os实验三

Lab5

练习3: 阅读分析源代码，理解进程执行 fork/exec/wait/exit 的实 现，以及系统调用的实现（不需要编码）

请在实验报告中简要说明你对 fork/exec/wait/exit函数的分析。

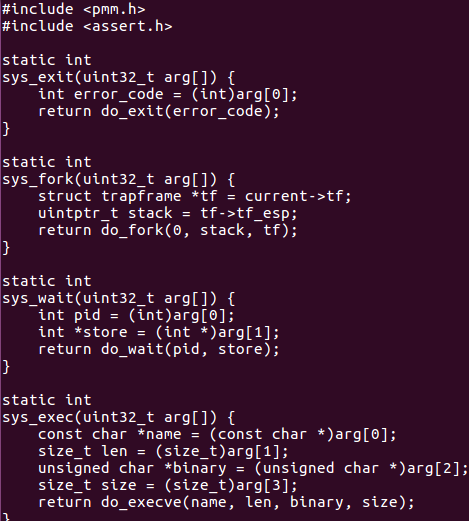
do\_fork：sys\_fork的相关函数。在该函数中，首先要为子进程创建进程控制块，设置好进程控制块中的上下文的中断帧等信息，为子进程创建用户栈、内核栈等。随后通过wakeup\_proc函数将子进程设置为RUNNABLE。之后该函数给父进程返回子进程的pid，给子进程返回0。随后在ucore循环执行进程调度schedule时，就会将子进程考虑进去。

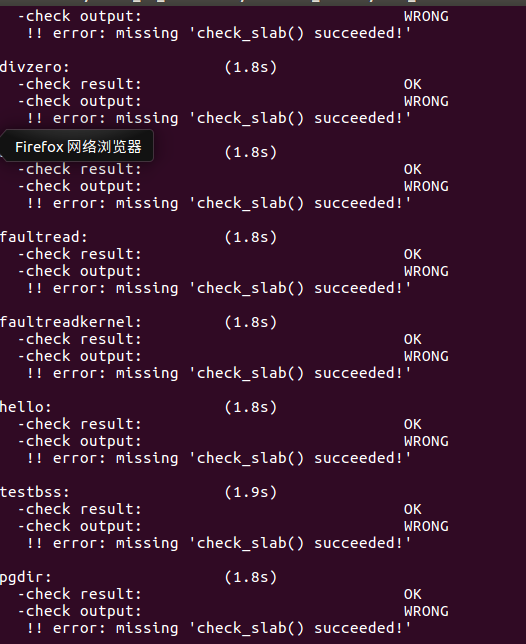
并回答如下问题：

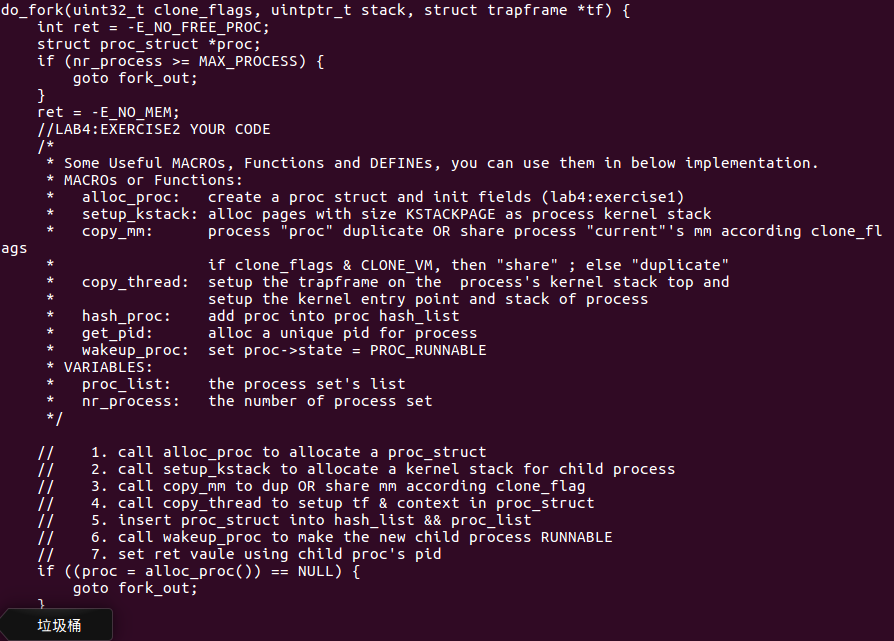
请分析fork/exec/wait/exit在实现中是如何影响进程的执行状态的？

