EXP4 经典同步互斥问题

汪楚文 2018202114 05/27/2020

实验概述

[实验目的]

·通过编写程序熟悉并发，同步/互斥，死锁等概念

[基本要求]

编写Linux环境下C程序，实现3个经典的同步/互斥问题

[具体要求]

·生产者/消费者问题（20）

·读者/写者问题（20）

·哲学家问题（10）

·代码有注释（10），提交实验报告和源代码（实验报告要求见3）

[进一步要求]

·多个生产者和消费者（10）

·多个读者和写者，考虑读者优先或是写者优先（10）

·用区别于示例代码的思想实现哲学家问题（10）

·可视化展示输出结果（10）

[特别说明]

·杜绝抄袭，一定杜绝抄袭，绝对杜绝抄袭

[实验报告要求]

详见文件README.txt

【重要】本实现的全部要求都已实现，生产者消费者个数在define中定义；读者和写者个数用户可在用户界面输入，并进一步在menu选择采取读者优先还是写者优先；哲学家问题采取了四种不同方法防止死锁，用户可在用户界面menu选择采取何种方法。另：三个程序都设计了比较完善的可视化，计时器等，具体效果见实验结果截图。

实验思路

[实验环境]

Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-88-generic x86_64)

[设计思路]

根据实验要求，mpro.c,mraw.c,mphi.c的主要设计思路如下。

一.生产者消费者问题(mpro.c)

[框架设计]

多个生产者分别对应不同的producer_id;多个消费者对应不同的consumer_id。设置共享内存buffer，buffer本身又作为一个index，索引至char *production中；于是不同的生产者可以生产出production中不同的物品。消费者也可以消费buffer中值所索引到的对应production；

[互斥与同步设计]

互斥和同步使用semaphore和mutex共同完成。其中两个信号量一个用于判断buffer中空的程度，另一个用于衡量buffer中满的程度。互斥量mutex用于控制每次只能有一个进程访问buffer；具体实现为：producer等待有空的位置-->等待现有进程访问buffer结束-->produce-->退出访问buffer，允许其他进程访问buffer-->描述buffer填充程度 的信号量加一；consumer等待填充-->等待现有进程访问buffer结束-->consumer-->退出访问buffer，允许其他进程访问buffer-->描述buffer空闲程度的信号量加一。

[可视化设计]

程序背景为“KFC”，char *production中储存的都是“KFC”的食品。在用户界面，用户可以在menu下选择（1）查看排队状态；（2）查看菜单；（3）离开。用户选择了（1）后，程序即可显示kfc的员工（producer）制作的产品，目前各窗口（buffer）等待取餐的产品，以及顾客（consumer）取走了几号窗口的什么产品。

二.哲学家问题(mphi.c)

[主体设计]

本程序有4种防止死锁方法供用户选择，用户可在menu下选择想用的防死锁方式，选择后，用户可以实时看见哲学家的吃饭/思考/饥饿情况，并且在哲学家就餐完成后，程序会输出通过用户选择的防死锁方式，哲学家问题所花费的时间。用户可以尝试不同防止死锁的方法，来对比各种方法所使用的时间来判断各种方法的效率。除此之外，可视化最终还会输出哲学家问题的整个事件发生时间轴，用户可以更加直观的看到任何哲学家是在什么时候吃饭的，由此用户可以更加深入理解每一种防死锁方法的具体执行情况。

[四种防死锁方法]

（1）用互斥量使将拿两只筷子变为一个原子操作

（2）奇数和偶数位置哲学家拿筷子顺序相反

（3）哲学家只能先拿编号大的筷子

（4）用信号量使同时拿筷子的哲学家数量小于n-1

三.读写者问题(mraw.c)

[同步与互斥设计]

通过四个mutex来实现对readcount和writecount加减时，读者进程之间或者写者进程之间不会相互干扰。再通过readcount和writecount的值来判断是否要给写者/读者进程上锁或者解锁。

[读者优先和写者优先]

读者优先即采用上课时老师展示的伪代码的思路；在将读者优先改为写者优先时，除了要多设置一个writecount来判断是否有write进程等待，若没有，这时才允许读者进程读取，于是就实现了写者优先。但是除此之外，写者进程还需要彼此互斥，故程序设置了一个互斥量来防止多个写者进程同时写。同时，读者进程也需要设置新的互斥量来保证readcount++和read操作的正常进行。

[可视化]

