb。VM会使用这个参数的值来计算系统中每个低端内存域的watermark[WMARK\_MIN]值。每个低端内存域都会根据这个参数保留一定数量的空闲内存页。

一部分少量的内存用来满足PF\_MEMALLOC类型的内存分配请求。如果进程设置了PF\_MEMALLOC标志，表示不能让这个进程分配内存失败，可以分配保留的内存。并不是所有进程都有的。kswapd、direct reclaim的process等在回收的时候会设置这个标志，因为回收的时候它们还要为自己分配一些内存。有了PF\_MEMALLOC标志，它们就可以获得保留的低端内存。

如果设置的值小于1024KB，系统很容易崩溃，在负载较高时很容易死锁。如果设置的值太大，系统会经常OOM。

17、min\_slab\_ratio

该参数只在NUMA内核中才有效，它是一个相对于每个内存域所有页面的百分比。

如果一个内存域中可以回收的slab页面所占的百分比（应该是相对于当前内存域的所有页面）超过min\_slab\_ratio，在回收区的slabs会被回收。这样可以确保即使在很少执行全局回收的NUMA系统中，slab的增长也是可控的。

注意，slab的回收在单个内存域或节点中发生。回收slab内存的进程现在并不是特定于某个节点，回收可能不及时。

默认的百分比是5。

18、min\_unmapped\_ratio

该参数只在NUMA内核中有效，它是一个相对于每个内存域所有页面的百分比。

只有在当前内存域中处于zone\_reclaim\_mode允许回收状态的内存页所占的百分比超过min\_unmapped\_ratio时，内存域才会执行回收操作。

如果zone\_reclaim\_mode的值是4，这个百分比（应该是百分比对应的页面数量）和所有基于文件的未映射页数比较，包括swapcache页和tmpfs文件。否则，只考虑基于普通文件的未映射页，不包括tmpfs文件。

默认的百分比是1。

19、mmap\_min\_addr

该参数定义了用户进程能够映射的最低内存地址。由于最开始的几个内存页面用于处理内核空引用错误，这些页面不允许写入。该参数的默认值是0，表示安全模块不需要强制保护最开始的页面。如果设置为64K，可以保证大多数的程序运行正常，避免比较隐蔽的内核BUG。

20、oom\_dump\_tasks

如果启用，在内核执行OOM-killing时会打印系统内进程的信息（不包括内核线程），信息包括pid、uid、tgid、vm size、rss、nr\_ptes，swapents，oom\_score\_adj和进程名称。这些信息可以帮助找出为什么OOM killer被执行，找到导致OOM的进程，以及了解为什么进程会被选中。

如果将参数置为0，不会打印系统内进程的信息。对于有数千个进程的大型系统来说，打印每个进程的内存状态信息并不可行。这些信息可能并不需要，因此不应该在OOM的情况下牺牲性能来打印这些信息。

如果设置为非零值，任何时候只要发生OOM killer，都会打印系统内进程的信息。

默认值是1（启用）。

21、oom\_kill\_allocating\_task

控制在OOM时是否杀死触发OOM的进程。

如果设置为0，OOM killer会扫描进程列表，选择一个进程来杀死。通常都会选择消耗内存内存最多的进程，杀死这样的进程后可以释放大量的内存。

如果设置为非零值，OOM killer只会简单地将触发OOM的进程杀死，避免遍历进程列表（代价比较大）。

如果panic\_on\_oom被设置，则会忽略oom\_kill\_allocating\_task的值。

默认值是0。

22、page-cluster

该参数控制一次写入或读出swap分区的页面数量。它是一个对数值，如果设置为0，表示1页；如果设置为1，表示2页；如果设置为2，则表示4页。如果设置为0，则表示完全禁止预读取。

默认值是3（一次8页）。如果swap比较频繁，调整该值的收效不大。

该参数的值越小，在处理最初的页面错误时延迟会越低。但如果随后的页面错误对应的页面也是在连续的页面中，则会有I/O延迟。

23、panic\_on\_oom

控制内核在OOM时是否panic。

如果设置为0，内核会杀死内存占用过多的进程。通常杀死内存占用最多的进程，系统就会恢复。

如果设置为1，在发生OOM时，内核会panic。然而，如果一个进程通过内存策略或进程绑定限制了可以使用的节点，并且这些节点的内存已经耗尽，oom-killer可能会杀死一个进程来释放内存。在这种情况下，内核不会panic，因为其他节点的内存可能还有空闲，这意味着整个系统的内存状况还没有处于崩溃状态。

如果设置为2，在发生OOM时总是会强制panic，即使在上面讨论的情况下也一样。即使在memory cgroup限制下发生的OOM，整个系统也会panic。

默认值是0。

将该参数设置为1或2，通常用于集群的故障切换。选择何种方式，取决于你的故障切换策略。

panic\_on\_oom=2和kdump一起使用，可以给你更多的信息，便于定位发生OOM的原因。

24、stat\_interval

VM统计信息更新的时间间隔，默认值是1s。

25、swappiness

该参数控制是否使用swap分区，以及使用的比例。设置的值越大，内核会越倾向于使用swap。如果设置为0，内核只有在看空闲的和基于文件的内存页数量小于内存域的高水位线（应该指的是watermark[high]）时才开始swap。

默认值是60。

26、vfs\_cache\_pressure

控制内核回收用于dentry和inode cache内存的倾向。

默认值是100，内核会根据pagecache和swapcache的回收情况，让dentry和inode cache的内存占用量保持在一个相对公平的百分比上。减小vfs\_cache\_pressure会让内核更倾向于保留dentry和inode cache。当vfs\_cache\_pressure等于0，在内存紧张时，内核也不会回收dentry和inode cache，这容易导致OOM。如果vfs\_cache\_pressure的值超过100，内核会更倾向于回收dentry和inode cache。

27、zone\_reclaim\_mode

该参数只有在启用CONFIG\_NUMA选项时才有效。

zone\_reclaim\_mode用来控制在内存域OOM时，如何来回收内存。如果设置为0，则禁止内存域回收，从系统中其他内存域或节点来满足内存分配请求。

这个参数可以使用下面的值通过或运算来设置：

1 = 启用内存域回收

2  =  刷脏页回收内存

4  = 通过swap回收内存

如果检测到从其他内存域分配内存会造成性能下降，启动的时候会将zone\_reclaim\_mode设置为1。页面分配器在从节点分配页面前，会先回收容易重用的页面（当前未使用的pagecache页面）。

如果系统是用作文件服务器，所有的内存都可以用作缓存磁盘上的文件，这种情况下关闭内存域回收是有好处的，缓存的作用要比数据的局部性作用大。

通过刷脏页进行内存域回收，会阻止写入大量数据的进程在其他节点上产生脏页。如果内存域填满了，会刷出脏页释放内存，因此可以使进程节流。这可能会降低单个进程的性能，因为它不能使用系统所有的内存来缓冲写入的数据，但是使用其他节点的进程不会受到影响。

通过swap释放内存可以将分配严格限制在本地节点，除非明确使用了特定的内存策略或绑定设置来覆盖这个设置。