

華 中 科 技 大 學

课 程 实 验 报 告

课程名称：操作系统

实验名称：线程的同步与互斥

院 系：计算机科学与技术

专业班级：cs1307

学 号：U201314970

姓 名：程校猛

指导教师：

2015 年 12 月 6 日

1、实验目的

- 1.掌握 Linux 系统用户界面中键盘命令的使用。
- 2.学会一种 linux 下的编程环境。
- 3.掌握 linux 下线程的概念。
- 4.通过信号灯实现线程的同步与互。

2、实验内容

实现两个线程，共享公共变量 a，线程 1 负责计算（+1），线程 2 负责打印。

3、实验步骤

1.信号灯的创建

使用系统调用 `semget()` 创建一个新的信号量集，或者存取一个已经存在的信号量集。因为要实现同步关系，所以需要两个信号灯。

2.定义 P、V 操作

p 操作使信号量减一，v 操作使信号量加一。

3.给信号量赋初值

系统调用 `semctl()`，将一个信号灯赋值为 1，另一个赋值为 0。

4.创建线程

4、程序源码

```
#include <stdio.h>
#include <pthread.h>
#include <sys/types.h>
#include <sys/sem.h>

void P(int semid,int index);
void V(int semid,int index);
void *subp1();
void *subp2();
int semid; //信号量
int s;    //数值

union semun{
    int val;
    struct semid_ds *buf;
    unsigned short *array;
}semopts; //定义联合体，用于初始化信号量

main(){
    s=0;
    pthread_t p1,p2;
    semid=semget(1234,2,IPC_CREAT|0666); //创建信号灯
    semopts.val=1; //第一个信号灯的初值为 1
    semctl(semid,0,SETVAL,semopts);
    semopts.val=0; //第二个信号灯的初值为 0
    semctl(semid,1,SETVAL,semopts);
    pthread_create(&p1,NULL,(void *)subp1,NULL);
    pthread_create(&p2,NULL,(void *)subp2,NULL); //创建两个线程 subp1 , subp2
    puts("init S=0");
    pthread_join(p1,NULL);
    pthread_join(p2,NULL);
    //等待两个线程运行结束
    semctl(semid,0,IPC_RMID);
    semctl(semid,1,IPC_RMID); //删除信号灯
}
```

```
void *subp1(){ //定义线程 1 的执行内容
```

```
    int i;
```

```
    for(i=0;i<10;i++){
```

```
        P(semid,0);
```

```
        //puts("subp1");
```

```
        s++;          //进行加一操作
```

```
        puts("s++");
```

```
        V(semid,1);
```

```
    }
```

```
    return;
```

```
}
```

```
void *subp2(){ //定义线程 2 的执行内容
```

```
    int j;
```

```
    for(j=0;j<10;j++){
```

```
        P(semid,1);
```

```
        //puts("subp2");
```

```
        printf("S=%d\n",s); //进行打印操作
```

```
        V(semid,0);
```

```
    }
```

```
    return;
```

```
}
```

```
//P 操作
```

```
void P(int semid,int index){
```

```
    struct sembuf sem;
```

```
    sem.sem_num=index; //要处理的信号量的下标
```

```
    sem.sem_op=-1;     //要执行的操作
```

```
    sem.sem_flg=SEM_UNDO;//操作标志
```

```
    semop(semid,&sem,1); //参数依次为 semget 函数返回的信号量
```

```
                        //标志符，指向结构数组的指针，操作次数
```

```
    return;
```

