操作系统实验三

一．实验目的

1.理解操作系统的调度管理机制

2.熟悉ucore的系统调度器框架，以及缺省的Round-Robin调度算法

3.基于调度器框架实现一个(Stride Scheduling)调度算法来替换缺省的调度算法

二．实验内容

练习1: 使用 Round Robin 调度算法（不需要编码）

请在实验报告中完成：

请理解并分析sched\_calss中各个函数指针的用法，并结合RoundRobin 调度算法描 ucore的调度执行过程 请在实验报告中简要说明如何设计实现“多级反馈队列调度算法”，给出概要设计，鼓励给出详细设计

答：RoundRobin 调度算法描 ucore的调度执行过程如下：

1.在ucore中调用调度器的主体函数（不包括init，proc\_tick）的代码仅存在在wakeup\_proc和schedule，前者的作用在于将某一个指定进程放入可执行进程队列中，后者在于将当前执行的进程放入可执行队列中，然后将队列中选择的下一个执行的进程取出执行；

2.当需要将某一个进程加入就绪进程队列中，则需要将这个进程的能够使用的时间片进行初始化，然后将其插入到使用链表组织的队列的对尾；这就是具体的Round-Robin enqueue函数的实现；

3.当需要将某一个进程从就绪队列中取出的时候，只需要将其直接删除即可；

4.当需要取出执行的下一个进程的时候，只需要将就绪队列的队头取出即可；

5.每当出现一个时钟中断，则会将当前执行的进程的剩余可执行时间减1，一旦减到了0，则将其标记为可以被调度的，这样在ISR中的后续部分就会调用schedule函数将这个进程切换出去

如何设计实现“多级反馈队列调度算法”如下：

1.在proc\_struct中添加总共N个多级反馈队列的入口，每个队列都有着各自的优先级，编号越大的队列优先级约低，并且优先级越低的队列上时间片的长度越大，为其上一个优先级队列的两倍；并且在PCB中记录当前进程所处的队列的优先级；

2.处理调度算法初始化的时候需要同时对N个队列进行初始化；

3.在处理将进程加入到就绪进程集合的时候，观察这个进程的时间片有没有使用完，如果使用完了，就将所在队列的优先级调低，加入到优先级低1级的队列中去，如果没有使用完时间片，则加入到当前优先级的队列中去；

4.在同一个优先级的队列内使用时间片轮转算法；

5.在选择下一个执行的进程的时候，有限考虑高优先级的队列中是否存在任务，如果不存在才转而寻找较低优先级的队列；（有可能导致饥饿）

6.从就绪进程集合中删除某一个进程就只需要在对应队列中删除即可；

7.处理时间中断的函数不需要改变；

8.至此完成了多级反馈队列调度算法的具体设计；

练习2:实现StrideScheduling调度算法（需要编码）

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。

1. 在完成编码任务前，需要对先前LAB所完成的代码进行更新，完成PCB的初始化等工作，由于过于琐碎，此处不再赘述；值得一提的是，需要在时钟中断的处理部分删去print\_ticks函数，否则在执行grade脚本的时候，执行到该函数处便会直接panic退出执行；
2. 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；
3. 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；
4. 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
5. 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
6. 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

