# lab3\_rcu实验说明

本实验是一个rcu的测试实验。

编写一个简单的内核模块，创建一个读者内核线程和一个写者内核线程来模拟同步访问共享变量的情景。

## 基本实验步骤

1. 进入rlk\_lab/rlk\_basic/chapter\_9/lab3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。

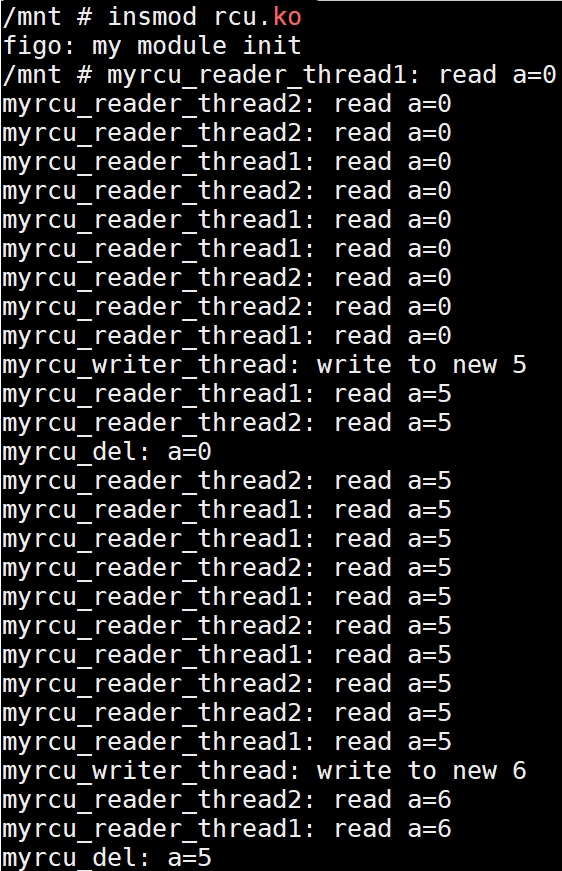
然后把rcu.ko拷贝到runninglinuxkernel\_4.0/kmodules目录下面。

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 rcu.ko |



可以看到：

该例子的目的是通过RCU机制保护my\_test\_init()分配的共享数据结构g\_ptr，另外创建了一个读者线程和一个写者线程来模拟同步场景。

对于读者线程myrcu\_reader\_thread：

❑ 通过rcu\_read\_lock()和rcu\_read\_unlock()来构建一个读者临界区。

❑ 调用rcu\_dereference()获取被保护数据g\_ptr指针的一个副本，即指针p，这时p和g\_ptr都指向旧的被保护数据。

❑ 读者线程每隔一段时间读取一次被保护数据。

对于写者线程myrcu\_writer\_thread：

❑ 分配一个新的保护数据new\_ptr，并修改相应数据。

❑ rcu\_assign\_pointer()让g\_ptr指向新数据。

❑ call\_rcu()注册一个回调函数，确保所有对旧数据的引用都执行完成之后，才调用回调函数来删除旧数据old\_data。

❑ 写者线程每隔一段时间修改被保护数据。

上述过程如图所示。



RCU时序图

在所有的读访问完成之后，内核可以释放旧数据，对于何时释放旧数据，内核提供了两个API函数：synchronize\_rcu()和call\_rcu()。

## 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>