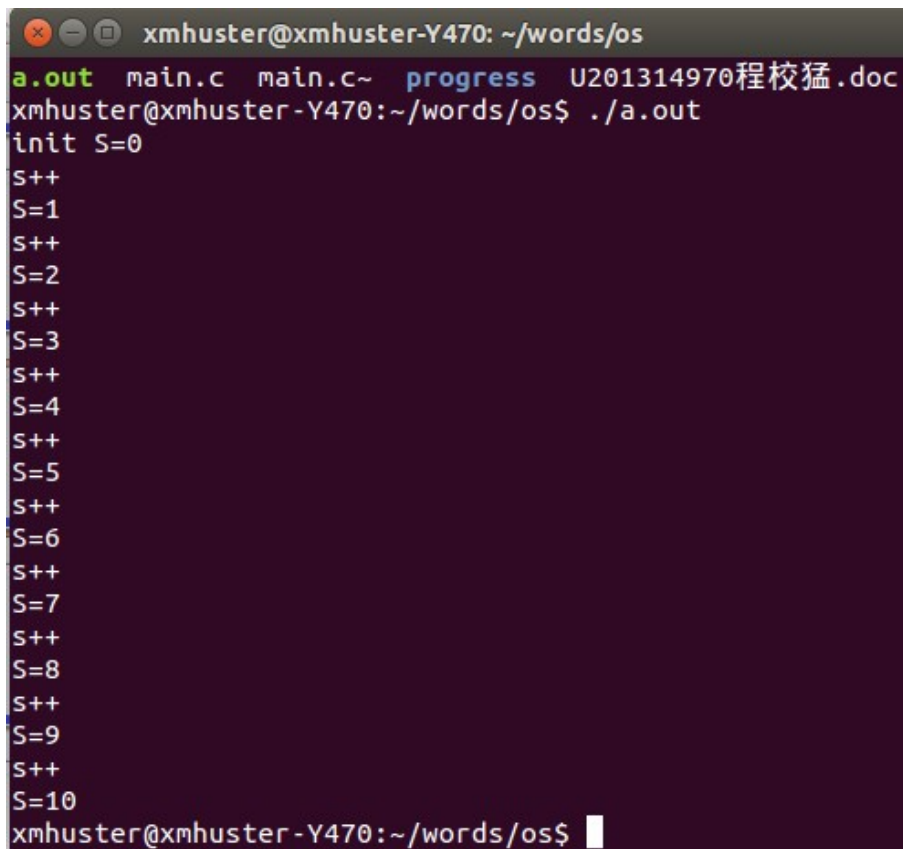
```
}
```

```
//V 操作
```

```
void V(int semid,int index){
```

```
struct sembuf sem;
sem.sem_num=index;
sem.sem_op=1;
sem.sem_flg=SEM_UNDO; //NO_WAIT 和 SEM_UNDO 的区别！
semop(semid,&sem,1);
return;
}
```

5、运行结果

A terminal window with a dark background and light-colored text. The window title is 'xmhuster@xmhuster-Y470: ~/words/os'. The prompt is 'xmhuster@xmhuster-Y470:~/words/os\$'. The user has entered './a.out'. The output shows a sequence of 'S++' and 'S=' operations, alternating between two threads. The value of 'S' starts at 0 and increases to 10. The output is as follows:

```
xmhuster@xmhuster-Y470: ~/words/os
a.out main.c main.c~ progress U201314970程校猛.doc
xmhuster@xmhuster-Y470:~/words/os$ ./a.out
init S=0
S++
S=1
S++
S=2
S++
S=3
S++
S=4
S++
S=5
S++
S=6
S++
S=7
S++
S=8
S++
S=9
S++
S=10
xmhuster@xmhuster-Y470:~/words/os$
```

图 1 线程同步的运行结果

运行说明：初始 s 设置为 0，线程 1 首先执行对 s 的加一操作，然后两个线程同步执行，交替进行打印和加一操作，循环次数为 10 次。

6、遇到问题及解决方案

1.如何对信号灯赋初值

对信号灯的赋初值操作需要一个联合类型 `semun`，但是 `sys/sem.h` 中并没有定义，所以通过查询资料，自己定了一个 `semun` 的联合类型，包含一个 `int` 类型的成员 `val` 进行信号灯的赋值操作。

2.创建线程后不能正确执行

原来自己仅仅只是创建了线程，并没有执行 `pthread_join()` 函数，加上函数后就正常了。

3.两个线程没有按照预想的同步进行

通过询问同学和对照自己的 P、V 操作，发现原来是自己的 P、V 操作中 `sem.sem_flg=SEM_UNDO;` 语句出了问题，我将其设置成了 `NO_WAIT`，导致信号灯不能正确控制线程的同步关系。

7、体会

原先有一些 linux 的使用经验，但也仅仅使一点点而已，这次在 linux 下进行关于线程的编程，通过老师提供的 ppt 学到了一些线程方面的东西，还有 vi 编辑器的使用以及如何编译运行 C 程序。期间遇到了很多问题，第一是自己的知识没有掌握，比如只设置了一个信号灯，第二是自己的理解有问题，没有正确理解老师提供的内容，比如上面遇到的第三个问题，还有就是初次涉及这方面的问题，多少有点生疏。不过通过自己询问老师以及同学，还是解决了这些个问题，最后得出了正确的结果。