# lab2\_workqueue实验说明

在第5章的虚拟FIFO设备中，我们并没有考虑多个进程同时访问设备驱动的情况，请使用互斥锁对虚拟FIFO设备驱动程序进行并发保护

## 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。

编译test测试app。

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

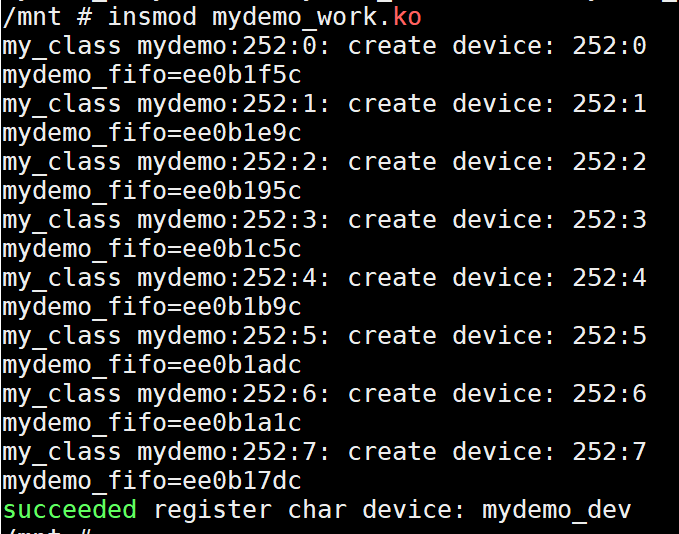
然后把\*.ko和test拷贝到runninglinuxkernel\_4.0/kmodules目录下面。

运行如下脚本启动Qemu。

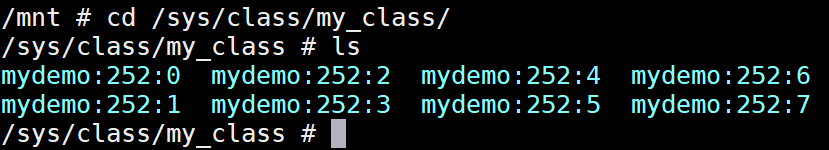
|  |
| --- |
| #cd runninglinuxkernel\_4.0  # sh run.sh arm32 #启动虚拟机 |

在Qemu虚拟机:

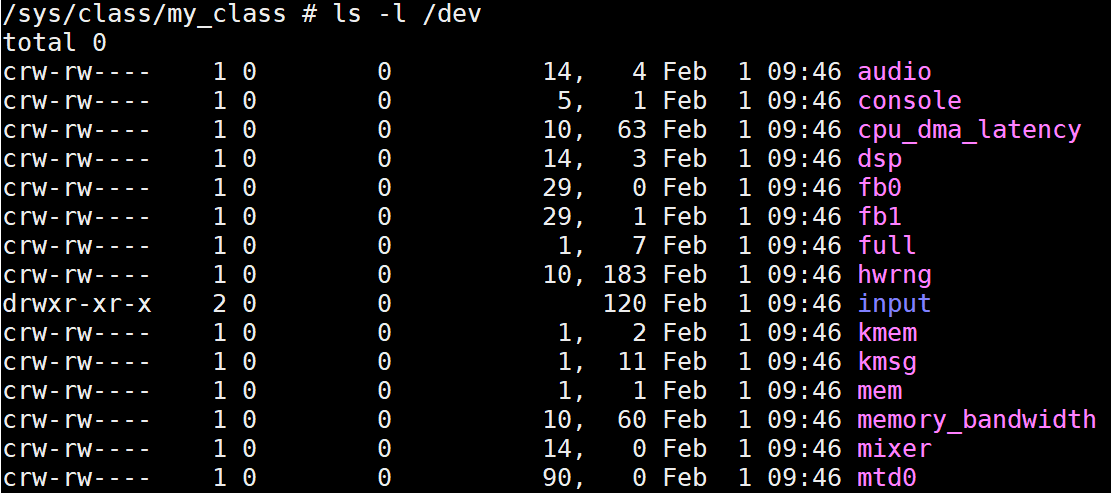
|  |
| --- |
| #cd /mnt  # insmod mydemo\_work.ko |



你会看到创建了8个设备。你可以到/sys/class/my\_class/目录下面看到这些设备。



我们可以看到创建了主设备号为252的设备。我们再来看一下/dev/目录。



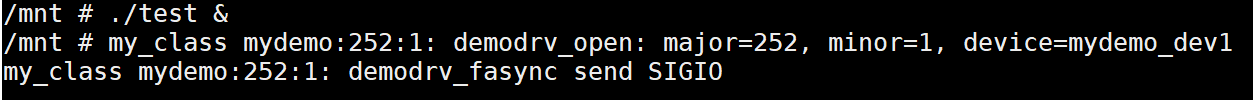
发现并没有主设备为252的设备。

所以我们需要手工创建一个设备用来test app。

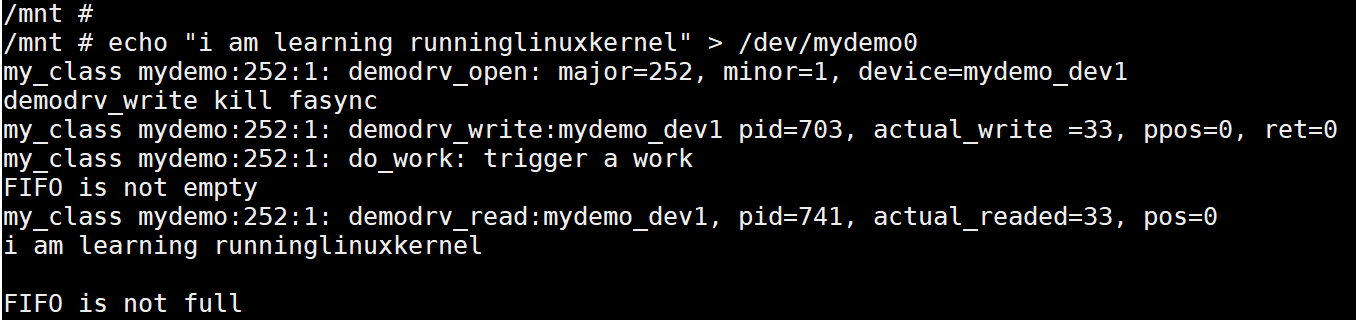
|  |
| --- |
| #mknod /dev/mydemo0 c 252 1 |

接下来跑我们的test程序：

|  |
| --- |
| # ./test & #这里让test程序在后台跑 |



然后使用echo命令来往/dev/mydemo0这个设备写入字符串。



可以看到从workqueue的回调函数打印的一句话“do\_work: trigger a work”。

## 进阶思考

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>

