


第6章 进程间通信

——信号量协调进程同步

主 讲：黄 茹

semop		
功能	完成信号量集的一组操作	
头文件	/usr/include/sys/sem.h	
函数原型	int semop(int semid, struct sembuf semoparr[ ], size_t nops);	
参数	semid	信号量集ID
	semoparr	存放一组信号量操作的数组
	nops	数组中操作的数量
返回值	0	成功
	-1	失败

说明

- sembuf的结构

```
struct sembuf
{
    unsigned short  sem_num;  //信号量的序号
             short  sem_op;     //正数、0或负数
             short  sem_flg;    //IPC_NOWAIT,SEM_UNDO
}
```

- 含义：以sem_flag的方式对第sem_num个信号量做操作sem_op

对信号量的操作

sem_op
- >0 进程要释放的资源数量
- <0 进程要申请的资源数量
  - 资源够用：修改信号量的值
  - 资源不够：
    - 指定了IPC_NOWAIT：出错返回
    - 未指定IPC_NOWAIT：进程挂起
- =0 进程希望等待到该信号量值变成0
  - 信号量为0：立即返回
  - 信号量非0：
    - 指定了IPC_NOWAIT：出错返回
    - 未指定IPC_NOWAIT：进程挂起

SEM_UNDO标识

- 进程exit时若占用了经由信号量分配的资源，会影响资源的使用
- 设置该标志，由内核记录该资源分配给各进程的情况
- 进程终止时，由内核负责调整该资源的信号量值

示例程序：写共享内存加锁

```
lockforwrite(int semid)  {
    struct sembuf action[2];
    action[0].sem_num=0;
    action[0].sem_flg=SEM_UNDO;
    action[0].sem_op=0;
    action[1].sem_num=1;
    action[1].sem_flg=SEM_UNDO;
    action[1].sem_op=-1;
    if(semop(semid,action,2)==-1)   {
        perror("semop");   exit(1);
    }
}
```

示例程序：写共享内存解锁

```
unlockafterwrite(int semid)     {
    struct sembuf action[1];
    action[0].sem_num=1;
    action[0].sem_flg=SEM_UNDO;
    action[0].sem_op=+1;
    if(semop(semid,action,1)==-1)    {
            perror("semop");        exit(1);
    }
}
```

信号量机制总结

信号量机制是为了协调多个进程对资源的使用，semop操作对应于操作系统中的PV原语。但与操作系统中的信号量有所不同：

1. 信号量机制中最小单位是信号量集，并非单个整型数，信号量集中的信号量数目在创建该集合时指定。

2. 创建信号量集与对其赋初值这两个操作是分开的，并非合在一起的原子操作，这是一个致命的弱点。

3. 有些程序在终止时并没有释放已经分配给它的信号量集，SEM_UNDO标识就是用来处理这些情况的。

谢谢大家！