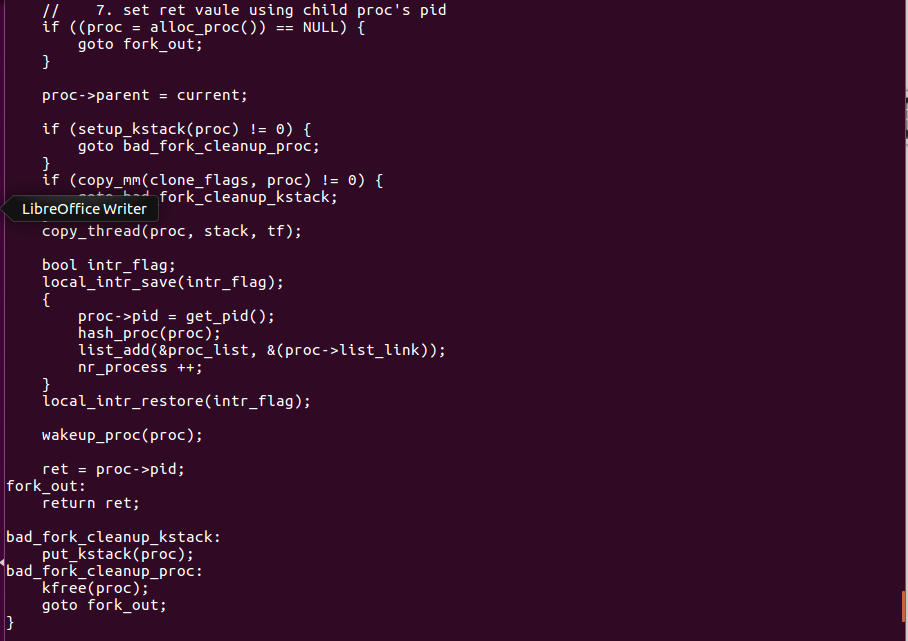
fork 不会影响当前进程的执行状态，但是会将子进程的状态标记为RUNNALB，使得可以在后续的调度中运行起来。

请给出ucore中一个用户态进程的执行状态生命周期图（包执行状态，执行状态之间的变 换关系，以及产生变换的事件或函数调用）。（字符方式画即可）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。（使用的是qemu1.0.1）





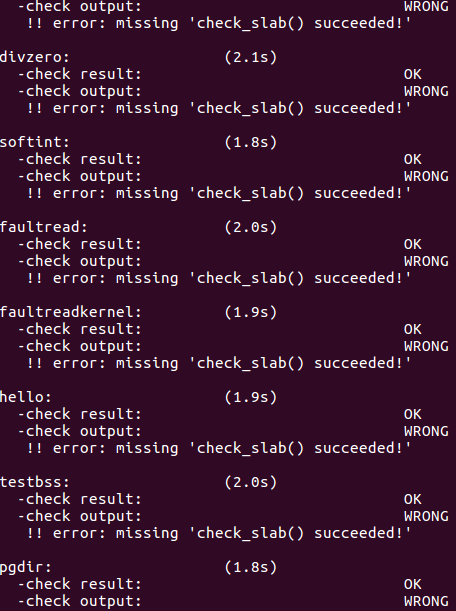




Lab6：

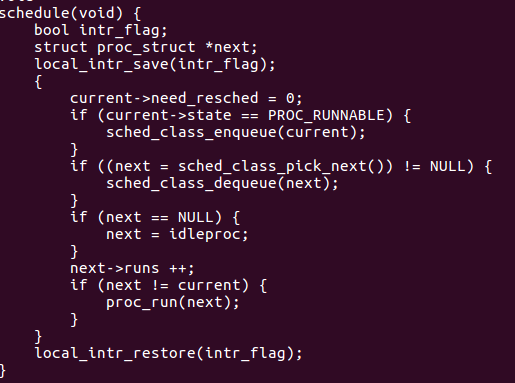
练习1: 使用 Round Robin 调度算法（不需要编码）

完成练习0后，建议大家比较一下（可用kdiff3等文件比较软件）个人完成的lab5和练习0完成 后的刚修改的lab6之间的区别，分析了解lab6采用RR调度算法后的执行过程。执行make grade，大部分测试用例应该通过。但执行priority.c应该过不去。



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，并接合Round Robin 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

分析：通过阅读sched.c中的代码，可以发现实现调度的主要函数schedule实际上调用的是sched\_class\_enqueue、sched\_class\_dequeue、sched\_class\_pick\_next、sched\_class\_proc\_tick和sched\_init这五个函数，在这五个函数内部，通过sched\_class类的实例中的函数指针来调用具体的调度算法。

当调用schedule函数时发生调度。首先清除当前进程需要调度的标记，把当前进程放进运行队列中去。sched\_class\_enqueue函数调用Round Robin算法的RR\_enqueue，也就是把当前进程加入到运行队列的最后，将当前进程的时间片时间重置为最大，更新队列中进程数目。随后，通过Round Robin调度算法选择下一个要运行的进程。具体地，sched\_class\_pick\_next函数调用Round Robin算法的RR\_pick\_next接口，Round Robin算法的RR\_pick\_next将返回运行队列中的第一个进程交给schedule作为下一个要运行的进程。如果可以选出下一个要运行的进程，则将这个进程从运行队列中删除。否则，运行idleproc进程，即返回查询是否有新的进程需要运行。sched\_class\_dequeue函数会调用Round Robin算法的RR\_dequeue函数。

随后通过proc\_run函数切换到新进程并进入新进程执行。proc\_run的具体切换过程在Lab5中已经完成。

此外，在每一次时钟tick的时候，都会调用Round Robin的RR\_proc\_tick函数，将当前进程的所占用的时间片剩余时间减一，当时间片耗尽时，设置为需要调度并等待调度。多级反馈队列算法可以实现在不同的队列间使用不同的调度算法。  
假设进程一共有4个调度优先级，分别为0、1、2、3，其中0位最高优先级，3位最低优先级。为了支持4个不同的优先级，在运行队列中开4个队列，分别命名为rq -> run\_list[0..3]。除此之外，在proc\_struct中加入priority成员表示该进程现在所处的优先级，初始化为0。