lab7_pin_page 实验说明

本实验是为了验证和学习如何去 pin 一个 page，也就是熟悉 get_user_pages()和 get_user_pages_fast()的使用。这两个接口函数在内核代码或者驱动代码中很常见。

基本实验步骤

1. 进入 runninglinuxkernel_4.0/rlk_lab/rlk_basic/chapter_7_mm/lab7_pin_page 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。

编译 test 测试 app。

```
# arm-linux-gnueabi-gcc test_ok.c -o test_ok
```

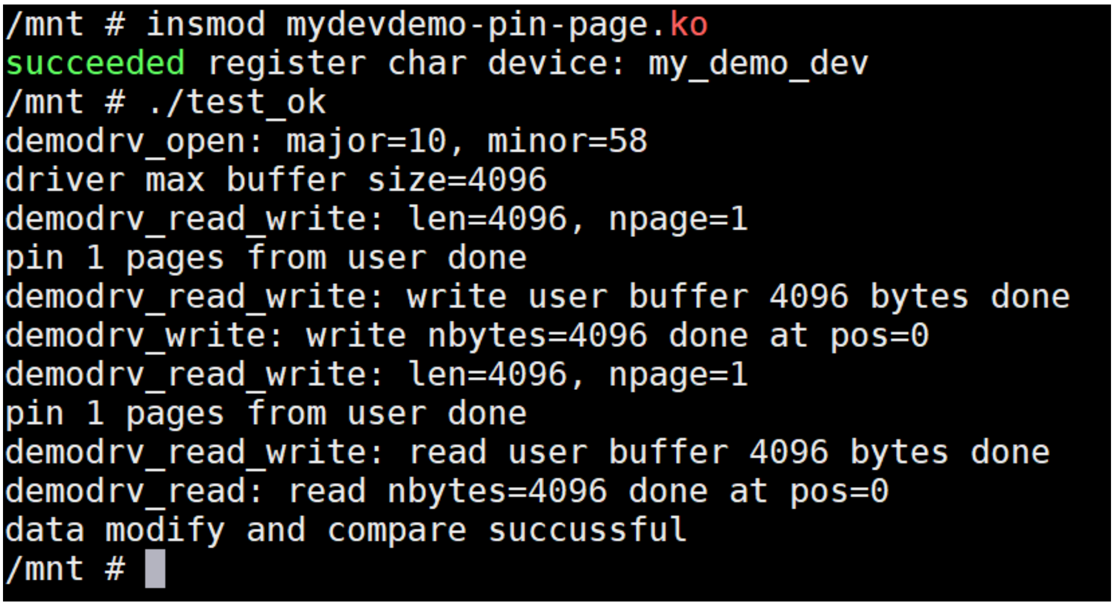
然后把 mydevdemo-pin-page.ko 和 test_ok 拷贝到 runninglinuxkernel_4.0/kmodules 目录下面。

