

lab3_procfs 实验说明

本实验的目的：写一个内核模块，利用 procfs 的 API 接口函数来创建新的节点，然后在节点里和用户空间进行数据交互。

基本实验步骤

1. 进入 `rlk_lab/rlk_basic/chapter_10/lab1` 目录。

```
# export ARCH=arm
# export CROSS_COMPILE=arm-linux-gnueabi-
# make BASEINCLUDE=/home/figo/work/runninglinuxkernel/runninglinuxkernel_4.0
```

这里 `BASEINCLUDE` 指定你当前 `runninglinuxkernel_4.0` 的目录路径。

然后把 `*.ko` 拷贝到 `runninglinuxkernel_4.0/kmodules` 目录下。

运行如下脚本启动 Qemu。

```
#cd runninglinuxkernel_4.0
# sh run.sh arm32 #启动虚拟机
```

在 Qemu 虚拟机:

```
#cd /mnt
# insmod proc-test.ko
```

```
/mnt # insmod proc-test.ko
I created benshushu/my_proc
```

在 `proc` 目录下面创建了一个名为 `benshushu` 的新的目录，然后新建了一个 `my_proc` 的节点。

`my_proc` 节点有一个默认值。

```
/proc # cd benshushu/
/proc/benshushu # ls
my_proc
/proc/benshushu # cat my_proc
100
```

通过 `echo` 命令来往这个节点里写入新的值。

```
/proc/benshushu # echo 200 > my_proc
param has been set to 200
/proc/benshushu # cat my_proc
200
/proc/benshushu #
```

进阶思考

创建 `procfs` 是内核调试或者说内核空间 and 用户空间进行交换的一个重要的手段。

内核里有不少全局的变量值，存放在 `procfs` 里面，有三个目录的节点，是值得我们去学习和研究的，特别是做运维和系统调优的朋友们。

1. `/proc/sys/kernel` 目录，里面存放了内核核心的调优参数
2. `/proc/sys/vm` 目录，里面存放了内核内存管理相关的调优参数
3. `/proc/pid/` 目录，这里 `pid` 指的是具体的进程的 `pid`，这里面存放的是每个进程相关的调优参数。

```
/proc/sys/vm # ls
admin_reserve_kbytes      max_map_count
block_dump                min_free_kbytes
compact_memory            mmap_min_addr
dirty_background_bytes    nr_pdflush_threads
dirty_background_ratio    oom_dump_tasks
dirty_bytes               oom_kill_allocating_task
dirty_expire_centisecs    overcommit_kbytes
dirty_ratio               overcommit_memory
dirty_writeback_centisecs overcommit_ratio
dirtytime_expire_seconds page-cluster
drop_caches               panic_on_oom
extfrag_threshold         percpu_pagelist_fraction
highmem_is_dirtyable      stat_interval
laptop_mode               swappiness
legacy_va_layout          user_reserve_kbytes
lowmem_reserve_ratio       vfs_cache_pressure
```

笨叔在已经录制好的视频节目里，有对 `/proc/sys/vm` 目录的每个节点都做了详细的介绍，有兴趣的朋友，可以关注笨叔的第一季旗舰篇的视频节目。

Linux内核中内存调优参数

笨叔叔

第一类 和cache和页面回收相关的优化参数

1. 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。

2. dirty_ratio

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，write调用会唤醒内核的flusher线程开始回写脏页数据，直到脏页比例低于此值，注意write调用此时会阻塞。

3. 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，write调用会唤醒内核的flusher线程开始回写脏页数据，直到脏页比例低于此值，与dirty_ratio不同，write调用此时并不会阻塞。也就是内核的flusher这个内核线程 在回写脏页面。

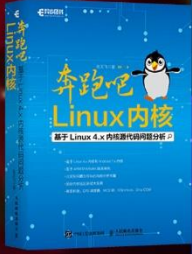
要注意这里说的 可用物理内存 不等于系统的全部内存。

如果大家对这些问题感兴趣，可以关注笨叔叔的旗舰篇视频，笨叔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 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



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：



扫码识别

淘宝店: <https://shop115683645.taobao.com/>

微信公众号:

