lab8_oom 实验说明

本实验是为了学习 oom killer 机制。

书上说，写一个用户空间的 app 来触发 oom 机制，这个可能不太容易触发 OOM 机制。为此，我们写一个简单的内核驱动来触发，更加直观一些。

基本实验步骤

1. 编译 lab8 的内核模块，并且拷贝到 Qemu 虚拟机里面。
   运行内核模块，过一会就会出现 OOM Kill 的现象了。

```
/mnt # insmod alloc-oom.ko
insmod invoked oom-killer: gfp_mask=0xd0, order=0, oom_score_adj=0
insmod cpuset=/ mems_allowed=0
CPU: 0 PID: 705 Comm: insmod Tainted: G           O    4.0.0+ #2
Hardware name: ARM-Versatile Express
[<c002475c>] (unwind_backtrace) from [<c001d640>] (show_stack+0x2c/0x38)
[<c001d640>] (show_stack) from [<c05956e4>] (__dump_stack+0x1c/0x24)
[<c05956e4>] (__dump_stack) from [<c05957bc>] (dump_stack+0xd0/0xf8)
[<c05957bc>] (dump_stack) from [<c01a81b4>] (dump_header+0x104/0x158)
[<c01a81b4>] (dump_header) from [<c01a87a4>] (oom_kill_process+0x208/0xa48)
[<c01a87a4>] (oom_kill_process) from [<c01a99a8>] (__out_of_memory+0x410/0x43c)
[<c01a99a8>] (__out_of_memory) from [<c01a9a38>] (out_of_memory+0x64/0x88)
[<c01a9a38>] (out_of_memory) from [<c01b08b4>] (__alloc_pages_nodemask+0xed8/0x1208)
[<c01b08b4>] (__alloc_pages_nodemask) from [<bf00209c>] (my_init+0x9c/0xf0 [alloc_oom])
[<bf00209c>] (my_init [alloc_oom]) from [<c0008dc8>] (do_one_initcall+0x68/0x190)
[<c0008dc8>] (do_one_initcall) from [<c0119af8>] (do_init_module+0xb4/0x278)
[<c0119af8>] (do_init_module) from [<c011a4ac>] (load_module+0x3ec/0x570)
[<c011a4ac>] (load_module) from [<c011a6d8>] (SyS_init_module+0xa8/0xc0)
[<c011a6d8>] (SyS_init_module) from [<c0014d40>] (ret_fast_syscall+0x0/0x34)
Mem-info:
Normal per-cpu:
CPU    0: hi:  186, btch:  31 usd:  69
CPU    1: hi:  186, btch:  31 usd: 160
CPU    2: hi:  186, btch:  31 usd:  10
CPU    3: hi:  186, btch:  31 usd: 154
```

进阶思考

本实验重点是希望读者可以对照代码来分析 oom killer 的 log。

比如：

```
/mnt # insmod alloc-oom.ko
insmod invoked oom-killer: gfp_mask=0xd0, order=0, oom_score_adj=0
insmod cpuset=/ mems_allowed=0
```

这句话是在 oom_kill_process()->dump_header()函数里打印出来的，告诉你那个进程触发的这次 oom killer，这里显示是 insmod 进程，它在分配多少的物理页面的时候触发的，分配的掩码是啥，当时的 oom_score_adj 的值是啥。

接下来。

```
CPU: 0 PID: 705 Comm: insmod Tainted: G           O    4.0.0+ #2
Hardware name: ARM-Versatile Express
[<c002475c>] (unwind_backtrace) from [<c001d640>] (show_stack+0x2c/0x38)
[<c001d640>] (show_stack) from [<c05956e4>] (__dump_stack+0x1c/0x24)
[<c05956e4>] (__dump_stack) from [<c05957bc>] (dump_stack+0xd0/0xf8)
[<c05957bc>] (dump_stack) from [<c01a81b4>] (dump_header+0x104/0x158)
[<c01a81b4>] (dump_header) from [<c01a87a4>] (oom_kill_process+0x208/0xa48)
[<c01a87a4>] (oom_kill_process) from [<c01a99a8>] (__out_of_memory+0x410/0x43c)
[<c01a99a8>] (__out_of_memory) from [<c01a9a38>] (out_of_memory+0x64/0x88)
[<c01a9a38>] (out_of_memory) from [<c01b08b4>] (__alloc_pages_nodemask+0xed8/0x1208)
[<c01b08b4>] (__alloc_pages_nodemask) from [<bf00209c>] (my_init+0x9c/0xf0 [alloc_oom]
[<bf00209c>] (my_init [alloc_oom]) from [<c0008dc8>] (do_one_initcall+0x68/0x190)
[<c0008dc8>] (do_one_initcall) from [<c0119af8>] (do_init_module+0xb4/0x278)
[<c0119af8>] (do_init_module) from [<c011a4ac>] (load_module+0x3ec/0x570)
[<c011a4ac>] (load_module) from [<c011a6d8>] (SyS_init_module+0xa8/0xc0)
[<c011a6d8>] (SyS_init_module) from [<c0014d40>] (ret_fast_syscall+0x0/0x34)
```

