Rockchip Linux 内存调试常用命令

文件标识：RK-KF-YF-140

发布版本：V1.0.0

日期：2021-01-06

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文档介绍Linux开发过程关于内存的常用命令，查看内存总大小信息、使用情况以及内存预留大小。

产品版本

芯片名称	内核版本
ARM系列芯片	4.19

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	许剑群	2021-01-06	初始版本

目录

1. total内存信息

1.1 meminfo

- cat /proc/meminfo

```
MemTotal:        1497696 kB
MemFree:          590524 kB
MemAvailable:     773252 kB
Buffers:             768 kB
Cached:           216892 kB
SwapCached:        14204 kB
Active:           260884 kB
Inactive:         176744 kB
Active(anon):     120620 kB
Inactive(anon):   106688 kB
Active(file):     140264 kB
Inactive(file):    70056 kB
Unevictable:        3224 kB
Mlocked:            3224 kB
SwapTotal:        748844 kB
SwapFree:         346156 kB
Dirty:               776 kB
Writeback:             0 kB
AnonPages:        220804 kB
Mapped:           194276 kB
```

```
Shmem:              4968 kB
KReclaimable:      43900 kB
Slab:             111060 kB
SReclaimable:      40648 kB
SUnreclaim:        70412 kB
KernelStack:       18496 kB
PageTables:        42096 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     1497692 kB
Committed_AS:   41751048 kB
VmallocTotal:   263061440 kB
VmallocUsed:       30368 kB
VmallocChunk:          0 kB
Percpu:             2528 kB
CmaTotal:         532480 kB
CmaFree:               0 kB
```

MemTotal = MemFree + Buffers + Cached

Swap：SwapTotal/SwapFree，是系统脚本中通过swap命令启用、分配

> swap 分区通常被称为交换分区，这是一块特殊的硬盘空间，即当实际内存不够用的时候，操作系统会从内存中取出一部分暂时不用的数据，放在交换分区中，从而为当前运行的程序腾出足够的内存空间

Slab：Slab当前使用大小，其中可回收SReclaimable:，不可回收SUnreclaim，如果增长太大，可能内存泄漏

Vmalloc：VmallocTotal是总的大小，VmallocUsed已用大小

Cma：CmaTotal总大小，CmaFree空闲大小。

> CMA被系统借用，也属于被使用，CmaFree会减小。

1.2 sysrq-trigger

- echo m > /proc/sysrq-trigger

```
Mem-Info:
active_anon:43194 inactive_anon:26284 isolated_anon:0
 active_file:35286 inactive_file:17959 isolated_file:0
 unevictable:806 dirty:0 writeback:0 unstable:0
 slab_reclaimable:10164 slab_unreclaimable:17630
 mapped:48836 shmem:1280 pagetables:10550 bounce:0
 free:134653 free_pcp:717 free_cma:0
Node 0 active_anon:172776kB inactive_anon:105136kB active_file:141144kB
inactive_file:71836kB unevictable:3224kB isolated(anon):0kB
isolated(file):0kB mapped:195344kB dirty:0kB writeback:0kB shmeo
DMA32 free:538612kB min:4884kB low:24024kB high:25520kB active_anon:172776kB
inactive_anon:105136kB active_file:141144kB inactive_file:71836kB
unevictable:3224kB writepending:0kB present:2095104kB
lowmem_reserve[]: 0 0 0
DMA32: 13625*4kB (UEH) 12210*8kB (UEH) 8286*16kB (UMEH) 4151*32kB (UMEH)
1157*64kB (UMEH) 223*128kB (UM) 46*256kB (U) 9*512kB (U) 2*1024kB (U)
0*2048kB 0*4096kB = 538612kB
```

```
58687 total pagecache pages
3564 pages in swap cache
Swap cache stats: add 155134, delete 151579, find 11930/52536
Free swap  = 346412kB
Total swap = 748844kB
523776 pages RAM
0 pages HighMem/MovableOnly
149352 pages reserved
133120 pages cma reserved
```

