|  |
| --- |
|  |
| 1. 实验目的及实验内容   （本次实验所涉及并要求掌握的知识；实验内容；必要的原理分析） |
| 实验目的：掌握Linux下信号量机制及其相关函数的使用，实现进程（线程）同步  实验内容：  在linux下编程实现：  参考课程P202的方法，采用信号量机制模拟读者/写者的三种优先方法，理发师，以及哲学家进餐问题，并进行测试与观察 |
| 1. 实验环境及实验步骤   （本次实验所使用的器件、仪器设备等的情况；具体的实验步骤） |
| 实验环境：  1. vmware workstation 15.0.0   1. ubuntu-18.10-desktop 2. 内核版本：linux-4.18.14   实验步骤：   1. 学习课本P202的内容。   课本P202的内容主要包括Pthread互斥锁  pthread\_mutex\_init(),pthread\_mutex\_unlock(),pthread\_mutex\_lock(),  以及<semaphore.h>中的sem\_t，sem\_init(),sem\_wait(),sem\_post()函数。  在参考完书上的示例后，又在网上进一步查找，了解了相关函数的原型，参数及其含义，用法。  之后分别实现  采用信号量机制模拟读者/写者的三种优先方法，理发师，以及哲学家进餐  各源码头文件相同，不再赘述。     1. 信号量机制模拟读者/写者的三种优先方法   读写公平：  形成fair.cpp  int变量readcount，记录当前已经在队列中的读者数量。  三个信号量 w，mutexreadcount，que。  que初值为1，限制线程根据时间先后进入等待队列。  mutexreadcount初值为1，限制readcount的改变。  w初值为1，读者和正在进行的写者可以阻止写者进入临界区，实现读写互斥，写写互斥。  当队列中有读者时会P(w)限制写者进入。  先后到来的线程会P(que)，根据入队先后顺序决定线程运行顺序。于是实现了读写公平  main函数中接收参数size决定有几个线程进行读写要求。采用rand函数随机出进程sleep时间以及进程属性（要读还是要写）尽量模拟线程的随机到来。  具体细节见代码注释。          读者优先：  形成readfirst.cpp  int变量readcount记录当前已经在队列中的读者数量。  两个信号量 w，mutexreadcount。  mutexreadcount初值为1，限制readcount的改变。  w初值为1，读者和正在进行的写者可以阻止写者进入临界区，实现读写互斥，写写互斥。  当队列中有读者时会P(w)限制写者进入。  由于没有限制，故后来的reader线程会无限制的增加readcount，而readcount减到0前会一致持有w，导致后来的写者线程无法进入临界区。于是实现读者优先。  main函数中接收参数size决定有几个线程进行读写要求。采用rand函数随机出进程sleep时间以及进程属性（要读还是要写）尽量模拟线程的随机到来。  具体细节见代码注释。        写者优先：  形成writefirst.cpp  int变量readcount，writecount记录当前已经在队列中的读者写者数量。  五个信号量 w，mutexreadcount，r，mutexwritecount，mutexpriority。  mutexreadcount初值为1，限制readcount的改变。  mutexwritecount初值为1，限制writecount的改变。  r初值为1，写者可以P(r)以限制读者进入临界区。  mutexpriority初值为1，在优先锁释放之前，若来了写者，读者，写者会P(r)而读者因为优先锁没有释放不能P(r)。同时写者P(r)，阻碍后续的读者增加readcount，于是实现写者优先于读者。  w初值为1，读者和正在进行的写者可以阻止写者进入临界区，实现读写互斥，写写互斥。  当队列中有读者时会P(w)限制写者进入。  由于没有限制，故后来的reader线程会无限制的增加readcount，而readcount减到0前会一致持有w，导致后来的写者线程无法进入临界区。于是实现读者优先。  main函数中接收参数size决定有几个线程进行读写要求。采用rand函数随机出进程sleep时间以及进程属性（要读还是要写）尽量模拟线程的随机到来。  具体细节见代码注释。         1. 信号量模拟理发师问题   形成barber.cpp  define chairs数量为5，若有需要可以更改。  三个信号量cuts，bars，mutex。  一个int变量count，初值为0。count量与cuts数相同。  mutex限制count更改。  cuts为等待理发的顾客数，不包括正在理发的客人数，初值为0。bars为正在等待理发师的顾客数，初值为0.  main函数中接收参数size决定有几个线程进行读写要求。采用rand函数随机出进程sleep时间尽量模拟想要理发的客人随机到来。  实现上就参考了课件。  Barber线程中有顾客（P（cuts））则v（bars）给客人理发。  customer线程中，当count数大于chairs，说明等待座位不够，顾客离开理发店。否则P(bars)等待理发师。  于是实现理发师问题的解决。  IMG_256  IMG_256  具体细节见代码注释。         1. 信号量模拟哲学家进餐问题   形成philosopher.cpp  五个线程thread0,1,2,3,4分别代表5个哲学家。  五个信号量c0，1,2,3，4代表五个筷子。初值为1.  采用资源分级解法：前四个哲学家i都必须先拿编号为(i+1)mod5的筷子，而最后一个哲学家先拿c4。放筷子的时候先放先后到的筷子，这样一定不会有死锁。  每个线程中rand sleeptime，模拟哲学家的思考时间。吃饭用1秒，sleep(1)。  主进程最后杀死线程，退出。  于是实现解决了哲学家进餐问题。 |
| 1. 实验过程分析   （实验分工，详细记录实验过程中发生的故障和问题，进行故障分析，说明故障排除的过程及方法。根据具体实验，记录、整理相应的数据表格等） |
| 遇到的问题及解决：  1.  IMG_256  解决：应该放到main函数里初始化  2.  IMG_256  低级的错误。不应该用函数名做变量名。  3.出现 coredump错误。  检查后发现是数组访问越界，修改变量增加范围后解决。  实验结果记录：  读写公平：  IMG_256  写者优先：  IMG_256  读者优先：      理发师：  IMG_256  哲学家；  IMG_256 |