说明

根据要求建立了一个基于平台设备的字符设备驱动，设备名为memdev0。

整体被分为两个模块：一个设备模块：mem\_dev；一个设备驱动驱动模块：mem\_drv;

文件夹内有脚本mod.sh 来安装和卸载模块：

安装：以最高权限执行脚本并附参数i，即：sudo ./mod.sh i；

卸载：以最高权限执行脚本并附参数r，即：sudo ./mod.sh r；

用户可以通过读写设备文件来完成用户空间和内核空间的交互，同时用户读写后会进行计数操作。

测试可通过测试文件：a.out （由memdevapp.c编译得来）来执行设备文件的读写；对于读写计次则可以通过sysfs文件系统下的/sys/class/mymemdev/memdev0/count属性文件读取来完成；

具体实现:

当设备模块mem\_dev和设备驱动模块mem\_drv都装载到内核中时，会由**uevent**机制根据两者的名字进行匹配，匹配成功则执行驱动中的probe函数；

Probe函数主要完成初始化工作：申请字符设备；申请结构体用到的内存；初始化设备结构体mem\_dev内部包含的timer、tasklet、spinklock等变量；创建设备类及设备属性文件等；

这样通过文件系统**sysfs**就可以完成用户空间和内核空间的交互：

通过测试程序（a.out）进行写文件时，首先利用copy\_from\_user完成交互，然后由**tasklet**后半部机制处理记录读写计数的原子变量count：tasklet的实现函数中调用延时队列：**delaywork**，delaywork的实现函数中具体实现原子变量count的增加操作；

读文件时：首先调用copy\_to\_user完成交互，然后开启定时器**timer**，延后执行工作队列**workqueue**，在工作队列的具体处理函数中实现读写计数的自加；

Workqueue和delaywork最终的实现函数调用的是同样的函数，因为它们都是用于原子量count的自加操作；

问题：为什么明明调用了两次的自加，而每次读写一次才增加一次，而逻辑上应该是读写各加一次，即进行一次读写增加两次。

至于自旋锁**spinklock**则用于互斥资源访问，比如设备文件和读写计数文件的读写时都会持有自旋锁。