# OS第四次实验报告

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、实验环境

虚拟机Ubuntu

三、实验内容

Lab6——练习二：实现 Stride Scheduling 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。

后面的实验文档部分给出了Stride调度算法的大体描述。这里给出Stride调度算法的一些相关的资料（目前网上中文的资料比较欠缺）。

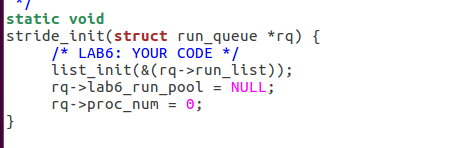
strid-shed paper location1 strid-shed paper location2 也可GOOGLE “Stride Scheduling” 来查找相关资料

执行：make grade。如果所显示的应用程序检测都输出ok，则基本正确。如果只是priority.c过不去，可执行 make runpriority 命令来单独调试它。大致执行结果可看附录。（ 使用的是 qemu-1.0.1 ）。 请在实验报告中简要说明你的设计实现过程。

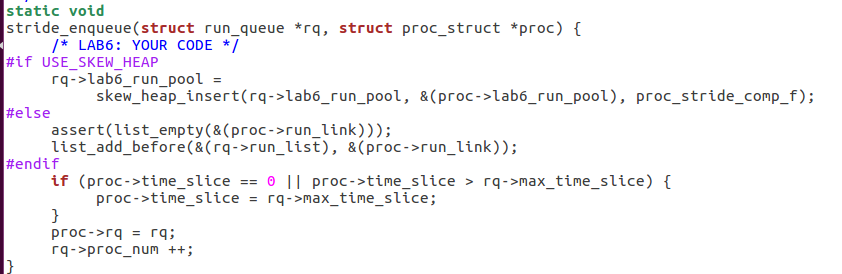
1. 实验过程及结果
2. 打开如下图所示根目录的文件



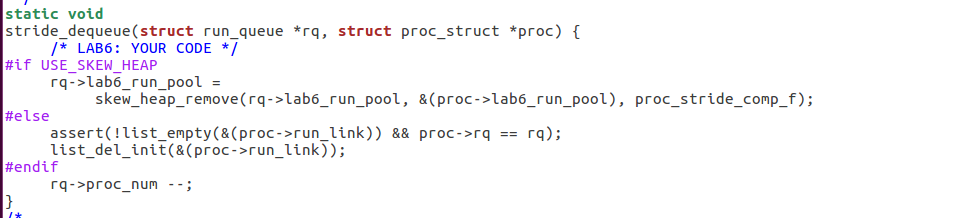
1. 首先是初始化函数stride\_init。   
   开始初始化运行队列，并初始化当前的运行队，然后设置当前运行队列内进程数目为0。



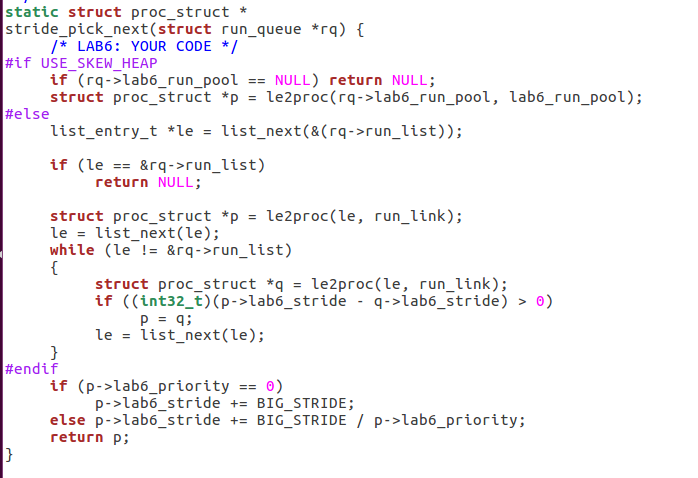
1. 入队函数stride\_enqueue，根据之前对该调度算法的分析，这里函数主要是初始化刚进入运行队列的进程 proc 的stride属性，然后比较队头元素与当前进程的步数大小，选择步数最小的运行，即将其插入放入运行队列中去，这里并未放置在队列头部。最后初始化时间片，然后将运行队列进程数目加一。



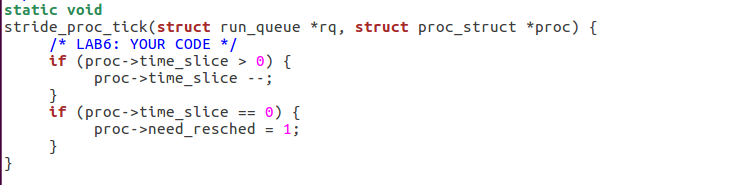
1. 出队函数stride\_dequeue，即完成将一个进程从队列中移除的功能，这里使用了优先队列。最后运行队列数目减一。



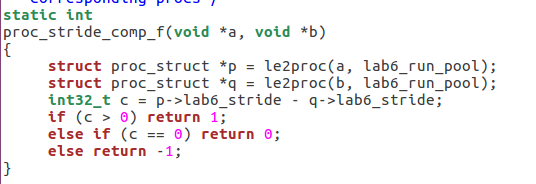
1. 进程的调度函数stride\_pick\_next，观察代码，它的核心是先扫描整个运行队列，返回其中stride值最小的对应进程，然后更新对应进程的stride值，将步长设置为优先级的倒数，如果为0则设置为最大的步长。



1. 时间片函数stride\_proc\_tick，主要工作是检测当前进程是否已用完分配的时间片。如果时间片用完,应该正确设置进程结构的相关标记来引起进程切换。这里和之前实现的RR调度算法一样.



1. 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。



1. 执行make grade