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 mydevdemo-pin-page.ko
#./test_ok
```



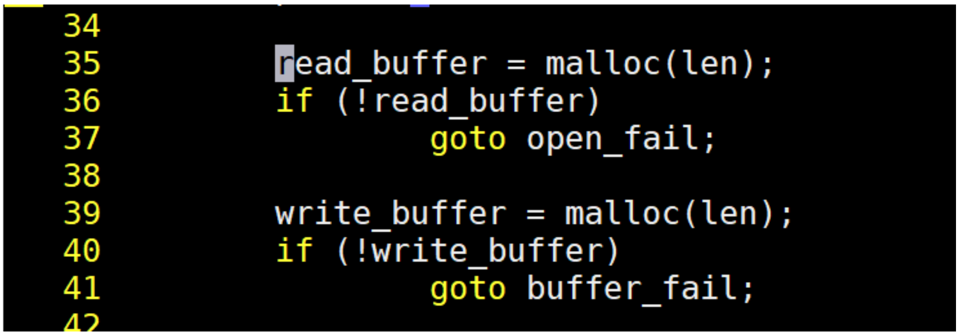
```
/mnt # insmod mydevdemo-pin-page.ko
succeeded register char device: my_demo_dev
/mnt # ./test_ok
demodrv_open: major=10, minor=58
driver max buffer size=4096
demodrv_read_write: len=4096, npage=1
pin 1 pages from user done
demodrv_read_write: write user buffer 4096 bytes done
demodrv_write: write nbytes=4096 done at pos=0
demodrv_read_write: len=4096, npage=1
pin 1 pages from user done
demodrv_read_write: read user buffer 4096 bytes done
demodrv_read: read nbytes=4096 done at pos=0
data modify and compare successful
/mnt #
```

可以看到测试成功了！

进阶思考

笨叔在实验里，设置一个进阶思考的问题。

代码里有一个 test_issue.c 文件，最大的不同就是使用 malloc 来分配 user buffer，而不是通过 mmap 来分配的匿名页面。

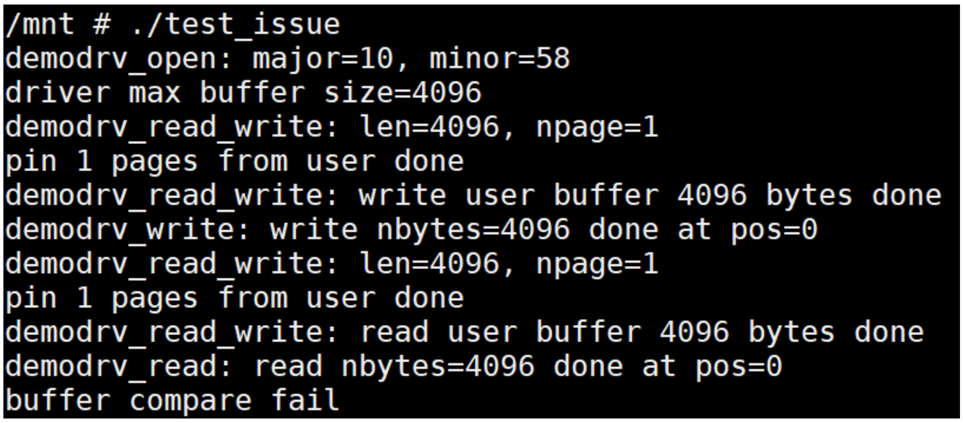


```

        read_buffer = malloc(len);
        if (!read_buffer)
                goto open_fail;

        write_buffer = malloc(len);
        if (!write_buffer)
                goto buffer_fail;
```

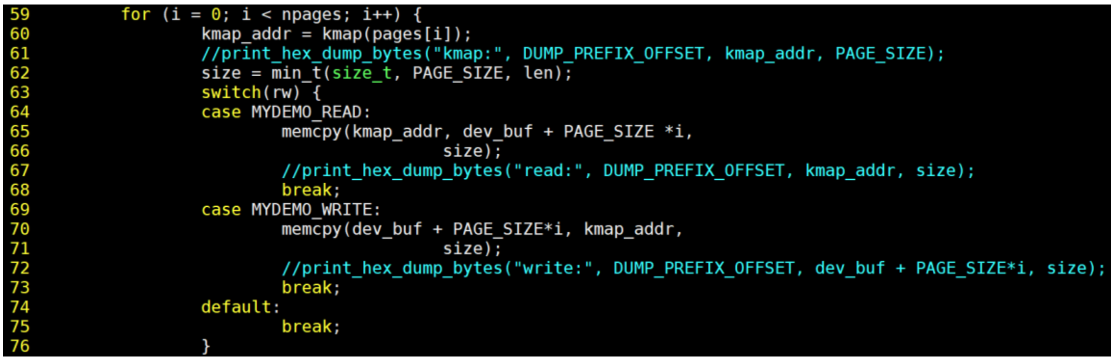
运行 test_issue 程序，得到如下结果：



