五级项目 5：信号量的申请和释放

1 学时

- 1 学时

2 实验目的

- 理解、掌握、应用 Linux 下 System V 信号量工具的使用。

3 实验内容

- 创建包含 3 个信号量的信号量集，命令行参数个数为 1 时对第 0 个信号量赋值为 1，然后对 0 号信号量执行一次申请和释放过程，执行前后输出 0 号信号量的值及时间；命令行参数个数为 2 且第二个参数为“del”时删除信号量。

4 实验原理

信号量（semaphore）是一种比较特殊的 IPC，它是一个计数器（counter）。一般情况下，多个进程在访问共享对象时使用信号量实现同步操作，如经典的生产者/消费者问题。

System V IPC 操作信号量的相关系统调用有三个：semget、semctl、semop。

semget 用于创建新的信号量集或打开已存在的信号量集。

semctl 用于信号量集的控制操作。如获取信号量集的内核信息，删除信号量集等。

semop 用于信号量操作。System V 信号量通过向 semop 传递不同的参数来完成 wait 和 signal 操作。如果信号量的值大于 0，wait 操作将信号量的值减 1；如果信号量的值等于 1，执行 wait 操作的进程被阻塞。

表 系统调用 semget

项目	描述
头文件	#include <sys/types.h> #include <sys/ipc.h>
原型	int semget(key_t key, int nsems, int semflg);
功能	创建信号量
参数	key：信号量的键值 nsems：信号量的个数 semflg：创建标志，可以是 IPC_CREAT 或 IPC_CREAT | IPC_EXCL
返回值	成功返回信号量 ID；失败返回-1，并设置 errno

表 系统调用 semctl

项目	描述
头文件	#include <sys/types.h> #include <sys/ipc.h> #include <sys/sem.h>
原型	int semctl(int semid, int semnum, int cmd, ...);
功能	信号量控制函数

参数	semid：信号量 ID
	cmd：控制命令
	semnum：信号量集中信号量的序号
	根据 cmd 命令不同可能有第四个参数，如果有则第四个参数是 union 类型
	union semun {
	int val; /* cmd 为 SETVAL */
	/* 获取或设置内核新的的缓冲，cmd 为 IPC_STAT，IPC_SET */
	struct semid_ds *buf;
	unsigned short *array; /* cmd 为 GETALL，SETALL */
	struct seminfo *_ _buf; /*cmd 为 IPC_INFO (用于 Linux) */
	};

返回值	成功返回非负值；失败返回-1，并置错误代码

semctl 接收可变参数，根据 cmd 的值确定参数的个数，可以为 3 或 4。semid 为 semget 返回的标识符。semnum 指定信号量集中信号量的序号，从 0 开始。cmd 取值如表 10.4 所示。

表 cmd 的取值

项目	说明
IPC_STAT	获取信号量集的内核信息
IPC_SET	设置信号量集的内核信息
IPC_RMID	从系统中删除该信号量集合。这种删除是立即的。仍在使用此信号量的其他进程在它们下次意图对此信号量进行操作时，将出错返回 EIDRM
GETVAL	返回成员 semnum 的 semval 值
SETVAL	设置成员 semnum 的 semval 值
GETPID	返回成员 semnum 的 sempid 值
GETNCNT	返回成员 semnum 的 semncnt 值
GETZCNT	返回成员 semnum 的 semzcnt 值
GETALL	取该集合中所有信号量的值，并将它们存放在由 arg.array 指向的数组中
SETALL	按 arg.array 指向的数组中的值设置该集合中所有信号量的值

semctl 出错时返回-1。成功时返回值如表 所示。

表 系统调用 semctl 返回值

项目	说明
GETNCNT	semncnt 的值
GETPID	sempid 的值
GETVAL	semval 的值
GETZCNT	semzcnt 的值
IPC_INFO	返回内核内部关于信号集信息的最大可用入口索引
SEM_INFO	与 IPC_INFO 相同
SEM_STAT	返回信号集标识符

表 系统调用 semop

项目	描述
头文件	#include <sys/types.h> #include <sys/ipc.h> #include <sys/sem.h>
原型	int semop(int semid, struct sembuf *sops, unsigned nsops);
功能	信号量操作
参数	semid：信号量 ID sops：指向在集合上执行操作的数组 nsops：在 sembuf 数组上操作的个数
返回值	成功返回 0；失败返回-1

sops 为一个 sembuf 结构变量数组，nsops 为数组中元素的个数。

```
struct sembuf{
    unsigned short  sem_num;    /*信号量序号，从 0 开始*/
    short           sem_op;     /*在信号量上的操作（可以为正、零、负）*/
    short           sem_flg;    /*IPC_NOWAIT 或 SEM_UNDO*/
}
```

5 预习要求和技术准备工作

- 掌握 Linux 基本操作
- 掌握 C 语言开发工具的使用
- 掌握信号量的申请和释放的方法

6 实验环境

- PC 机
- 在 Windows 环境中的 VMware 虚拟机上运行 Ubuntu 操作系统或者独立的 Ubuntu 操作系统
- 基于 Linux 的 vi 编辑器和 gcc 编译器

7 实验设计及操作步骤

7.1 以 root 身份登录系统，在/home 目录中创建目录 exp55

```
cd  /home
mkdir  exp55
```

7.2 进入刚创建的目录

```
cd  exp55
```

7.3 使用 vi 编辑文件，文件名是 sem.c。

7.4 编写程序，实现要求的功能

★编程思路

main(int argc,char *argv[])

{

① 判断命令行参数，区分功能，如果 argv[1]为 del 则为删除信号量，若没有 argv[1]则为信号量操作

② 通过 ftok 和 semget 创建 3 个信号量并得到信号量集 id

③ 如果是信号量操作

将信号量 0 设置初值为 1，如果执行失败则删除信号量集。

获取信号量 0 的值并显示

对信号量 0 执行 p 操作，获取信号量 0 的值并显示

等待 5 秒后，对信号量 0 执行 v 操作，再获取值并显示。

④ 如果是删除信号量：

利用 semctl 删除信号量集，并提示被删除的信号量集的 id

}

7.5 编译可执行文件。

```
#gcc sem.c -o sem
```

7.6 运行程序

```
#./sem
```

执行了对 0 号信号量的申请和释放过程。

```
cst@ubuntu:~/yanshou/55$ ./sem
semid 0
after semctl setval sem[0].val=[1]
2021 年  08 月  23 日  星期一  14:58:08 CST
P operate begin
P operate end
after P sem[0].val=[0]
2021 年  08 月  23 日  星期一  14:58:08 CST
waiting for 5 seconds
V operate begin
V operate end
after V sem[0].val=[1]
2021 年  08 月  23 日  星期一  14:58:13 CST
```

使用#ipcs -s 查看信号量资源情况

```
cst@ubuntu:~/yanshou/55$ ipcs -s
--------- 信号量数组 -----------
键          semid      拥有者   权限       nsems
0x66050004 0          cst        600        3
```

#./sem del

```
cst@ubuntu:~/yanshou/55$ ./sem del
semid 0
semaphore 0 deleted!
cst@ubuntu:~/yanshou/55$ ipcs -s

--------- 信号量数组 -----------
键          semid      拥有者    权限       nsems
```

8 实验报告提交要求：

将实验操作每个步骤中的命令、源程序以及截图写入实验报告，实验报告命名为“学号姓名-实验 55.doc”，交给指定人员。

9 项目思考

调研思考信号量在内核中的实现原理。