# Lab6

实现 Stride Scheduling 调度算法,请在实验报告中简要说明你的设计实现过程

设置一个大整数，将来的所有pass值都由这个大整数除以进程的优先级得到

将进程加入运行队列时，插入到斜堆中，并将运行队列的进程计数加一，同时需要在进程的数据结构中关联运行队列

将进程从运行队列移走时，需要将进程从斜堆中删除，并将运行队列的进程计数减一

选择接下来调度的进程时，如果使用斜堆只需要取出堆顶，随后要更新进程的stride值。

处理时钟和RR算法一样，如果time slice大于0，则将值减一。否则以为着时间片用完，需要调度

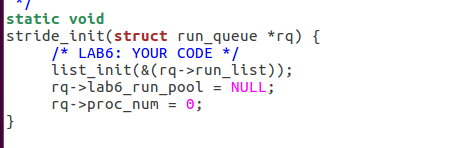
首先需要换掉RR调度器的实现，即用default\_sched\_stride\_c覆盖default\_sched.c。然后根据 此文件和后续文档对Stride度器的相关描述，完成Stride调度算法的实现。

#### 设计实现

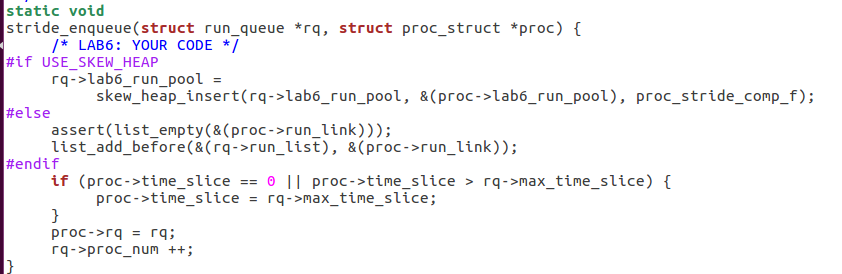
由于在ucore中使用面向对象编程的思想，将所有与调度算法相关的函数封装在了调度器sched\_class中，因此其实可以不需要覆盖掉default\_sched.c，只需要将default\_sched\_stride\_c改名成default\_sched\_stride.c，然后注释掉default\_sched.c中的sched\_class的定义，这样由于default\_sched\_stride.c中也有sched\_class的定义，其他代码在调用调度器的接口的时候就直接调用了新实现的Stride Scheduling算法实现的函数了；

stride调度算法的设计实现：

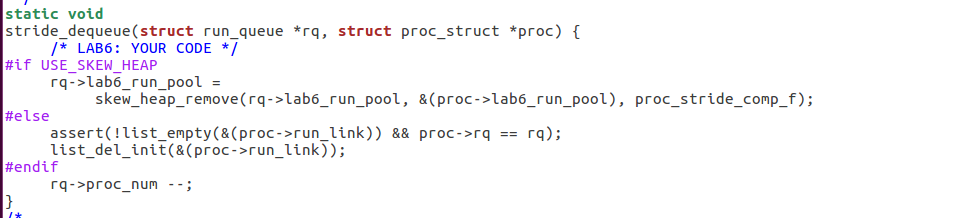
stride\_init: 进行调度算法初始化的函数，在本stride调度算法的实现中使用了斜堆来实现优先队列，因此需要对相应的成员变量进行初始化；



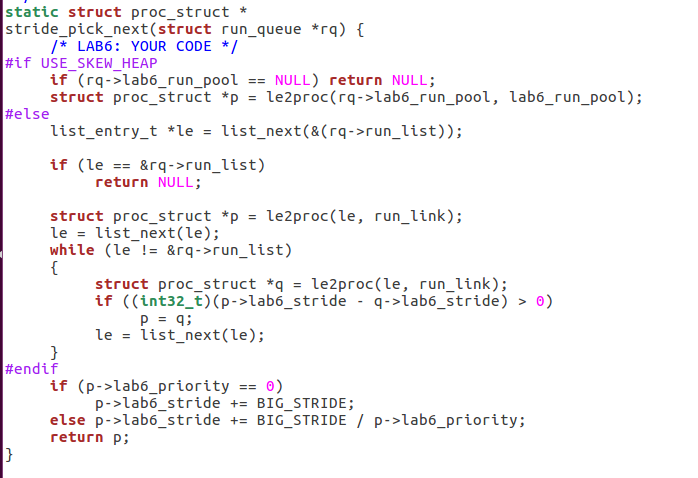
stride\_enqeue: 在将指定进程加入就绪队列的时候，需要调用斜堆的插入函数将其插入到斜堆中，然后对时间片等信息进行更新；

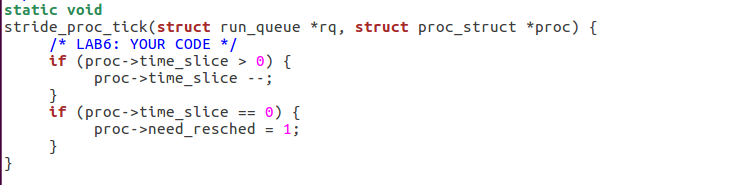


stride\_dequeue：将指定进程从就绪队列中删除，只需要将该进程从斜堆中删除掉即可

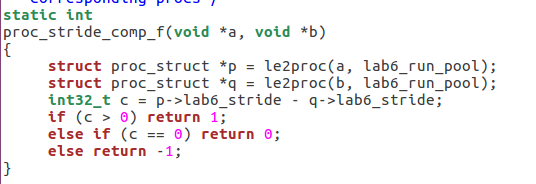


stride\_pick\_next: 选择下一个要执行的进程，根据stride算法，只需要选择stride值最小的进程，即斜堆的根节点对应的进程即可：

stride\_proc\_tick：每次时钟中断需要调用的函数，与RR算法中的实现没有区别



优先队列的比较函数proc\_stride\_comp\_f的实现，主要思路就是通过步数相减，然后根据其正负比较大小关系。



最终运行结果如下：

