Linux系统环境进程间通信：信号灯（4） 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_Linux_E7_B3_BB_E7_BB_c103_144776.htm 五、信号灯的限制 1、 一次系统调用semop可同时操作的信号灯数目SEMOPM，semop中的参数nsops如果超过了这个数目，将返回E2BIG错误。SEMOPM的大小特定与系统，redhat 8.0为32。 2、 信号灯的最大数目：SEMVMX，当设置信号灯值超过这个限制时，会返回ERANGE错误。在redhat 8.0中该值为32767。 3、 系统范围内信号灯集的最大数目SEMMNI以及系统范围内信号灯的最大数目SEMMNS。超过这两个限制将返回ENOSPC错误。redhat 8.0中该值为32000。 4、 每个信号灯集中的最大信号灯数目SEMMSL，redhat 8.0中为250。 SEMOPM以及SEMVMX是使用semop调用时应该注意的；SEMMNI以及SEMMNS是调用semget时应该注意的。SEMVMX同时也是semctl调用应该注意的。 六、竞争问题 第一个创建信号灯的进程同时也初始化信号灯，这样，系统调用semget包含了两个步骤：创建信号灯；初始化信号灯。由此可能导致一种竞争状态：第一个创建信号灯的进程在初始化信号灯时，第二个进程又调用semget，并且发现信号灯已经存在，此时，第二个进程必须具有判断是否有进程正在对信号灯进行初始化的能力。在参考文献[1]中，给出了绕过这种竞争状态的方法：当semget创建一个新的信号灯时，信号灯结构semid_ds的sem_otime成员初始化后的值为0。因此，第二个进程在成功调用semget后，可再次以IPC_STAT命令调用semctl，等待sem_otime变为非0值，此时可判断该信号灯已经初始化完毕。下图描述了竞

争状态产生及解决方法： 实际上，这种解决方法也是基于这样一个假定：第一个创建信号灯的进程必须调用semop，这样sem_otime才能变为非零值。另外，因为第一个进程可能不调用semop，或者semop操作需要很长时间，第二个进程可能无限期等待下去，或者等待很长时间。 七、信号灯应用实例 本实例有两个目的：1、获取各种信号灯信息；2、利用信号灯实现共享资源的申请和释放。并在程序中给出了详细注释。 #include #include #include #define SEM_PATH "/unix/my_sem"#define max_tries 3 int semid.main(){int flag1,flag2,key,i,init_ok,tmperrno.struct semid_ds sem_info.struct seminfo sem_info2.union semun arg. //union semun： 请参考附录2struct sembuf askfor_res, free_res.flag1=IPC_CREAT|IPC_EXCL|00666.flag2=IPC_CREAT|00666.key=ftok(SEM_PATH,a).//error handling for ftok here.init_ok=0.semid=semget(key,1,flag1).//create a semaphore set that only includes one semphore.if(semidsem_otime!=0){ i=max_tries. init_ok=1.}else sleep(1). }}if(!init_ok)// do some initializing, here we assume that the first process that creates the sem will // finish initialize the sem and run semop in max_tries*1 seconds. else it will not run // semop any more.{arg.val=1.if(semctl(semid,0,SETVAL,arg)==-1) perror("semctl setval error").} }else{perror("semget error, process exit"). exit(). }}else //semid>=0. do some initializing {arg.val=1.if(semctl(semid,0,SETVAL,arg)==-1)perror("semctl setval error").}//get some information about the semaphore and the limit of semaphore in

redhat8.0arg.buf=amp.sem_info2.if(semctl(semid,0,IPC_INFO,arg)==-1)perror("semctl IPC_INFO").printf("the number of entries in semaphore map is %d \n", arg.__buf->semmap).printf("max number of semaphore identifiers is %d \n", arg.__buf->semmni).printf("mas number of semaphores in system is %d \n", arg.__buf->semmns).printf("the number of undo structures system wide is %d \n", arg.__buf->semmnu).printf("max number of semaphores per semid is %d \n", arg.__buf->semmsl).printf("max number of ops per semop call is %d \n", arg.__buf->semopm).printf("max number of undo entries per process is %d \n", arg.__buf->semume).printf("the sizeof of struct sem_undo is %d \n", arg.__buf->semusz).printf("the maximum semaphore value is %d \n", arg.__buf->semvmx).//now ask for available resource: askfor_res.sem_num=0.askfor_res.sem_op=-1.askfor_res.sem_flg=SEM_UNDO. if(semop(semid,amp.free_res,1)==-1)//free the resource.if(errno==EIDRM)printf("the semaphore set was removed\n").//you can comment out the codes below to compile a different version: if(semctl(semid, 0, IPC_RMID)==-1)perror("semctl IPC_RMID").else printf("remove sem ok\n").}