2. kernel内存信息

2.1 slab

内核申请的内存调试接口，编译需要打开两个宏定义

```
CONFIG_SLUB_SYSFS=y
CONFIG_SLUB_DEBUG=y
```

统计slab总大小命令

```
echo `cat /proc/slabinfo |awk 'BEGIN{sum=0;}{sum=sum+$3*$4;}END{print
sum/1024/1024}'` MB
```

2.1.1 kernfs_node_cache

如下命令看到kernfs使用了2.5M，655 pages = 2.55859375 MBytes

```
[root@RV1126_RV1109:/]# cat /proc/slabinfo |grep kernfs_node_cache
kernfs_node_cache  18319  18340    144   28    1 : tunables    0    0    0 :
slabdata    655    655      0
```

代码位置在`fs/kernfs/mount.c`

```
void __init kernfs_init(void)
{

    /*
     * the slab is freed in RCU context, so kernfs_find_and_get_node_by_ino
     * can access the slab lock free. This could introduce stale nodes,
     * please see how kernfs_find_and_get_node_by_ino filters out stale
     * nodes.
     */
    kernfs_node_cache = kmem_cache_create("kernfs_node_cache",
                          sizeof(struct kernfs_node),
                          0,
                          SLAB_PANIC | SLAB_TYPESAFE_BY_RCU,
                          NULL);
}
```

2.1.2 inode_cache

如下命令看到inode cache使用了3.2M，816 pages = 3.1875 MBytes

```
[root@RV1126_RV1109:/]# cat /proc/slabinfo |grep inode_cache
inode_cache          6282   6340    408   20    2 : tunables    0    0    0 :
slabdata    317    317      0
```

代码位置在`fs/inode.c`

```
void __init inode_init(void)
{
    /* inode slab cache */
    inode_cachep = kmem_cache_create("inode_cache",
                     sizeof(struct inode),
                     0,
                     (SLAB_RECLAIM_ACCOUNT|SLAB_PANIC|
                     SLAB_MEM_SPREAD|SLAB_ACCOUNT),
                     init_once);
}
```

2.2 memblock

系统预留内存调试接口，编译需要在bootargs中添加`memblock=debug`

```
chosen {
    bootargs = "earlycon=uart8250,mmio32,0xff570000 console=ttyFIQ0
memblock=debug";
};
```

```
[    0.000000] MEMBLOCK configuration:
[    0.000000]  memory size = 0x3fdb8000 reserved size = 0x11706eb8
[    0.000000]  memory.cnt  = 0x2
[    0.000000]  memory[0x0]     [0x00000000-0x083fffff], 0x08400000 bytes flags:
0x0
[    0.000000]  memory[0x1]     [0x08648000-0x3fffffff], 0x379b8000 bytes flags:
0x0
[    0.000000]  reserved.cnt  = 0x7
[    0.000000]  reserved[0x0]   [0x00004000-0x00007fff], 0x00004000 bytes flags:
0x0
[    0.000000]  reserved[0x1]   [0x00100000-0x00e32b2f], 0x00d32b30 bytes flags:
0x0
[    0.000000]  reserved[0x2]   [0x08000000-0x080fffff], 0x00100000 bytes flags:
0x0
[    0.000000]  reserved[0x3]   [0x08300000-0x08318fff], 0x00019000 bytes flags:
0x0
[    0.000000]  reserved[0x4]   [0x2dc00000-0x3dbfffff], 0x10000000 bytes flags:
0x0 CMA
[    0.000000]  reserved[0x5]   [0x3df00000-0x3dfb7387], 0x000b7388 bytes flags:
0x0
[    0.000000]  reserved[0x6]   [0x3f800000-0x3fffffff], 0x00800000 bytes flags:
0x0 CMA
```

