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

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

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。

假设现有Thread1和Thread2两个线程，其nice值分,4和8，tick=0时开始运行，两个线程的初始recent\_cpu均为0。首先进行判断tick%100==0是否成立，此时tick%100==0，计算load\_avg和两个线程的recent\_cpu，load\_avg = 59/60 \* 0 + 1/60\*2 = 1/30，线程A的recent\_cpu = nice = 4，线程B的recent\_cpu = nice = 8。然后判断tick%4 == 0是否成立，此时成立，计算priority，分别为54和45，此时两个线程分别属于优先级数值为54和45对应的队列。

此时，ThreadA的优先级更高，先于ThreadB执行。此后每一个tick都会进行上述的判断，并将正在运行的线程的recent\_cpu加1。36个tick之后，ThreadA的优先级降低到45，插入到优先级45对应的队列的队尾。下4个tick，ThreadB运行。此后两个线程交替执行，直到线程运行结束。