

同步机制及应用编程实现与比较

|  |  |
| --- | --- |
| 学 号： |  |
| 姓 名： |  |
| 学 院： |  |
| 提交日期： | 2022.10.10 |

1. **实验目的**

探索、理解并掌握操作系统同步机制的设计和实现机理，针对所谓的银行账户转账同步问题，构建基于 Peterson 算法的同步解决方案以及基于 Windows（或 Linux）操作系统同步机制（主要是互斥机制）的解决方案，并进行分析和比较。

1. **实验内容**

针对所谓的银行账户转账同步问题，分析、设计和利用 C 语言编程实现基于 Peterson 算法的同步解决方案，以及基于 Windows（或 Linux）操作系统同步机制的相应解决方案，并就自编同步机制与操作系统自身同步机制的效率进行比较和分析。

1. **实验要求**

同步机制及应用编程实现与比较实验功能设计要求：

（1）银行账户转账同步问题的抽象及未采取同步控制情况下的编程实现；

（2）基于 Peterson 算法的银行账户转账同步问题解决方案；

（3）基于 Windows（或 Linux）操作系统同步机制的银行账户转账同步问题解决方案；

（4）Peterson 算法同步机制和 Windows（或 Linux）操作系统同步机制的效率比较。

**4.实验环境**

操作系统：Windows11

IDE：codeblocks

**5.实验分析**

首先本次实验要求针对银行账户转账问题实现三个内容。简单来说银行账户转账问题就是将资金从一个账户转移到另一个账户上的问题，但在执行上，会出现许多问题。

其中最为重要也是难以解决的是同步线程的问题。比方说一个人在转账，另一个资金账户上那么必然有着应有的到账，这二者从理论上来说应该是同步的，这一性质决定了整个问题的安全性和有效性，也意味着同步线程是解决问题的关键。

首先分析第一种情况，即在未采取同步控制情况下的编程实现。我是这样分析这个问题的，既然这种情况的要求是不需要采取同步控制，而实际肯定要让账户内容尽量“同步”，那么也就说明我需要采取一种“伪同步”的思想。即我可以在一个账户转出了指定的资金后，另一个接收账户立刻增加接收的资金。这样可以解决未采取同步控制情况下的银行账户转账问题。

第二个要求是用peterson算法实现银行账户转账同步问题。该算法为每个进程设置标志flag，用于表示该进程是否有意访问临界资源（进入临界区）；又设置了标志turn，用于表示临界资源是否有其他进程在访问。

只有在对方进程的访问标志flag为true，并且turn也为该进程标识时，才表明对方进程在访问临界资源（占据临界区），需要等对方进程访问结束并且释放该临界资源后，才能开始访问；否则可以直接进入临界区，无需等待。

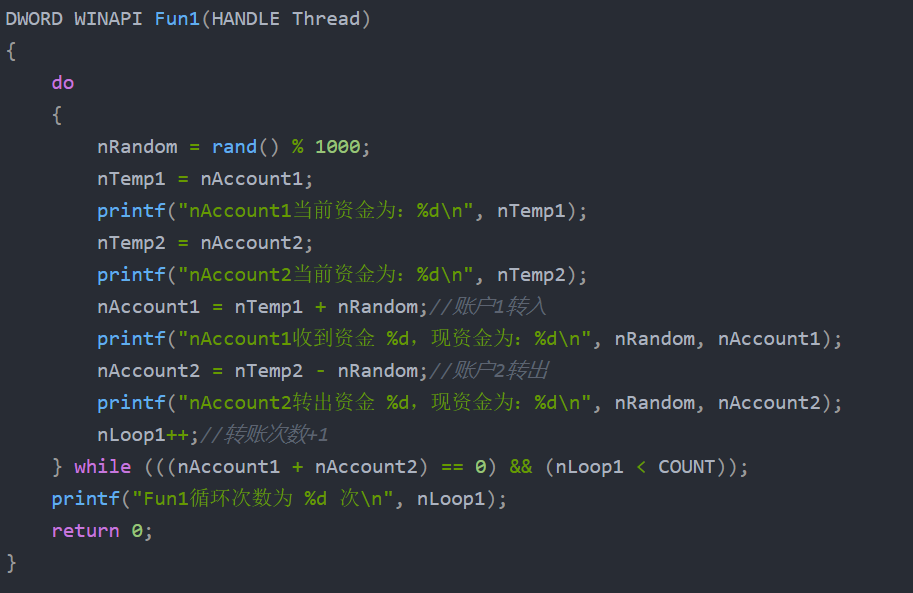
对于一个进程Pi而言，如果它想进入临界区，首先它要告诉操作系统它想进入临界区（flag[i]=true），但此时它还不能进入临界区。这时操作系统说，由于进程Pi没有进入临界区，进程Pj就可以进入临界区（turn=j）。当然Pj也可以选择不进入临界区，那么Pi就进入临界区，在它操作完成之后，它需要告诉操作系统自己不再想进入临界区了，然后把flag[i]设为false，以避免占有“进入的欲望”。但是，如果进程Pj选择进入临界区，那么进程Pi就得等待。这个代码的设计思路在于——“让”。进程Pi提出请求后，操作系统会先把进入权给予其他进程，确保其他进程都不想进入的情况把，才允许进程Pi进入临界区。