这是在 oom_kill_process()->dump_header()->dump_stack()函数里打印的，这也是打印触发这次 oom killer 的函数调用栈的关系。

接下来：

```
Mem-info:
Normal per-cpu:
CPU    0: hi:  186, btch:  31 usd:  69
CPU    1: hi:  186, btch:  31 usd: 160
CPU    2: hi:  186, btch:  31 usd:  10
CPU    3: hi:  186, btch:  31 usd: 154
HighMem per-cpu:
CPU    0: hi:   90, btch:  15 usd:  15
CPU    1: hi:   90, btch:  15 usd:  27
CPU    2: hi:   90, btch:  15 usd:   4
CPU    3: hi:   90, btch:  15 usd:   8
active_anon:559 inactive_anon:1065 isolated_anon:0
 active_file:0 inactive_file:0 isolated_file:0
 unevictable:0 dirty:0 writeback:0 unstable:0
 free:66636 slab_reclaimable:584 slab_unreclaimable:986
 mapped:343 shmem:1596 pagetables:10 bounce:0
 free_cma:0
Normal free:3348kB min:3468kB low:4332kB high:5200kB active_anon:92kB inactive_anon:0kB active_file:0kB inactiv
e_file:0kB unevictable:0kB isolated(anon):0kB isolated(file):0kB present:778240kB managed:755144kB mlocked:0kB
dirty:0kB writeback:0kB mapped:0kB shmem:4kB slab_reclaimable:2336kB slab_unreclaimable:3944kB kernel_stack:384
kB pagetables:40kB unstable:0kB bounce:0kB free_cma:0kB writeback_tmp:0kB pages_scanned:0 all_unreclaimable? ye
s
lowmem_reserve[]: 0 2112 2112
HighMem free:263196kB min:264kB low:572kB high:884kB active_anon:2144kB inactive_anon:4260kB active_file:0kB in
active_file:0kB unevictable:0kB isolated(anon):0kB isolated(file):0kB present:270336kB managed:270336kB mlocked
:0kB dirty:0kB writeback:0kB mapped:1372kB shmem:6380kB slab_reclaimable:0kB slab_unreclaimable:0kB kernel_stac
k:0kB pagetables:0kB unstable:0kB bounce:0kB free_cma:0kB writeback_tmp:0kB pages_scanned:0 all_unreclaimable?
yes
lowmem_reserve[]: 0 0 0
Normal: 1*4kB (R) 0*8kB 1*16kB (R) 0*32kB 0*64kB 0*128kB 1*256kB (R) 0*512kB 1*1024kB (R) 1*2048kB (R) 0*4096kB
 = 3348kB
HighMem: 1*4kB (U) 1*8kB (M) 1*16kB (M) 2*32kB (UM) 1*64kB (U) 1*128kB (U) 1*256kB (U) 1*512kB (M) 2*1024kB (UM
) 1*2048kB (U) 63*4096kB (MR) = 263196kB
1596 total pagecache pages
0 pages in swap cache
Swap cache stats: add 0, delete 0, find 0/0
Free swap  = 0kB
Total swap = 0kB
262144 pages of RAM
67147 free pages
5774 reserved pages
881 slab pages
852 pages shared
0 pages swap cached
```

这么一大段都是在 oom_kill_process()->dump_header()->show_mem ()函数里打印的，告诉你现在触发 oom killer 的那个时刻，系统所有的内存系统，包括有多少匿名页面，多少 page cache 等等。

```
[ pid ]   uid  tgid total_vm      rss nr_ptes nr_pmds swapents oom_score_adj name
[  702]     0   702      577        1       3       0        0             0 sh
[  705]     0   705      577        1       3       0        0             0 insmod
```

这里是在 oom_kill_process()->dump_header()->dump_tasks ()函数里打印的，列举出有可能被杀的进程。

```
Out of memory: Kill process 702 (sh) score 0 or sacrifice child
Killed process 705 (insmod) total-vm:2308kB, anon-rss:4kB, file-rss:0kB
```