程序有menu供用户选择读/写者数量，以及采取读者优先还是写者优先策略，程序会实时显示哪一个读者开始/结束读，哪一个写者开始/结束写。并且用户可以限制读写进程运行时间，这样程序过一段时间就会停止，不会无限循环下去。停止后，可视化最终还会输出读写的整个事件发生时间轴，用户可以更加直观的看到任何读者/写者的执行过程，由此用户可以更加深入理解互斥与同步的具体执行情况。

实验结果

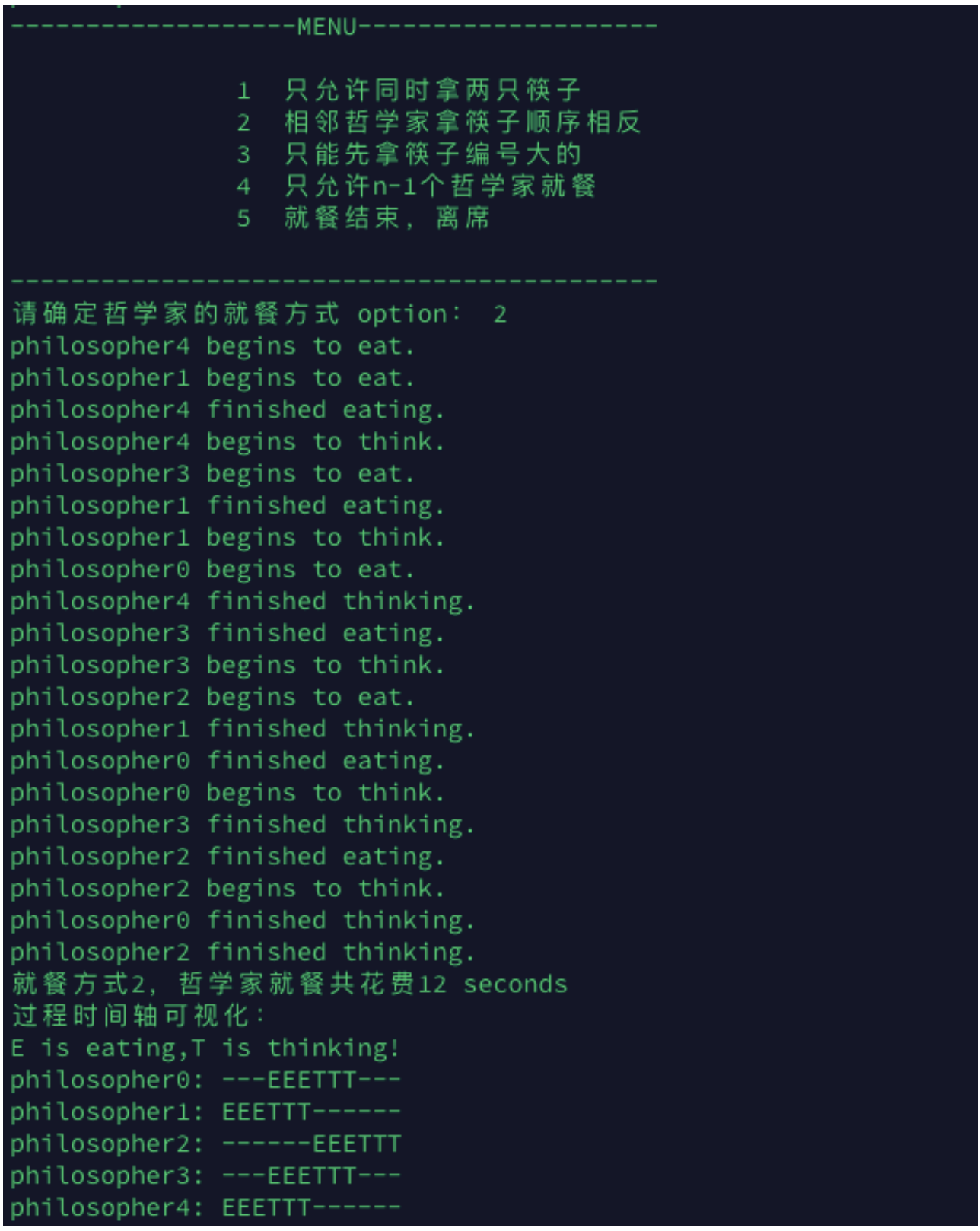
一.生产者消费者问题(mpro.c)

```
欢迎来到KFC!
----------------------MENU----------------------

                    1    查看排队状态
                    2    查看菜单
                    3    离开

------------------------------------------------
option: 1
请输入您的等待时间:(建议输小点，这样很快结束，如8) 3
嫩牛五方-已经做好，请到4号窗口取餐
KFC前台:
  0号窗口      1号窗口      2号窗口      3号窗口      4号窗口
   请等待       请等待       请等待       请等待     嫩牛五方-
顾客4已取走在4窗口的嫩牛五方-.
KFC前台:
  0号窗口      1号窗口      2号窗口      3号窗口      4号窗口
   请等待       请等待       请等待       请等待       请等待
香辣鸡翅-已经做好，请到0号窗口取餐
KFC前台:
  0号窗口      1号窗口      2号窗口      3号窗口      4号窗口
 大波纹薯条     请等待       请等待       请等待       请等待
顾客5已取走在0窗口的香辣鸡翅-.
KFC前台:
  0号窗口      1号窗口      2号窗口      3号窗口      4号窗口
   请等待       请等待       请等待       请等待       请等待
香甜玉米棒已经做好，请到1号窗口取餐
KFC前台:
  0号窗口      1号窗口      2号窗口      3号窗口      4号窗口
   请等待     香甜玉米棒     请等待       请等待       请等待
顾客6已取走在1窗口的香甜玉米棒.
KFC前台:
  0号窗口      1号窗口      2号窗口      3号窗口      4号窗口
   请等待       请等待       请等待       请等待       请等待
嫩牛五方-已经做好，请到2号窗口取餐
KFC前台:
  0号窗口      1号窗口      2号窗口      3号窗口      4号窗口
   请等待       请等待     嫩牛五方-     请等待       请等待
顾客4已取走在2窗口的嫩牛五方-.
KFC前台:
```

