实验三 进程同步

实验环境

Linux 平台

实验时间

4 小时

实验目的

利用信号量机制来实现访问同一共享内存的进程间同步。

实验内容

实现多个进程通过同一内存空间进行通信，要求使用操作系统提供的信号量机制同步这些进程的操作，保证数据一致性。具体要求：请在实验中对比实现使用及未使用信号量的实验效果，并说明是否保证了数据的一致性；可使用多线程实现。

背景知识

一、Linux 系统提供的信号量函数

主要由 semget、semop、semctl 三个函数组成。下面列出了这三个函数的函数原型及具体说明。所需头文件为：

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
```

1 semget 函数

它的作用是创建一个新信号量或取得一个已有信号量，原型为：

```
int semget(key_t   key, int num_sems, int sem_flags);
```

semget 函数成功返回一个相应信号标识符（非零），失败返回-1.

第一个参数 key 是整数值（唯一非零），不相关的进程可以通过它访问一个信号量，它代表程序可能要使用的某个资源，程序对所有信号量的访问都是间接的，程序先通过调用 semget 函数并提供一个键，再由系统生成一个相应的信号标识符（semget 函数的返回值），只有 semget 函数才直接使用信号量键，所有其他的信号量函数使用由 semget 函数返回的信号量标识符。如果多个程序使用相同的 key 值，key 将负责协调工作。
第二个参数 num_sems 指定需要的信号量数目，该参数只在创建信号量集时有效，它的值几乎总是 1。
第三个参数 sem_flags 是一组标志，当想要当信号量不存在时创建一个新的信号量，可以和值 IPC_CREAT 做按位或操作。设置了 IPC_CREAT 标志后，即使给出的键是一个已有信号量的键，也不会产生错误。而 IPC_CREAT|IPC_EXCL 则可以创建一个新的，唯一的信号量，如果信号量已存在，返回一个错误。

2 semop 函数
它的作用是改变信号量的值，即对信号量集标识符为 semid 中的一个或多个信号量的 P 操作或 V 操作。原型为：
int semop(int semid, struct sembuf *sops, unsigned nsops)
函数传入值：
semid：信号量集标识符
sops：指向进行操作的信号量集结构体数组的首地址，此结构的具体说明如下：
```
struct sembuf {
    short semnum; /*信号量集合中的信号量编号，0 代表第 1 个信号量*/
    short val;/*若 val>0 进行 V 操作信号量值加 val，表示进程释放控制的资源 */
            /*若 val<0 进行 P 操作信号量值减 val，若(semval-val)<0（semval 为该信号量值），
        则调用进程阻塞，直到资源可用；若设置 IPC_NOWAIT 不会睡眠，进程直接返回
        EAGAIN 错误*/
             /*若 val==0 时阻塞等待信号量为 0，调用进程进入睡眠状态，直到信号值为 0；
        若设置 IPC_NOWAIT，进程不会睡眠，直接返回 EAGAIN 错误*/
    short flag;  /*0  设置信号量的默认操作*/
                 /*IPC_NOWAIT 设置信号量操作不等待*/
                 /*SEM_UNDO  选项会让内核记录一个与调用进程相关的 UNDO 记录，如果该
                 进程崩溃，则根据这个进程的 UNDO 记录自动恢复相应信号量的计数值*/
  };
```
nsops：进行操作信号量的个数，即 sops 结构变量的个数，需大于或等于 1。最常见设置此值等于 1，只完成对一个信号量的操作
函数返回值
成功：返回信号量集的标识符
出错：-1，错误原因存于 error 中

sops 为指向 sembuf 数组，定义所要进行的操作序列。下面是信号量操作举例。
```
struct sembuf sem_get={0,-1,IPC_NOWAIT}; /*将信号量对象中序号为 0 的信号量减 1*/
struct sembuf sem_get={0,1,IPC_NOWAIT};  /*将信号量对象中序号为 0 的信号量加 1*/
struct sembuf sem_get={0,0,0};           /*进程被阻塞，直到对应的信号量值为 0*/
```

flag 一般为 0，若 flag 包含 IPC_NOWAIT，则该操作为非阻塞操作。若 flag 包含 SEM_UNDO，

则当进程退出的时候会还原该进程的信号量操作，这个标志在某些情况下是很有用的，比如某进程做了 P 操作得到资源，但还没来得及做 V 操作时就异常退出了，此时，其他进程就只能都阻塞在 P 操作上，于是造成了死锁。若采取 SEM_UNDO 标志，就可以避免因为进程异常退出而造成的死锁。

3 semctl 函数

该函数用来得到一个信号量集标识符或创建一个信号量集对象并返回信号量集标识符，

函数原型

int semctl(int semid, int semnum, int cmd, union semun arg)

函数传入值

semid 信号量集标识符

semnum 信号量集数组上的下标，表示某一个信号量

cmd 见下文表 1

arg

