**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）

**fork/exec/wait/exit函数的分析**

fork: 系统调用sys\_fork主要是由do\_fork和wakeup\_proc来完成的.

Exec: 调用SYS\_exec系统调用,而当ucore收到此系统调用的时候，则会使用do\_execve()函数来实现.在do\_execve()函数中，它的工作主要分成两部分。1. 首先为加载新的执行码做好用户态内存空间清空准备。2. 载应用程序执行码到当前进程的新创建的用户态虚拟空间中。之后就是调用load\_icode从而使之准备好执行。

Wait:直接看sys\_wait,在这里面是调用do\_wait来完成主要工作。主要工作就是父进程如何完成对子进程的最后回收工作。

Exit：依然是系统调用sys\_exit，

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

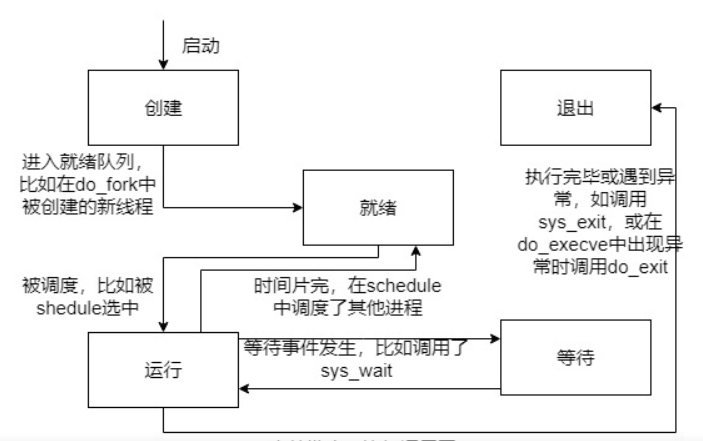
wait系统调用取决于是否存在可以释放资源的子进程。

exit会将当前进程的状态修改为ZOMBIE态，并且会将父进程唤醒，然后主动让出CPU使用权。

其余两个几乎不影响进程的执行状态。

**请给出ucore中一个用户态进程的执行状态生命周期图**

这个借鉴了网上的别人的实验报告。



**Lab6**

**练习1: 使用Round Robin 调度算法。**

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。

**Init**：是初始化环节，初始化rq的进程队列，并将其进程数量置零。

**enqueue**：是一个进程入队的操作：进程队列是一个双向链表，一个进程加入队列的时候，会将其加入到队列的第一位，并给它初始数量的时间片；并更新队列的进程数量。

**dequeue**：从就绪队列中取出这个进程，并将其调用list\_del\_init删除。同时，进程数量减一。

**pick\_next**：通过list\_next函数的调用，会从队尾选择一个进程，代表当前应该去执行的那个进程。如果选不出来有处在就绪状态的进程，那么返回NULL，并将执行权交给内核线程idle，idle的功能是不断调用schedule，直到整个系统出现下一个可以执行的进程。

**proc\_tick**：产生时钟中断的时候，会触发tick函数的调用

最后，是在schedule初始化的时候，需要填写一个初始化信息，那么这里就填上我们所实现的类函数，那么系统就可以按照这个方式去执行了。

**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。**

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，其中0位最高优先级，2是最低优先级。为了支持3个不同的优先级，在运行队列中开3个队列