```
/mnt # ./test_issue
demodrv_open: major=10, minor=58
driver max buffer size=4096
demodrv_read_write: len=4096, npage=1
pin 1 pages from user done
demodrv_read_write: write user buffer 4096 bytes done
demodrv_write: write nbytes=4096 done at pos=0
demodrv_read_write: len=4096, npage=1
pin 1 pages from user done
demodrv_read_write: read user buffer 4096 bytes done
demodrv_read: read nbytes=4096 done at pos=0
buffer compare fail
```

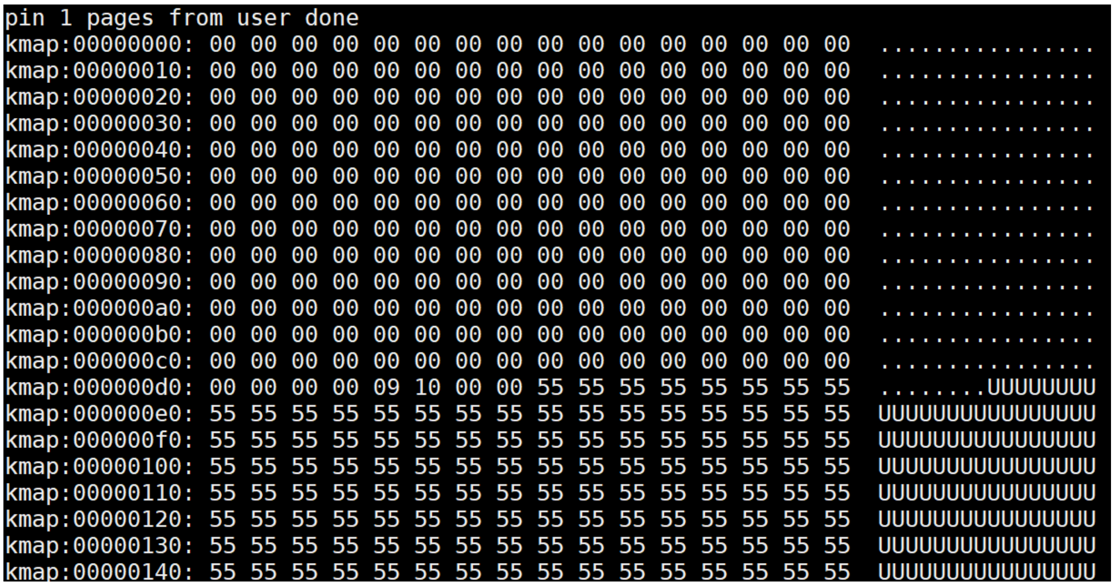
程序的最后结果是“buffer compare fail”，说明程序运行有问题，也就是 read buffer 和 write buffer 数据不对，究竟怎么回事呢？

遇到这种问题，我们最简单最粗暴也是最有效的调试办法就是把 buffer 打印出来看看。在内核里，可以使用 print_hex_dump_bytes()函数。比如在 demodrv_read_write()函数的第 61,67,72 行添加打印语句。



```
59          for (i = 0; i < npages; i++) {
60                  kmap_addr = kmap(pages[i]);
61                  //print_hex_dump_bytes("kmap:", DUMP_PREFIX_OFFSET, kmap_addr, PAGE_SIZE);
62                  size = min_t(size_t, PAGE_SIZE, len);
63                  switch(rw) {
64                  case MYDEMO_READ:
65                          memcpy(kmap_addr, dev_buf + PAGE_SIZE *i,
66                                          size);
67                          //print_hex_dump_bytes("read:", DUMP_PREFIX_OFFSET, kmap_addr, size);
68                          break;
69                  case MYDEMO_WRITE:
70                          memcpy(dev_buf + PAGE_SIZE*i, kmap_addr,
71                                          size);
72                          //print_hex_dump_bytes("write:", DUMP_PREFIX_OFFSET, dev_buf + PAGE_SIZE*i, size);
73                          break;
74                  default:
75                          break;
76                  }
```

打印结果如下：



```
pin 1 pages from user done
kmap:00000000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
kmap:00000010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
kmap:00000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
kmap:00000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
kmap:00000040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
kmap:00000050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
kmap:00000060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
kmap:00000070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
kmap:00000080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
kmap:00000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
kmap:000000a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
kmap:000000b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
kmap:000000c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
kmap:000000d0: 00 00 00 00 09 10 00 00 55 55 55 55 55 55 55 55  ........UUUUUUUU
kmap:000000e0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUU
kmap:000000f0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUU
kmap:00000100: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUU
kmap:00000110: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUU
kmap:00000120: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUU
kmap:00000130: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUU
kmap:00000140: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55  UUUUUUUUUUUUUUUU
```

发现 pin 的 page，开头的数据都是 0，而不是 0x55，这是为什么呢？理论上这个 page 应该全部都是 0x55 才对。

这是为什么呢？

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
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：