其中预留比较大的有initrd（加载root使用）和node memmap（管理页内存使用）

initrd预留13MBytes

```
void __init arm_memblock_init(const struct machine_desc *mdesc)
{
    arm_initrd_init();
}
```

node memmap预留8MBytes

```
[    0.000000] On node 0 totalpages: 262006
[    0.000000] memblock_reserve: [0x3ef75000-0x3f774fff]
memblock_virt_alloc_internal+0x108/0x1a4
[    0.000000] alloc_node_mem_map: node 0, pgdat b0d4bd00, node_mem_map eef75000
[    0.000000]   Normal zone: 2048 pages used for memmap
[    0.000000]   Normal zone: 0 pages reserved
[    0.000000]   Normal zone: 262006 pages, LIFO batch:63
```

设备内存情况开机打印：

```
[    0.000000] Virtual kernel memory layout:
[    0.000000]     vector  : 0xffff0000 - 0xffff1000   (   4 kB)
[    0.000000]     fixmap  : 0xffc00000 - 0xfff00000   (3072 kB)
[    0.000000]     vmalloc : 0xf0800000 - 0xff800000   ( 240 MB)
[    0.000000]     lowmem  : 0xb0000000 - 0xf0000000   (1024 MB)
[    0.000000]     pkmap   : 0xafe00000 - 0xb0000000   (   2 MB)
[    0.000000]     modules : 0xaf000000 - 0xafe00000   (  14 MB)
[    0.000000]       .text : 0x(ptrval) - 0x(ptrval)   (9184 kB)
[    0.000000]       .init : 0x(ptrval) - 0x(ptrval)   (1024 kB)
[    0.000000]       .data : 0x(ptrval) - 0x(ptrval)   ( 332 kB)
[    0.000000]        .bss : 0x(ptrval) - 0x(ptrval)   ( 896 kB)
```

```
[    0.000000] Memory: 752320K/1048024K available (8192K kernel code, 331K
rwdata, 1900K rodata, 1024K init, 895K bss, 25368K reserved, 270336K cma-
reserved, 0K highmem)
```

752320K: 空闲页可用内存，`nr_free_pages() << (PAGE_SHIFT - 10)`

1048024K：总共可见物理内存，`physpages << (PAGE_SHIFT - 10)`

kernel code：内核代码，`codesize = _etext - _stext`

rwdata：可读可写数据段，`datasize = _edata - _sdata`

rodata：只读数据段，`rosize = __end_rodata - __start_rodata`

init：包括`init_data_size = __init_end - __init_begin`和`init_code_size = _einittext - _sinittext`

bss：用来存放程序中未初始化的全局变量的一块内存区域，`bss_size = __bss_stop - __bss_start`

reserved：memblock预留内存，`physpages - totalram_pages - totalcma_pages`

cma-reserved：CMA预留内存，`totalcma_pages`

```
[    0.000000] Reserved memory: created CMA memory pool at 0x3f800000, size 8 MiB
[    0.000000] OF: reserved mem: initialized node linux,cma, compatible id
shared-dma-pool
[    0.000000] Reserved memory: created CMA memory pool at 0x2dc00000, size 256
MiB
[    0.000000] OF: reserved mem: initialized node isp, compatible id shared-dma-
pool
```

2.2.1 memory

内核的内存支持多个block，Rockchip平台在进入内核前，会有部分内存作特殊用途，例如OPTEE/ATF的代码，需要放在132M～135M，如下大小为0x248000的内存块被用于OPTEE存放代码

```
[root@RV1126_RV1109:/]# cat /sys/kernel/debug/memblock/memory
   0: 0x00000000..0x083fffff
   1: 0x08648000..0x3fffffff
```