在使用Windows系统进行线程同步时，我使用了WaitForSingleObject函数。这是一个windows API 函数。当等待仍在[挂起状态](https://baike.baidu.com/item/%E6%8C%82%E8%B5%B7%E7%8A%B6%E6%80%81/9359976?fromModule=lemma_inlink)时，句柄被关闭，那么函数行为是未定义的。该句柄必须具有 SYNCHRONIZE 访问权限。

WaitForSingleObject函数用来检测hHandle事件的信号状态，在某一线程中调用该函数时，线程暂时挂起，如果在挂起的dwMilliseconds毫秒内，线程所等待的对象变为有信号状态，则该函数立即返回；如果时间已经到达dwMilliseconds毫秒，但hHandle所指向的对象还没有变成有信号状态，函数照样返回。

**6.具体代码实现**

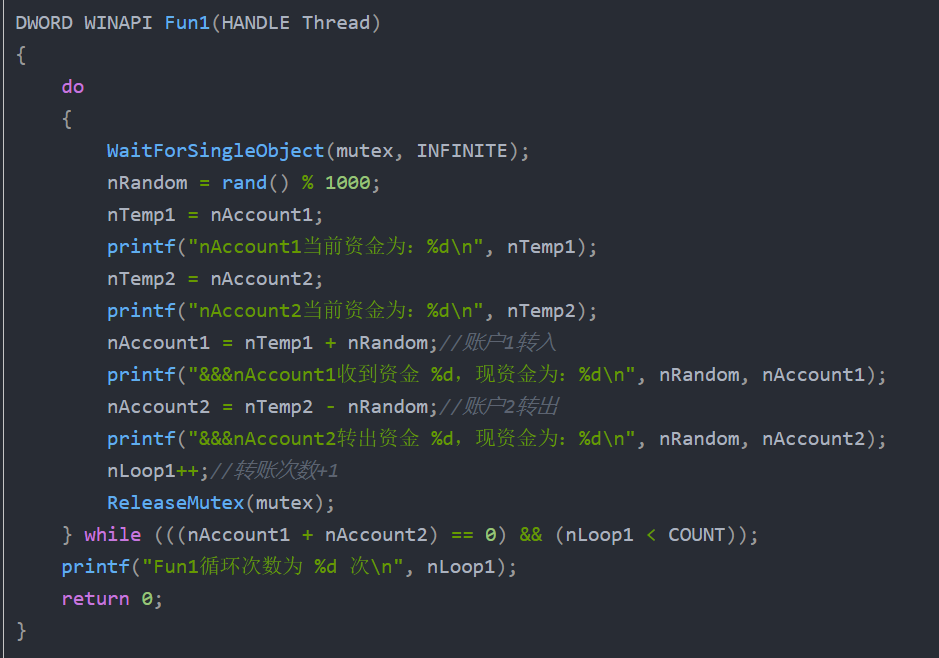
（1）对于银行账户转账同步问题的抽象及未采取同步控制情况下，关键函数如下：



（2）基于 Peterson 算法的银行账户转账同步问题解决方案，关键函数如下：

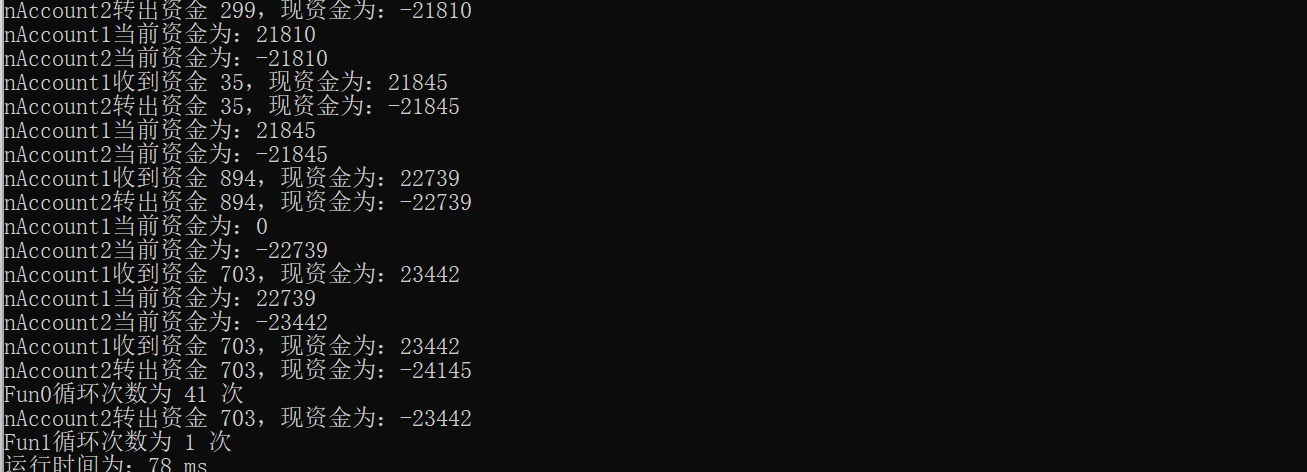


（3）基于 Windows（或 Linux）操作系统同步机制的银行账户转账同步问题解决方案，关键函数如下：



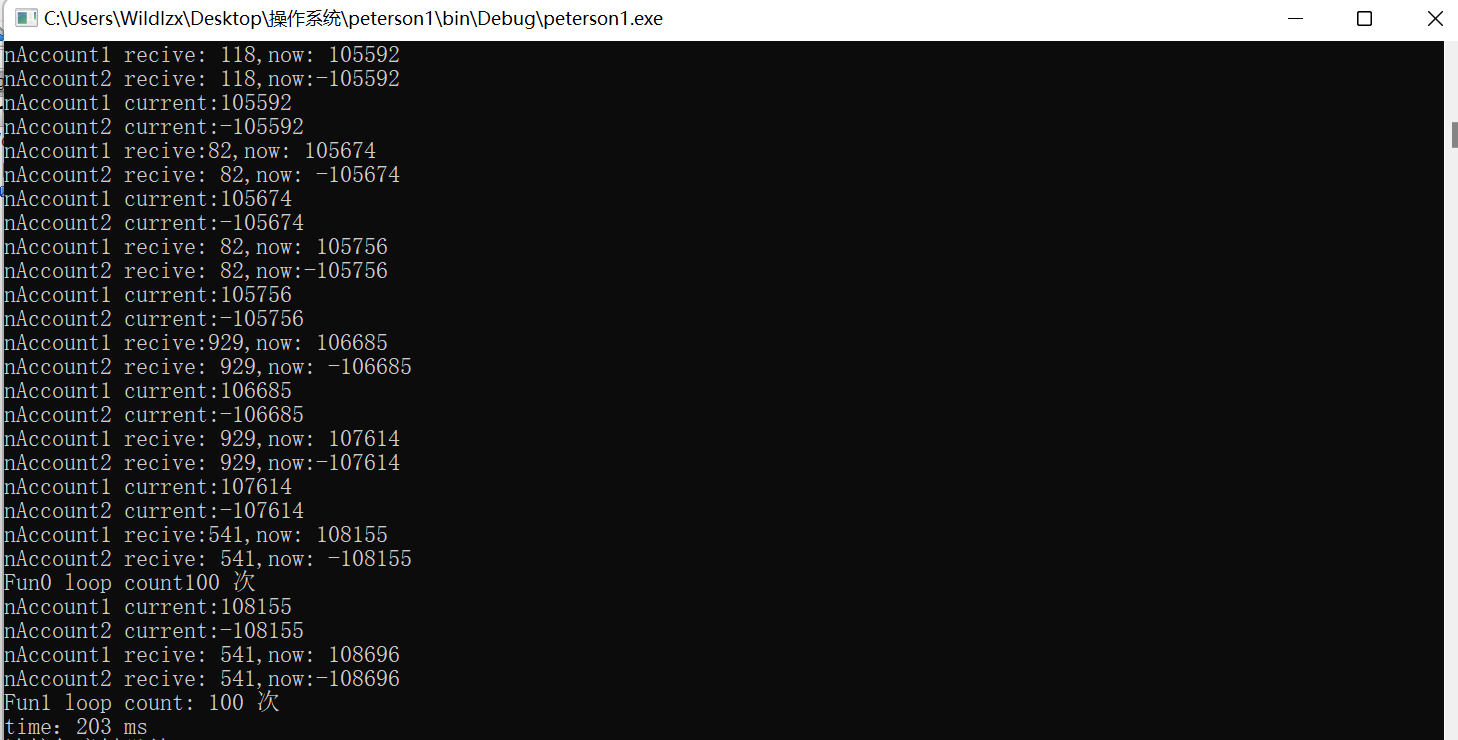
**7.功能测试**

**（1）未采取同步控制下结果：**

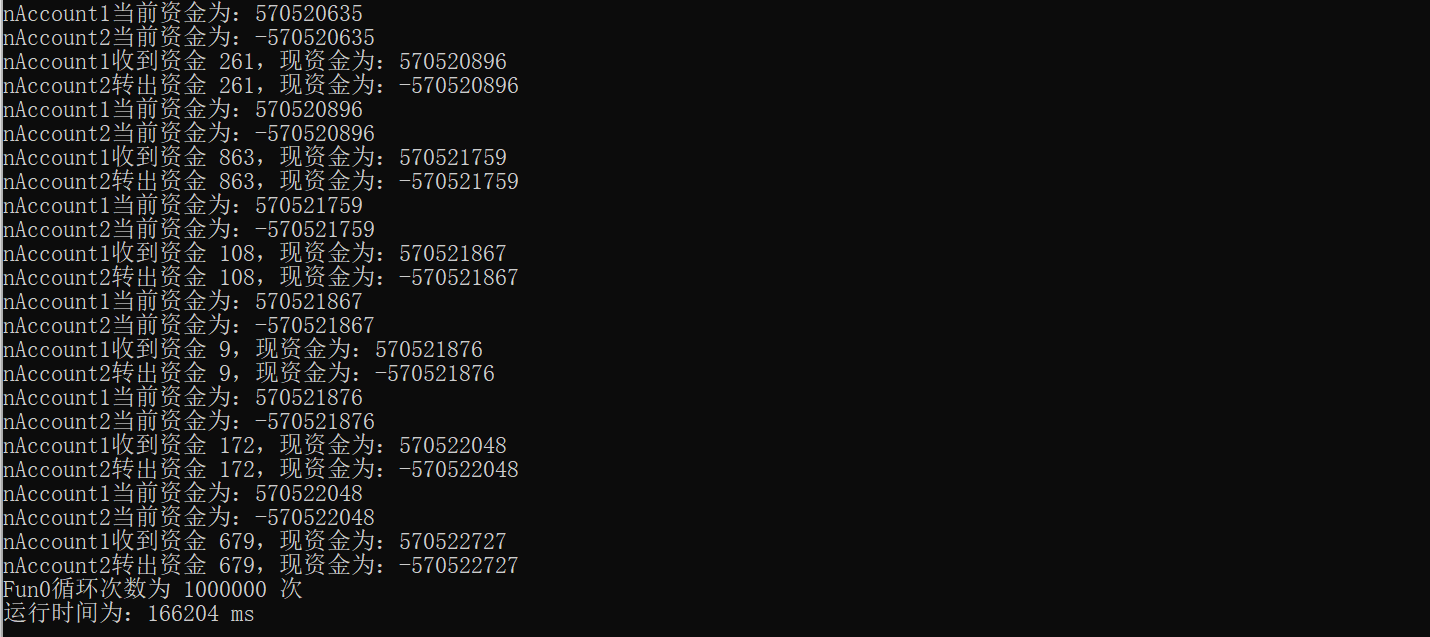
****

由于该方法没有设置线程信号量，所以在线程的执行上有一定的随机性，因此可能会在某一次交易执行完成后两账户和并不为定值，所以跳出循环，未到预定的循环次数。

**（2）peterson算法测试结果**

****

**（3）基于windows操作系统解决同步测试结果:**

****

比较所有测试结果可以看出，peterson算法的时间最短，也保持了正确性；而基于windows操作系统解决同步可以保证正确性，但时间较长；未采取同步控制下会出现问题从而提前结束。综上所述，peterson算法时间最短，效率最高，最为优秀。

**8.心得体会**

本次实验是操作系统课程的一个基础实验，是对于线程同步的基本的探究和实现，通过三种不同的方法实现银行转账问题，并对比了不同方法的正确性和效率。

通过这次实验，我学习到了很多。首先更加深刻地理解了操作系统中线程同步的设计原理和具体实现的方法，再通过自己对peterson算法的研究和理解，成功实现了目的。并且，这次实验也锻炼了我测试和完善程序的能力，通过一步一步的修改，使之达到了预计的效果，收获颇多。