Linux内核中的同步和互斥分析报告 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/224/2021_2022_Linux_E5_86_85_E6_A0_c103_224250.htm 先看进程间的互斥。在linux内核中主要通过semaphore机制和spin_lock机制实现。主要的区别是在semaphore机制中，进不了临界区时会进行进程的切换，而spin_lock刚执行忙等（在SMP中）。先看内核中的semaphore机制。前提是对引用计数count增减的原子性操作。内核用atomic_t的数据结构和在它上面的一系列操作如atomic_add()、atomic_sub()等等实现。（定义在atomic.h中）semaphone机制主要通过up()和down()两个操作实现。semaphone的结构为: struct semaphore{atomic_t count.int sleepers.wait_queue_head_t wait. }.相应的down()函数为: static inline void down(struct semaphore*sem){/* 1 */sem->count--. //为原子操作if(sem->countstate = TASK_UNINTERRUPTIBLE.add_wait_queue_exclusive(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