**Semaphore使用实验**

一、题目

使用semaphore，并利用该程序生成2个进程（注意：非线程），这两个进程写同一个文件，要求：

互斥写，即只有一个进程写完后，才能让另一个进程写;

一个进程写入内容：“自己学号的后3位PROC1 MYFILE1”；

另一个进程写入内容：“自己学号的后3位PROC2 MYFILE2”），将该程序的semaphore替换成使用strict alternation算法的忙等待互斥锁完成。

回答问题：

i. 自己程序中关键句的含义

ii. 请用实际操作证明当进程A占用semaphore后，进程B想要占用semaphore时，进程B进入睡眠。

iii.移植Modern Operating System一书中的strict alternation算法时，该算法中的turn变量访问时是否需要加锁，以避免读写冲突？

二、编写程序

1.sem1.c（信号量初始值为1）

1. #include <stdio.h>
2. #include <sys/types.h>
3. #include <sys/ipc.h>
4. #include <sys/sem.h>
5. int main(){
6. int semid;
7. union semun{
8. int val;
9. struct semid\_ds \*buf;
10. unsigned short \*array;
11. struct seminfo \*\_buf;
12. } semval;
13. semval.val = 1;
14. semid = semget(0x123, 1, IPC\_CREAT | IPC\_EXCL | 0600);
15. semctl(semid, 0, SETVAL, semval);
16. }

2. semaphore1.c

1. #include <stdio.h>
2. #include <sys/types.h>
3. #include <sys/ipc.h>
4. #include <sys/sem.h>
5. #include <unistd.h>
6. int main(){
7. int flag = semget(0x123, 1, 0);
8. while(1) {
9. while(semctl(flag, 0, GETVAL) != 1); *// 等待flag为1*
10. FILE \*fp = fopen("out.txt", "a");
11. fprintf(fp, "894 PROC1 MYFILE1\n");
12. fclose(fp);
13. printf("flag = %d\n", semctl(flag, 0, GETVAL));
15. semctl(flag, 0, SETVAL, 0); *// 设置flag为0，允许semaphore2写*
16. printf("flag = %d\n", semctl(flag, 0, GETVAL));
17. }
18. return 0;
19. }

3. semaphore2.c

1. #include <stdio.h>
2. #include <sys/types.h>
3. #include <sys/ipc.h>
4. #include <sys/sem.h>
5. #include <unistd.h>
6. int main(){
7. int flag = semget(0x123, 1, 0);
8. while(1) {
9. while(semctl(flag, 0, GETVAL) != 0); *// 等待flag为0*
10. FILE \*fp = fopen("out.txt", "a");
11. fprintf(fp, "894 PROC2 MYFILE2\n");
12. fclose(fp);
13. printf("flag = %d\n", semctl(flag, 0, GETVAL));
15. semctl(flag, 0, SETVAL, 1); *// 设置flag为1，允许semaphore1写*
16. printf("flag = %d\n", semctl(flag, 0, GETVAL));
17. }
18. return 0;
19. }

semaphore1.c 在信号量值为 1 时进行操作，并将信号量值设置为 0。

semaphore2.c 在信号量值为 0 时进行操作，并将信号量值设置为 1。

运行`vi sem1.c`创建文件

运行`./sem1`创建信号量

运行`gcc -g -o semaphore1 semaphore1.c`，`gcc -g -o semaphore2 semaphore2.c`编译文件

当flag未设置成1时semaphore2.c被阻塞，程序semaphore1中flag设置成1时semaphore2执行，而后semaphore1被阻塞，如此交替循环。out.txt被交替写入，通过`cat out.txt`查看写入内容。

三、关键句的含义

1. while(semctl(flag, 0, GETVAL) != 1)

进程等待信号量 `flag` 的值变为1，然后才能继续执行。这里使用了 `semctl` 函数来获取信号量的值。

2. semctl(flag, 0, SETVAL, 0)

设置信号量 `flag` 的值为0，表示让另一个进程可以开始执行。这里使用了 `semctl` 函数来设置信号量的值。

四、证明当进程A占用semaphore后，进程B想要占用semaphore时进入睡眠

当flag未设置成1时semaphore2.c被阻塞，程序semaphore1中flag设置成1时semaphore2执行，而后semaphore1被阻塞，如此交替循环。out.txt被交替写入。

Semaphore2执行到’ while(semctl(flag, 0, GETVAL) != 0); // 等待flag为0’时因为此时flag为1，被阻塞，执行Semaphore1，当flag置1时Semaphore2可以继续执行。out.txt被交替写入。

五、strict alternation算法中的`turn`变量是否需要加锁

严格轮换算法中的 `turn` 变量是一个标志，用来指示当前轮到哪个进程执行。在严格轮换算法中，`turn` 变量是交替被读取和写入的，但是在单核系统中，读取和写入一个整数是原子操作，因此不需要额外的锁。

在多核系统中，如果多个核心同时访问 `turn` 变量，可能会出现竞态条件。因此，在多核系统中，为了保证严格轮换算法的正确性，需要使用同步原语（如互斥锁、原子操作等）来保护 `turn` 变量的读写操作。