# Lab3\_为虚拟设备编写驱动

## 基本实验步骤

1. 进入对应的lab目录。

|  |
| --- |
| # export ARCH=arm  # export CROSS\_COMPILE=arm-linux-gnueabi-  # make BASEINCLUDE= 你的runninglinuxkernel\_4.0目录的绝对路径 |

这里BASEINCLUDE指定你当前runninglinuxkernel\_4.0的目录路径，每个人机器上路径可能不一样，请确保路径正确。

编译test测试程序。

|  |
| --- |
| # arm-linux-gnueabi-gcc test.c -o test --static |

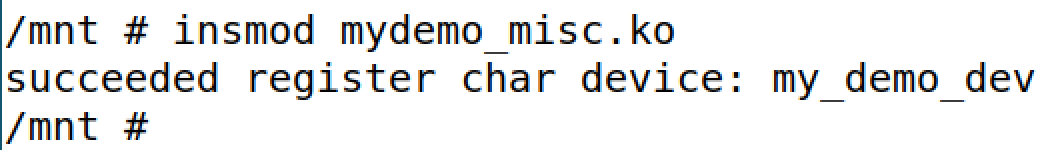
然后把ko内核模块和test程序拷贝到runninglinuxkernel\_4.0/kmodules目录下面。

运行如下脚本启动Qemu。

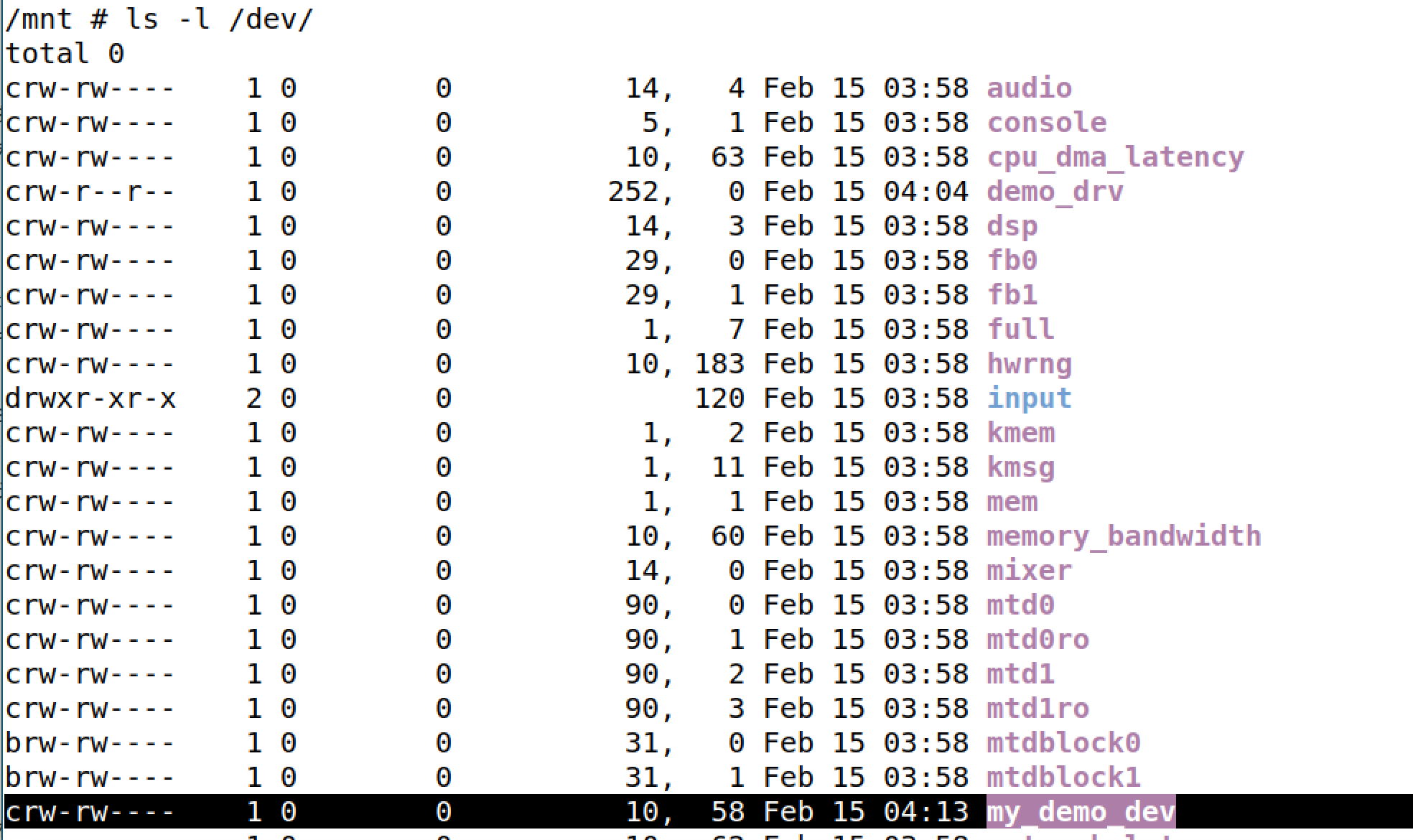
|  |
| --- |
| #cd runninglinuxkernel\_4.0  # sh run.sh arm32 #启动虚拟机，启动虚拟机之前，请先编译runninglinuxkernel\_4.0整个内核 |

