实验八-进程同步与信号量机制

1.实验内容

在操作系统中各个线程同步运行的时候，经常会出现多个进程共享数据，对同一个资源进行访问的情况。我们在操作线程存取这些共享数据的时候，必须保证数据之间的一致性。这就是进程的同步保障机制。随之产生的有经典的生产者-消费者问题。

信号量机制是保障进程同步的一个有效机制，我们通过原子操作来对信号量进行访问。

在本节实验中，需要同学们使用信号量机制来解决一个简单的生产者-消费者问题。

- 任务一

  1. 建立一个生产者进程，五个消费者进程；
  2. 用文件建立一个共享缓冲区；
  3. 生产者进程依次向缓冲区写入整数0,1,2,...,M，M>=500；
  4. 消费者进程从缓冲区读数，每次读一个，并将读出的数字从缓冲区删除，然后将本进程ID和数字输出到标准输出；（缓冲区最多存放十个数字）

- 任务二

  去掉信号量机制相关代码，观察会发生什么现象，并尝试给出一个合理的解释。

2.实验代码示例

- 文件读写相关函数

```
从一个文件流中读数据，最多读取count个元素，每个元素size字节，如果调用成功返回实际读取到的元素个数，如果不成功或读到文件末尾返回 0。
size_t fwrite(const void* buffer, size_t size, size_t count, FILE* strea
m);
（1）buffer：是一个指针，对fwrite来说，是要获取数据的地址；
（2）size：要写入内容的单字节数；
（3）count:要进行写入size字节的数据项的个数；
（4）stream:目标文件指针；
（5）返回实际写入的数据项个数count。

int fseek(FILE *stream, long offset, int fromwhere);
函数设置文件指针stream的位置。
如果执行成功，stream将指向以fromwhere为基准，偏移offset（指针偏移量）个字节的位
```

```
置，函数返回0。
如果执行失败(比如offset超过文件自身大小)，则不改变stream指向的位置，函数返回一个非0
值。

size_t fread(void *buffer,size_t size,size_t count, FILE *stream );
buffer    是读取的数据存放的内存的指针
size      是每次读取的字节数
count     是读取次数
stream    是要读取的文件的指针
```

- 信号量相关函数

```
// 信号量的创建
sem_t * empty=(sem_t *)sem_open("empty",O_CREAT,0064,10);
sem_t * full=(sem_t *)sem_open("full",O_CREAT,0064,0);
sem_t * mutex=(sem_t*)sem_open("mutex",O_CREAT,0064,1);

// 信号量的删除
sem_unlink("xxx");

// 信号量的操作
  // 获取信号量的值，大于0时减一并返回，小于0时一直等待
  sem_wait(sem_t*);
  // 获取信号量的值，加一并返回
  sem_post(sem_t*);
```

- 提示

在目标txt共享文件中，可以用0-9位存储缓冲区数据，第10位存储当前读取到的数据下标。消费者可以通过两次对文件的读取获取到正确的数据。

3.实验结果示例

```
zr@zr-VirtualBox:~/桌面/ex9$ gcc -o sem sem.c -pthread
zr@zr-VirtualBox:~/桌面/ex9$ ./sem
2148:   0
2148:   1
2148:   2
2148:   3
2148:   4
2148:   5
2148:   6
2148:   7
2148:   8
2148:   9
2147:   10
2147:   11
2147:   12
2147:   13
2147:   14
2147:   15
2147:   16
2148:   17
2146:   18
2145:   19
2146:   20
2146:   21
```