# 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 |

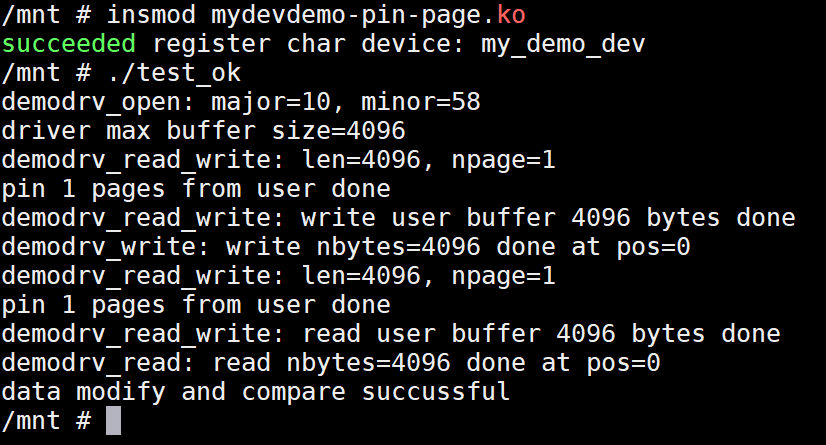
然后把mydevdemo-pin-page.ko和test\_ok拷贝到runninglinuxkernel\_4.0/kmodules目录下面。