2.2.2 reserved

```
[root@RV1126_RV1109:/]# cat /sys/kernel/debug/memblock/reserved
   0: 0x00004000..0x00007fff
   1: 0x00100000..0x00e32b2f
   2: 0x08000000..0x080fffff
   3: 0x08300000..0x08318fff
   4: 0x083ff000..0x083fffff
   5: 0x2dc00000..0x3dbfffff
   6: 0x3df00000..0x3dfb7387
   7: 0x3ee41000..0x3ef3cfff
   8: 0x3ef3f480..0x3ef3f4f7
   9: 0x3ef3f500..0x3ef3f803
  10: 0x3ef3f834..0x3f7fefff
```

```
11: 0x3f7ff040..0x3f7ff384
12: 0x3f7ff3c0..0x3f7ff3fb
13: 0x3f7ff400..0x3f7ff583
14: 0x3f7ff5c0..0x3f7ff784
15: 0x3f7ff7c0..0x3f7ff837
16: 0x3f7ff840..0x3f7ff84f
17: 0x3f7ff880..0x3f7ff88f
18: 0x3f7ff8c0..0x3f7ff8c3
19: 0x3f7ff900..0x3f7ff903
20: 0x3f7ff940..0x3f7ffa4b
21: 0x3f7ffa80..0x3f7ffb8b
22: 0x3f7ffbc0..0x3f7ffccb
23: 0x3f7ffcf0..0x3f7ffd08
24: 0x3f7ffd0c..0x3f7ffd24
25: 0x3f7ffd28..0x3f7ffd72
26: 0x3f7ffd74..0x3f7ffd8e
27: 0x3f7ffd90..0x3f7ffdaa
28: 0x3f7ffdac..0x3f7ffdc6
29: 0x3f7ffdc8..0x3f7ffde2
30: 0x3f7ffde4..0x3f7ffdfe
31: 0x3f7ffe00..0x3f7ffedf
32: 0x3f7ffee8..0x3f7fff9f
33: 0x3f7fffb0..0x3fffffff
```

2.3 zoneinfo

- cat /proc/zoneinfo

```
Node 0, zone    DMA32
  per-node stats
      nr_inactive_anon 28250
      nr_active_anon 34245
      nr_inactive_file 29724
      nr_active_file 33943
      nr_unevictable 806
      nr_slab_reclaimable 10088
      nr_slab_unreclaimable 17495
      nr_isolated_anon 0
      nr_isolated_file 0
      workingset_refault 258089
      workingset_activate 54883
      workingset_restore 35120
      workingset_nodereclaim 4574
      nr_anon_pages 59384
      nr_mapped    65408
      nr_file_pages 70824
      nr_dirty     31
      nr_writeback 0
      nr_writeback_temp 0
      nr_shmem     1581
      nr_shmem_hugepages 0
      nr_shmem_pmdmapped 0
      nr_anon_transparent_hugepages 0
      nr_unstable  0
      nr_vmscan_write 148946
```

```
      nr_vmscan_immediate_reclaim 2540
      nr_dirtied   798484
      nr_written   938957
      nr_kernel_misc_reclaimable 0
      nr_unreclaimable_pages 0
      nr_ion_heap  0
      nr_ion_heap_pool 0
      nr_gpu_heap  0
  pages free     133183
        min      1221
        low      6006
        high     6380
        spanned  523776
        present  523776
        managed  374424
        protection: (0, 0, 0)
      nr_free_pages 133183
      nr_zone_inactive_anon 28250
      nr_zone_active_anon 34245
      nr_zone_inactive_file 29724
      nr_zone_active_file 33943
      nr_zone_unevictable 806
      nr_zone_write_pending 31
      nr_mlock     806
      nr_page_table_pages 10425
      nr_kernel_stack 18256
      nr_bounce    0
      nr_zspages   26061
      nr_free_cma  0
  pagesets
    cpu: 0
              count: 342
              high:  378
              batch: 63
  vm stats threshold: 30
    cpu: 1
              count: 301
              high:  378
              batch: 63
  vm stats threshold: 30
    cpu: 2
              count: 32
              high:  378
              batch: 63
  vm stats threshold: 30
    cpu: 3
              count: 296
              high:  378
              batch: 63
  vm stats threshold: 30
  node_unreclaimable:  0
  start_pfn:           512
Node 0, zone   Normal
  pages free     0
        min      0
        low      0
        high     0
        spanned  0
```

```
        present  0
        managed  0
        protection: (0, 0, 0)
Node 0, zone  Movable
  pages free     0
        min      0
        low      0
        high     0
        spanned  0
        present  0
        managed  0
        protection: (0, 0, 0)
```

