Pintos Project 1 设计方案

1. 概述
2. 需求分析

2.1 Alarm Clock 需求分析

2.2 Priority Scheduling 需求分析

此任务是实现一个优先级捐献的问题，主要完成的目标是：若当前线程占用的锁优先级比当前线程高，则将当前线程的优先级提升，如果当前线程占用的锁优先级低则锁的优先级提高到与当前线程一样；当释放锁时，锁的优先级应该检查是否需要更新，当前线程的优先级也要检查是否需要更新。

2.3 Advanced Scheduler 需求分析

此任务是实现一个多级反馈队列调度算法（mlfqs），其的主要目的是：根据不同CPU区间的特点以区分进程，将占用过多CPU时间的进程转移到更低优先级队列，以保证I/O约束和交互进程等占用CPU时间较少的线程留在优先级更高的队列，从而减少系统的平均响应时间。

为了实现这一功能，我们需要设计一定的算法，将占有CPU时间过长的进程转移到优先级低的队列中。

1. 设计思路

3.1 Alarm Clock 设计思路

3.2 Priority Scheduling 设计思路

1. 在当前线程获得一个锁时，如果线程的优先级比拥有该锁的其他线程的最大优先级高，则提高锁的优先级，并且如果这个锁还被其他的锁锁着，则递归的捐赠优先级。当释放这个锁之后，递归地恢复到之前的优先级
2. 如果一个线程被多个线程捐赠，则应当将当前线程的优先级设置为被捐赠的最高级
3. 当释放锁的时候，如果一个锁的优先级发生改变，应该考虑其他被捐赠的优先级和当前的优先级
4. 将信号量的等待队列实现为优先队列
5. 将condition队列实现为优先队列

具体做法如下：

在thread数据结构中加入成员init\_priority, locks, lock\_waiting，其中init\_priority记录创建时的优先级，locks记录线程所拥有的锁，lock\_waiting记录的时线程申请的锁。

在lock数据结构中加入成员elem, max\_priority成员，其中elem记录等待该锁的线程，max\_priority为上述队列中的最大优先级。通过添加以上成员记录线程捐献优先级的过程。

修改lock\_acquire函数，让线程能够递归地实现优先级捐赠。

添加thread\_donate\_priority和thread\_hold\_the\_lock函数，thread\_donate\_priority函数作用是线程将优先级捐赠出去，thread\_hold\_the\_lock函数则是将线程插入锁的elem队列当中。

添加锁队列排序函数，让队列能够按照优先级从大到小顺序排列。

Thread\_remove\_lock实现的时当释放锁后线程优先级的变化。

Thread\_update\_priority实现线程优先级的更新。

修改thread\_set\_priority函数，实现一个线程处于被捐赠状态时，对init\_priority进行设置，并且如果设置后的优先级大于当前的优先级，则改变当前的优先级。

实现condition队列和信号量的等待队列为优先队列。

3.3 Advanced Scheduler 设计思路

根据文档，BSD Scheduler的实现思路为：每过一定的时间，重新计算线程的优先级，以降低占用过多CPU时间的线程的优先级，从而实现将其“转移到低优先级队列”。

此调度算法程序可以用以下参数来定义：

（1）队列数量：64

可以认为是每个优先级数值对应一个队列。

（2）每个队列的调度算法：FCFS

由于此64个队列，都属于ready\_list，因此采取的调度算法均为FCFS

（3）升高/降低到其他队列的算法：通过动态计算线程优先级的数值来确定线程队列的 改变。

（4）确定进程在需要服务时应进入的队列：可以认为按优先级的数值插入到对应队列。

其中，（3）的具体做法为：

首先，为每一个线程增加参数nice、recent\_cpu属性用于实现线程优先级的动态计算。设定全局变量load\_avg，用于计算recent\_cpu。

其中nice值的含义为：该线程对其他线程的友好程度，nice值越大，该线程的优先级会越低。

recent\_cpu的作用为：衡量线程所占用cpu时间的多少。

load\_avg的作用为：估计过去1分钟运行线程的平均数量。

其次，改变的优先级算法为：

① 每个时钟周期，正在运行的线程的recent\_cpu增加1。即在时间中断函数time\_interrupt() 中加入功能，每次中断为当前运行线程的recent\_cpu值增加1.

② 每过4个tick( tick % 4 == 0 )，重新计算优先级，公式为

③ 每过100个tick(tick % timer\_freq == 0),通过以下公式更新recent\_cpu和load\_avg

此外，由于pintos不支持浮点数的运算，需按照浮点数运算规则，实现浮点数的运算。