深入浅出Linux设备驱动之并发控制（2） PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/144/2021_2022__E6_B7_B1_E5_85_A5_E6_B5_85_E5_c103_144578.htm 该宏释放自旋锁lock，它与spin_trylock或spin_lock配对使用；除此之外，还有一组自旋锁使用于中断情况下的API。 下面进入对并发控制的实战。首先，在globalvar的驱动程序中，我们可以通过信号量来控制对int global_var的并发访问，下面给出源代码： #include #include #include #include #include MODULE_LICENSE("GPL").#define MAJOR_NUM 254static ssize_t globalvar_read(struct file *, char *, size_t, loff_t*).static ssize_t globalvar_write(struct file *, const char *, size_t, loff_t*).struct file_operations globalvar_fops ={ read: globalvar_read, write: globalvar_write,}.static int global_var = 0.static struct semaphore sem.static int __init globalvar_init(void){ int ret. ret = register_chrdev(MAJOR_NUM, "globalvar", amp.sem). } return ret.}static void __exit globalvar_exit(void){ int ret. ret = unregister_chrdev(MAJOR_NUM, "globalvar"). if (ret) { printk("globalvar unregister failure"). } else { printk("globalvar unregister success"). }}static ssize_t globalvar_read(struct file *filp, char *buf, size_t len, loff_t *off){ //获得信号量 if (down_interruptible(amp.global_var, sizeof(int))) { up(amp.sem). return sizeof(int).}ssize_t globalvar_write(struct file *filp, const char *buf, size_t len, loff_t *off){ //获得信号量 if (down_interruptible(amp.global_var, buf, sizeof(int))) { up(amp.sem). return

sizeof(int).}module_init(globalvar_init).module_exit(globalvar_exit). 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com