3. 空闲内存

3.1 free

- total：是内核页管理的总内存；
- used：是指正在被使用的内存；
- free：是指空闲的内存；
- shared：是指共享的内存，如匿名页；
- buffers：是指缓冲内存数；
- cached：是指缓存内存数；

> buffers（缓冲）和 cached（缓存）的区别。cached 是给读取数据时加速的，buffers 是给写入数据加速的。cached 是指把读取出来的数据保存在内存中，当再次读取时，不用读取硬盘而直接从内存中读取，加速了数据的读取过程；buffers 是指在写入数据时，先把分散的写入操作保存到内存中，当达到一定程度后再集中写入硬盘，减少了磁盘碎片和硬盘的反复寻道，加速了数据的写入过程。

free

```
                 total        used        free      shared     buffers
  Mem:       1533640704  1308901376   224739328     6422528      921600
-/+ buffers/cache:     1307979776   225660928
  Swap:       766816256   401342464   365473792
```

free -m

```
                 total        used        free      shared     buffers
  Mem:             1462        1249         213           6           0
-/+ buffers/cache:           1248         214
  Swap:             731         382         348
```

free -h

```
                total        used        free      shared     buffers
Mem:             1.4G        1.2G        212M        6.1M        900K
-/+ buffers/cache:           1.2G        213M
Swap:            731M        383M        349M
```

4. 清除缓存

4.1 drop_caches

- echo x > /proc/sys/vm/drop_caches

  清空 pagecache

  ```
  echo 1 > /proc/sys/vm/drop_caches
  ```

  清空 dentries 和 inodes

  ```
  echo 2 > /proc/sys/vm/drop_caches
  ```

  清空所有缓存（pagecache、dentries 和 inodes）

  ```
  echo 3 > /proc/sys/vm/drop_caches
  ```

5. 虚拟内存使用情况

5.1 vmallocinfo

- cat /proc/vmallocinfo

  ```
  0x0000000000000000-0x0000000000000000    8192 bpf_jit_binary_alloc+0x70/0x110
  pages=1 vmalloc
  ...
  0x0000000000000000-0x0000000000000000   20480 start_kernel+0x330/0x4f0
  pages=4 vmalloc
  0x0000000000000000-0x0000000000000000    8192 of_iomap+0x4c/0xb8
  phys=0x00000000fdd00000 ioremap
  0x0000000000000000-0x0000000000000000   20480 start_kernel+0x330/0x4f0
  pages=4 vmalloc
  0x0000000000000000-0x0000000000000000    8192 of_iomap+0x4c/0xb8
  phys=0x00000000fdd20000 ioremap
  0x0000000000000000-0x0000000000000000   20480 start_kernel+0x330/0x4f0
  pages=4 vmalloc
  ```

5.2 vmstat

- cat /proc/vmstat

6. 内存压力

6.1 pressure

- cat /proc/pressure/memory
- **VSS**- Virtual Set Size 虚拟耗用内存（包含共享库占用的内存）
- **RSS**- Resident Set Size 实际使用物理内存（包含共享库占用的内存）
- **PSS**- Proportional Set Size 实际使用的物理内存（比例分配共享库占用的内存）
- **USS**- Unique Set Size 进程独自占用的物理内存（不包含共享库占用的内存）

一般来说内存占用大小有如下规律：VSS >= RSS >= PSS >= USS