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 |

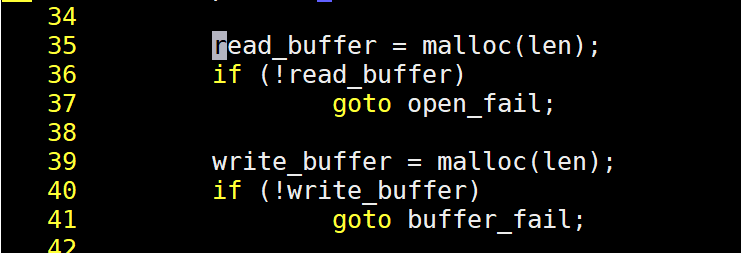


可以看到测试成功了！

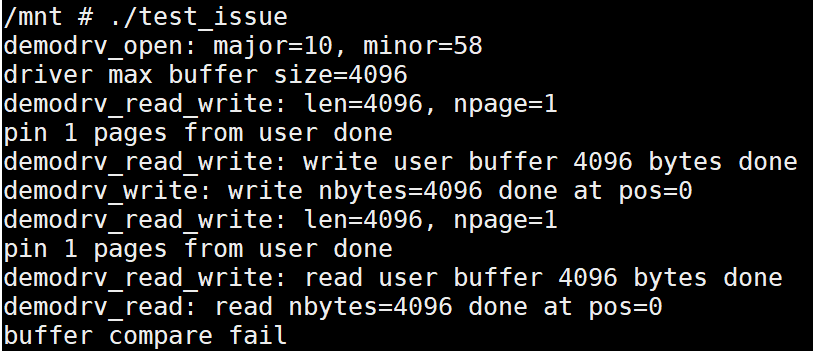
## 进阶思考

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。

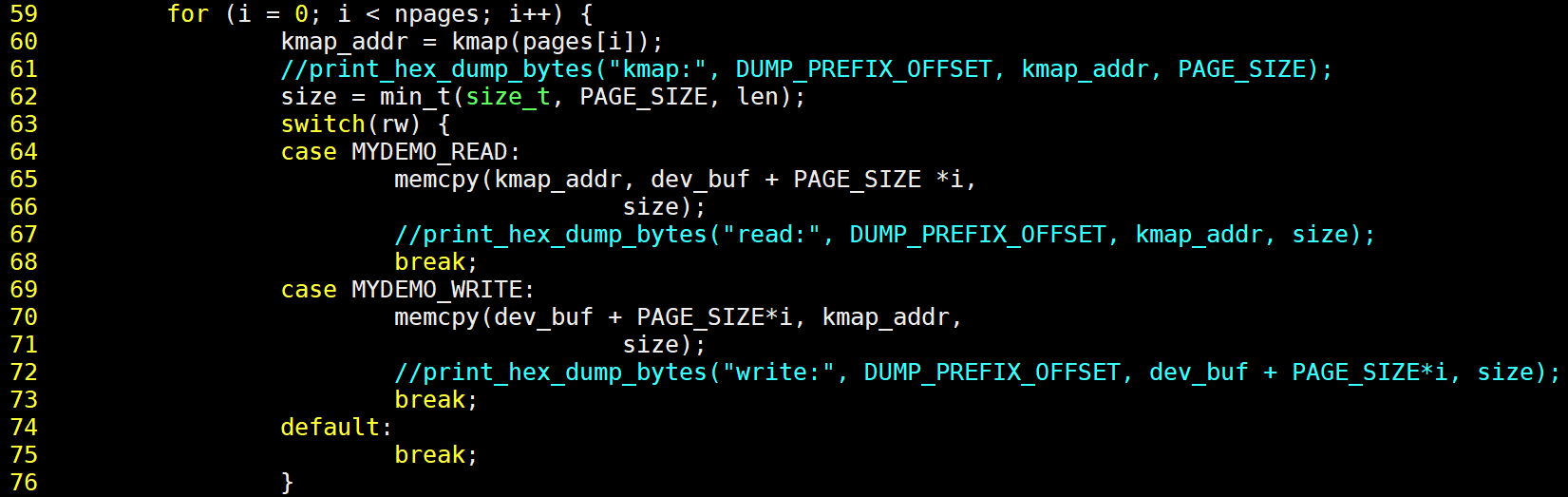


运行test\_issue程序，得到如下结果：

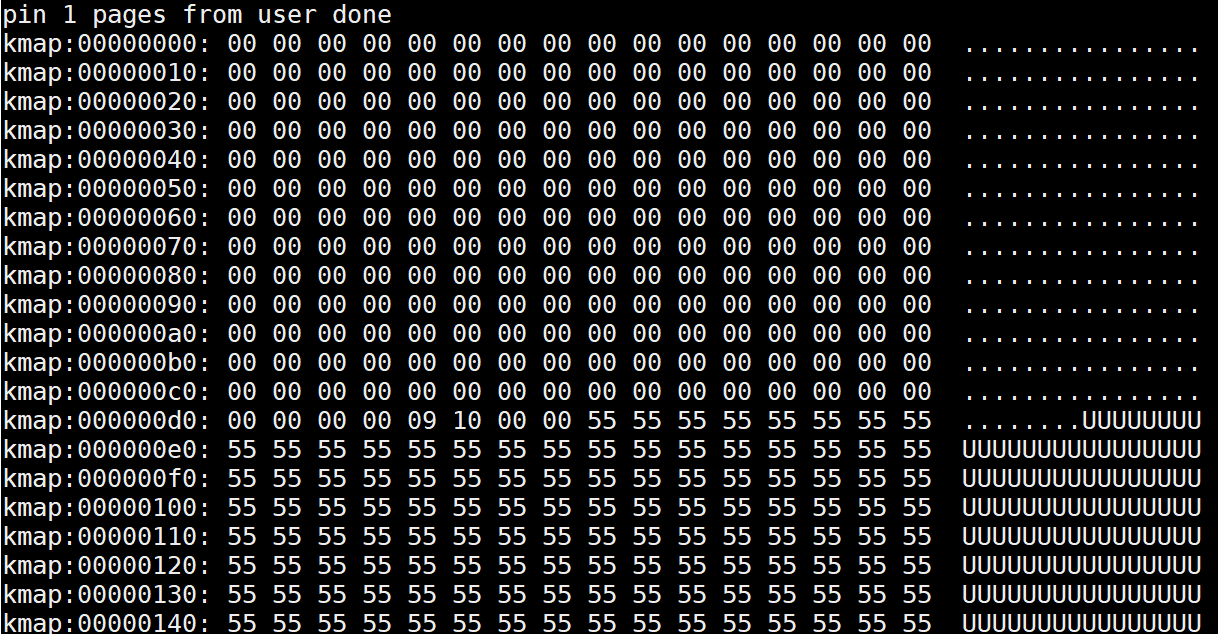


程序的最后结果是“buffer compare fail”，说明程序运行有问题，也就是read buffer和write buffer数据不对，究竟怎么回事呢？

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。



打印结果如下：



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。

请关注奔跑吧Linux社区微信公众号，里面有更多更有料更好玩的Linux干货：



奔跑吧Linux社区的旗舰篇视频节目，帮您打通Linux和职场任督二脉！





笨叔最新力作，全程约5小时高清，140多页ppt，8大实验，基于x86\_64的Centos 7.6和arm64，提供全套实验素材和环境。全面介绍kdump+crash在死机黑屏方面的实战应用，全部案例源自线上云服务器和嵌入式产品开发实际案例！

不能为客户和老板解决死机黑屏问题的都不是合格的linux运维或者研发攻城狮！





订阅视频请到淘宝店：<https://shop115683645.taobao.com>