二.哲学家问题(mphi.c)



三.读写者问题(mraw.c)

```
请输入读者和写者的数量，用空格隔开：3 3
--------------------MENU---------------------

                1  读者优先
                2  写者优先
                3  退出

-----------------------------------------------
 option:  2
WRITER 2: waiting to write
WRITER 2: start writing
WRITER 1: waiting to write
WRITER 0: waiting to write
READER 2: waiting to read
READER 1: waiting to read
READER 0: waiting to read
WRITER 2: end writing
WRITER 1: start writing
WRITER 1: end writing
WRITER 0: start writing
WRITER 0: end writing
READER 2: start reading
READER 1: start reading
READER 0: start reading
READER 2: end reading
READER 1: end reading
READER 0: end reading
过程时间轴可视化：
R is reading,W is writing!
READER0: --------------------RR
READER1: --------------------RR
READER2: --------------------RR
WRITER0: ------------------WW
WRITER1: ----------------WW
WRITER2: --------------WW
```

实验中遇到的问题及解决办法

一.在哲学家问题中，使用信号量进行互斥，但是不知道怎么给信号量初始化，一开始只直接令其等于一个int，产生错误。

解决方法：在网上了解了semaphore的初始化后，调用函数int error=sem_init(&count,0,4);来初始化信号量。

二.想设计一个计时器，一来比较各个策略的效率，另一方面可以保存各个时间的发生时间，可以最后建立时间轴输出，有利于可视化。但在一开始，调用函数clock得到的时间总是特别短。

解决方法：经过查询，clock是记录程序的cpu时间，并不能准确计算如sleep的时间。于是换了计时方法start=time((time_t*)NULL)来计时，这样得到的时间是整数，而且包含sleep的时间。然后再用二维数组保存各个时间点，最后利用visual函数可视化输出。

三.在将读者优先改为写者优先时，忘记加写者之间的互斥，导致逻辑错误。

解决方法：用信号量控制写者之间的互斥，逻辑恢复正常。

四.在读写者程序中，用scanf输入读写者数量，然后根据输入的值直接建立相应大小的数组，编译的时候报错。

解决方法：通过学习，发现有的时候scanf num然后再array[b]是不允许的，于是改为动态分配内存的方法。 readerid=malloc(sizeof(int) * a); writerid=malloc(sizeof(int) * b);

五.创建信号量后不知道要销毁。

解决方法：用 sem_destroy(&count);销毁创建的信号量。

六.编译时忘加-lpthread

解决方法：想起来之前写过的程序，意识到忘记加了，于是补上了。

项目相关文件说明

OS_EXP4.pdf----------实验报告

README.txt----------实验要求文件

mpro.c----------生产者消费者问题

mphi.c----------哲学家问题

mraw.c----------读写者问题
超链接：点击此处可在github中查看项目：）