这里是在 oom_kill_process()函数打印的，告诉你，我现在选择 705 进程来杀了，痛快！

接下来：

```
insmod: page allocation failure: order:0, mode:0xd0
CPU: 0 PID: 705 Comm: insmod Tainted: G           O    4.0.0+ #2
Hardware name: ARM-Versatile Express
[<c002475c>] (unwind_backtrace) from [<c001d640>] (show_stack+0x2c/0x38)
[<c001d640>] (show_stack) from [<c05956e4>] (__dump_stack+0x1c/0x24)
[<c05956e4>] (__dump_stack) from [<c05957bc>] (dump_stack+0xd0/0xf8)
[<c05957bc>] (dump_stack) from [<c01af0f0>] (warn_alloc_failed+0x1a8/0x1e4)
[<c01af0f0>] (warn_alloc_failed) from [<c01b0a34>] (__alloc_pages_nodemask+0x1058/0x1208)
[<c01b0a34>] (__alloc_pages_nodemask) from [<bf00209c>] (my_init+0x9c/0xf0 [alloc_oom])
[<bf00209c>] (my_init [alloc_oom]) from [<c0008dc8>] (do_one_initcall+0x68/0x190)
[<c0008dc8>] (do_one_initcall) from [<c0119af8>] (do_init_module+0xb4/0x278)
[<c0119af8>] (do_init_module) from [<c011a4ac>] (load_module+0x3ec/0x570)
[<c011a4ac>] (load_module) from [<c011a6d8>] (SyS_init_module+0xa8/0xc0)
[<c011a6d8>] (SyS_init_module) from [<c0014d40>] (ret_fast_syscall+0x0/0x34)
```

杀完之后，insmod 进程有发现分配内存失败了，又继续重复刚才的故事。。。

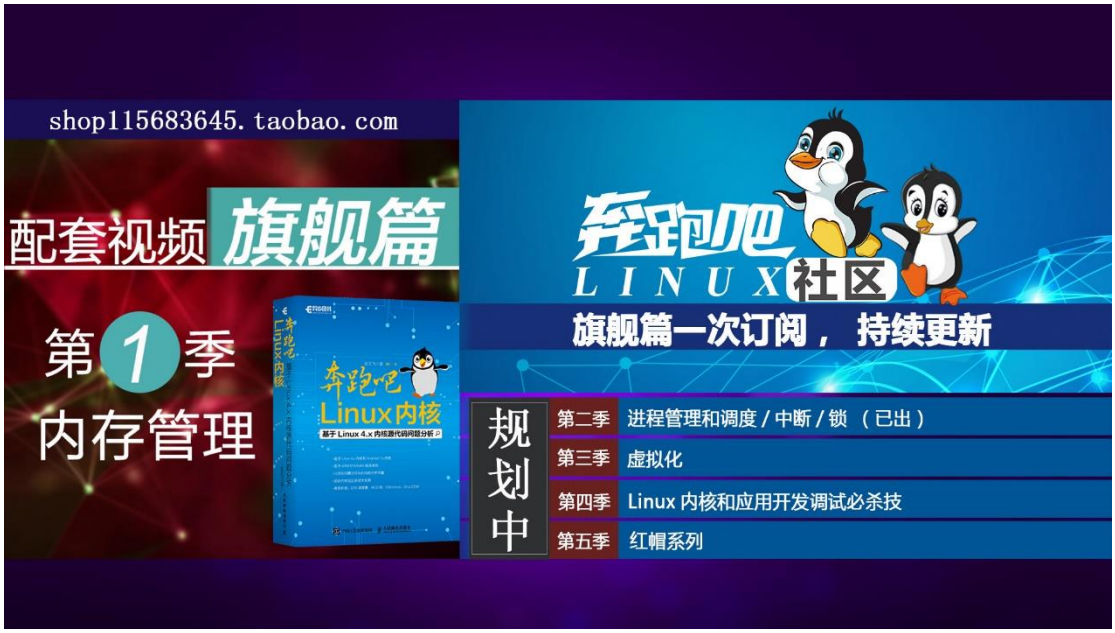
进阶思考：

我们从 log 里看到：

```
Swap cache stats: add 0, delete 0, find 0/0
Free swap  = 0kB
Total swap = 0kB
262144 pages of RAM
67147 free pages
5774 reserved pages
881 slab pages
852 pages shared
0 pages swap cached
```

系统明明空闲页面还有 67147 个，为啥我分配一个页面都会让失败呢？

笨叔在这里没有太详细解析每个参数，如果大家对这个问题感兴趣，可以关注笨叔的第一季旗舰篇视频，笨叔会在视频中和大家详细解答。


shop115683645.taobao.com
配套视频 旗舰篇
第1季
内存管理
奔跑吧 Linux内核
基于 Linux 4.x 内核源代码问题分析
奔跑吧 LINUX社区
旗舰篇一次订阅， 持续更新
规划中
第二季 进程管理和调度 / 中断 / 锁 （已出）
第三季 虚拟化
第四季 Linux 内核和应用开发调试必杀技
第五季 红帽系列


shop115683645.taobao.com
Linux视频课程
奔跑吧 LINUX社区
微信公众号： 奔跑吧 linux 社区
1. 一键订阅， 持续更新
2. 最有深度和广度的 Linux 视频
3. 手把手解读 Linux 内核代码
4. 紧跟 Linux 开源社区技术热点
5. 笨叔叔的 VIP 私密群答疑
6. 图书 + 视频， 全新学习模式

微店：

微店 奔跑吧Linux内核

扫码识别

淘宝店：https://shop115683645.taobao.com/

微信公众号：