**华东师范大学计算机科学技术系实验报告**

|  |  |  |
| --- | --- | --- |
| **课程名称**：操作系统实践 | **年级**：2015级 | **实践作业成绩：** |
| **指导教师**：吴苑斌 | **姓名**：陈越 | **提交作业日期**：2017-12-8 |
| **实践编号**：4 | **学号**：10152130155 | **实践作业编号**： |

1. 实验名称： Locks and Threads

实验目的：

了解并熟练掌握锁的用法及实现

进一步掌握makefile文件的用法

对数据结构的温习

写报告和肝实验的能力

1. 实验工具

Ubuntu 16.04 LTS

Vmware 17.0

Sublime Text 3

1. 实验过程

**（一）Spinlock**

spinlock的思路非常简单，对资源设置一个变量，这个变量的取值有两个：0和1。如果变量值为0，那么表示这个资源处于可以访问的状态；变量值若为1则表示这个资源处于不可访问状态，需要等到资源处于可访问状态时才能访问资源。

实现起来也非常简单，上锁就将变量设为1，释放就将变量设为0。一个进程需要访问资源的话，先询问资源是否可以访问，可以访问则访问并上锁，如果无法访问则空转等待直到资源可以访问。

**（二）Spinlock(LoadLinked-StoreConditional)**

对于上一个Spinlock，我们可以看到如果一个进程访问一个已上锁的资源会处于空转状态，这样十分浪费CPU资源。Spinlock(LoadLinked-StoreConditional)以下简称（Spinlock(LS)算法）则稍微缓解了一些情况，Spinlock(LS)算法则先把要访问的资源的锁状态进行访问，如果是0也就是可以访问资源，则访问资源并上锁；如果无法访问资源，则回退。

**（三）Mutex**

Spinlock虽然易于实现，但是如果调度策略不够好，进程要么得到资源要么就空转，极大的影响了CPU资源，而且还隐藏了一个问题：处于后面的进程可能永远无法访问到资源，这当然不是我们想看到的。

此时，我们需要另外一种策略：当进程阻塞的时候，我们需要让其睡眠而不是让其白白空转；等到资源空闲的时候，我们在将睡眠的进程唤醒。结合Linux上的函数，我们也可以比较容易地实现算法。（具体代码见附件）。

**（四）Two\_Phase Lock**

与之前的算法相比，Two\_Phase算法意识到空转或许是一种有效的措施，只是我们之前没有意识到这一点。进程遇到已经上锁的进程阻塞一下，或许不久就能得到资源，而如果我们将其睡眠再唤醒可能要比空转所消耗的资源更大。基于这个思想，我们根据mutex，也能实现这个算法。

1. 实验结果

五种锁的对比：（Counter）

我们可以看到：Counter数量较小的时候，五种锁波动都很大，单都处于上升的趋势，其中Spinlock很不稳定；随着Counter数量的逐渐增大时，时间仍处于增长的趋势，并且spinlock增长的趋势最大。同时：当线线程数量较小时，Mutex的表现不是非常好，可能因为需要大量的调用CPU资源处理有关。

综上分析：五种锁的综合性能：Pthread\_lock>Two\_Phase>Spinlock(LS)>

Mutex>Spinlock

List:

综上分析：当list数目较小时，各种锁略微有一些波动，其中还是Spinlock的波动较大，随着list数目的增大，消耗的时间也都随着增大，其中值得注意的是，Mutex消耗的时间有些显著，这或许与其不断地调用CPU有关系。

Hash：

综上分析：当hash数目较小时，各种锁略微有一些波动，其中还是Spinlock的波动较大，随着hash数目的增大，消耗的时间也都随着增大，其中值得注意的是，Mutex消耗的时间有些显著，这或许与其不断地调用CPU有关系。

在List上进行插入和删除操作

在HashTable上进行插入和删除操作

通过对list和hashtable进行insert和delete操作，我们可以看到时间明显消耗时间明显增加了，另外从中也可以看到锁的性能大概是Pthread\_lock>

Two\_Phase>Mutex>Spinlock

最后实验平台也可能会对实验结果有所影响：CPU可以是单核或者多核，通过对虚拟机对硬件进行设置便可以进行测验，我们组认为这个只需要设置一下重复实验即可，没有太多需要阐述的内容，故而这一段不做过多叙述。

以上便是我们组的本次实验结果。

五 附录

实验代码有README，里面有运行程序的方法。