在Qemu虚拟机:

|  |
| --- |
| #cd /mnt  # insmod xxxx.ko //加载刚才编译的内核模块 |

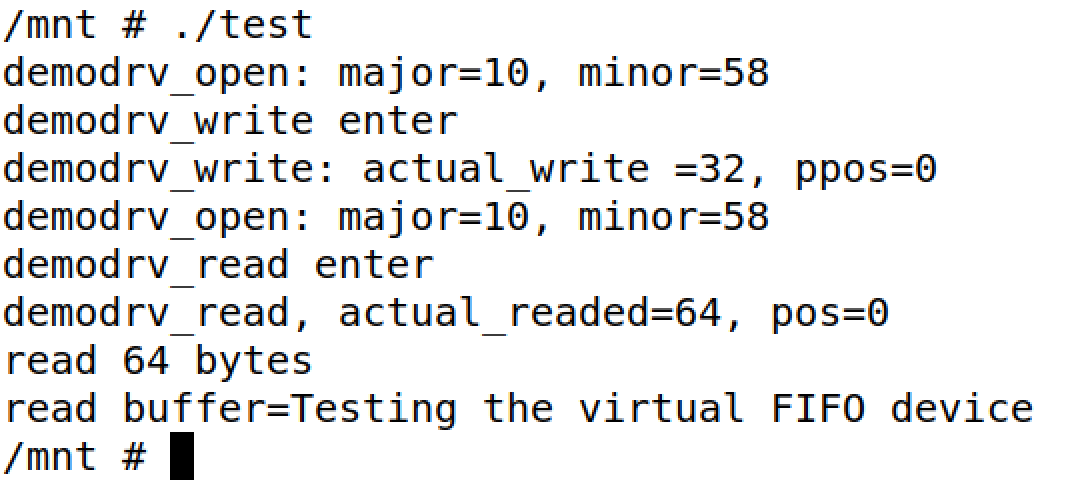


然后查看dev目录情况。



可以看到my\_demo\_dev设备自动创建了，主设备号是10，次设备号是58.

直接跑test程序看看是神马情况。



测试成功了！

## 进阶思考

1. 读者可以思考一下，为什么这里需要关闭设备后重新打开一次设备？如果不进行这样的操作，是否可以呢？
2. 这个lab，是最经典的字符设备读写操作的示例了，读者可以查阅现有的内核代码drivers目录，看看有哪些现成的字符设备驱动，学习一下，他们是如何实现读写函数的？

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>