```
union semun {
   short val;          /*SETVAL 用的值*/
   struct semid_ds* buf; /*IPC_STAT、IPC_SET 用的 semid_ds 结构*/
   unsigned short* array; /*SETALL、GETALL 用的数组值*/
   struct seminfo *buf;  /*为控制 IPC_INFO 提供的缓存*/
  } arg;
```

函数返回值

成功：大于或等于 0，具体说明请参照表 1

出错：-1，错误原因存于 error 中

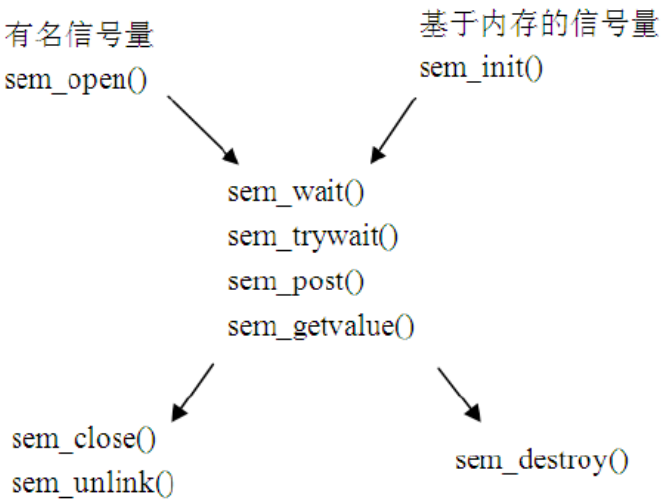
表 1 semctl 函数 cmd 形参说明表

命 令	解 释
IPC_STAT	从信号量集上检索 semid_ds 结构，并存到 semun 联合体参数的成员 buf 的地址中
IPC_SET	设置一个信号量集合的 semid_ds 结构中 ipc_perm 域的值，并从 semun 的 buf 中取出值
IPC_RMID	从内核中删除信号量集合
GETALL	从信号量集合中获得所有信号量的值，并把其整数值存到 semun 联合体成员的一个指针数组中
GETNCNT	返回当前等待资源的进程个数
GETPID	返回最后一个执行系统调用 semop()进程的 PID
GETVAL	返回信号量集合内单个信号量的值
GETZCNT	返回当前等待 100%资源利用的进程个数
SETALL	与 GETALL 正好相反
SETVAL	用联合体中 val 成员的值设置信号量集合中单个信号量的值

二、POSIX 提供的信号量

POSIX 提供两种信号量，**有名信号量**和**无名信号量**，有名信号量一般是用在进程间同步，无名信号量一般用在线程间同步。两种信号量的操作流程，主要在于两种信号量初始化和销毁的方式

不同。



以下函数所需头文件:

#include <semaphore.h>

注意：编译信号量操作函数时，需要加上参数-lpthread。

信号量数据类型为：sem_t。

1. 有名信号量操作

创建有名信号量：

创建或者打开一个信号量，需要使用 sem_open()函数，函数原形如下：

sem_t sem_open(const char * name, int oflag, mode_t mode, unsigned int value)

返回值 sem_t 是一个结构，如果函数调用成功，则返回指向这个结构的指针，里面装着当前信号量的资源数。

参数 name，就是信号量的名字，两个不同的进程通过同一个名字来进行信号量的传递。

参数 oflag，当是 O_CREAT 时，如果 name 给出的信号量不存在，那么创建，此时必须给出 mode 和 vaule。

参数 mode，用来指定信号量的权限。

参数 vaule，则是信号量的初始值。

关闭有名信号量：

关闭有名信号量所使用的函数是 sem_close(sem_t *sem)

这个函数只有一个参数，就是指信号量的名字。

信号量操作：

前面已经说过，在使用信号量时，有两个非常重要的操作

P 操作：使用的函数是 sem_wait(sem_t *sem)

如果信号量的值大于零，sem_wait 函数将信号量减一，并且立即返回。如果信号量的值小于零，那么该进程会被阻塞在原地。

V 操作：使用的函数是 sem_post(sem_t *sem)

当一个进程使用完某个信号量时，他应该调用 sem_post 函数来告诉系统收回资源。

sem_post 函数和 sem_wait 函数的功能刚好相反，会把指定的信号量加一

删除有名信号量：

当使用完有名信号后，需要调用函数 sem_unlink 来释放资源。

函数原形：int sem_unlink(const char *name)

2. 无名信号量操作

1）初始化信号量

int sem_init(sem_t *sem, int pshared, unsigned int value);

功能：

创建一个信号量并初始化它的值。一个无名信号量在被使用前必须先初始化。

参数：

sem：信号量的地址。

pshared：等于 0，信号量在线程间共享（常用）；不等于 0，信号量在进程间共享。

value：信号量的初始值。

返回值：成功：0 ； 失败：-1

2）信号量 P 操作（减 1）

int sem_wait(sem_t *sem);

功能：

将信号量的值减 1。操作前，先检查信号量（sem）的值是否为 0，若信号量为 0，此函数会阻塞，直到信号量大于 0 时才进行减 1 操作。

参数：sem：信号量的地址。

返回值：成功：0；失败：-1

int sem_trywait(sem_t *sem);

以非阻塞的方式来对信号量进行减 1 操作。若操作前，信号量的值等于 0，则对信号量的操作失败，函数立即返回。

3）信号量 V 操作（加 1）

int sem_post(sem_t *sem);

功能：将信号量的值加 1 并发出信号唤醒等待线程（sem_wait()）。

参数：sem：信号量的地址。

返回值：成功：0；失败：-1

4）获取信号量的值

int sem_getvalue(sem_t *sem, int *sval);

功能：获取 sem 标识的信号量的值，保存在 sval 中。

参数：

sem：信号量地址。

sval：保存信号量值的地址。

返回值：成功：0；失败：-1

5）销毁信号量

int sem_destroy(sem_t *sem);

功能：删除 sem 标识的信号量。

参数：sem：信号量地址。

返回值：成功：0；失败：-1

实验步骤

实验心得