**实验五：用户进程管理**

**实验六: 调度器**

1. **实验目的**
2. 了解第一个用户进程创建过程
3. 了解系统调用框架的实现机制
4. 了解ucore如何实现系统调用sys\_fork/sys\_exec/sys\_exit/sys\_wait来进行进程管理
5. 理解操作系统的调度管理机制
6. 熟悉 ucore 的系统调度器框架，以及缺省的Round-Robin 调度算法
7. 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
8. **实验内容**

实验4完成了内核线程，但到目前为止，所有的运行都在内核态执行。实验5将创建用户进 程，让用户进程在用户态执行，且在需要ucore支持时，可通过系统调用来让ucore提供服 务。为此需要构造出第一个用户进程，并通过系统调用sys\_fork/sys\_exec/sys\_exit/sys\_wait 来支持运行不同的应用程序，完成对用户进程的执行过程的基本管理。相关原理介绍可看附 录B。

实验五完成了用户进程的管理，可在用户态运行多个进程。但到目前为止，采用的调度策略 是很简单的FIFO调度策略。本次实验，主要是熟悉ucore的系统调度器框架，以及基于此框架 的Round-Robin（RR） 调度算法。然后参考RR调度算法的实现，完成Stride Scheduling调度算法

1. **练习**

对实验报告的要求：

基于markdown格式来完成，以文本方式为主

填写各个基本练习中要求完成的报告内容

完成实验后，请分析ucore\_lab中提供的参考答案，并请在实验报告中说明你的实现与参 考答案的区别

列出你认为本实验中重要的知识点，以及与对应的OS原理中的知识点，并简要说明你对 二者的含义，关系，差异等方面的理解（也可能出现实验中的知识点没有对应的原理知 识点）

列出你认为OS原理中很重要，但在实验中没有对应上的知识点

**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。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？**

答：对fork/exec/wait/exit四个系统调用进行分析：

* fork：在执行了fork系统调用之后，会执行正常的中断处理流程，最终将控制权转移给syscall，之后根据系统调用号执行sys\_fork函数，进一步执行了上文中的do\_fork函数，完成新的进程的进程控制块的初始化、设置、以及将父进程内存中的内容到子进程的内存的复制工作，然后将新创建的进程放入可执行队列（runnable），这样的话在之后就有可能由调度器将子进程运行起来了；
* exec：在执行了exec系统调用之后，会执行正常的中断处理流程，最终将控制权转移给syscall，之后根据系统调用号执行sys\_exec函数，进一步执行了上文中的do\_execve函数，在该函数中，会对内存空间进行清空，然后将新的要执行的程序加载到内存中，然后设置好中断帧，使得最终中断返回之后可以跳转到指定的应用程序的入口处，就可以正确执了；
* wait：在执行了wait系统调用之后，会执行正常的中断处理流程，最终将控制权转移给syscall，之后根据系统调用号执行sys\_wait函数，进一步执行了的do\_wait函数，在这个函数中，将搜索是否指定进程存在着处于ZOMBIE态的子进程，如果有的话直接将其占用的资源释放掉即可；如果找不到这种子进程，则将当前进程的状态改成SLEEPING态，并且标记为等待ZOMBIE态的子进程，然后调用schedule函数将其当前线程从CPU占用中切换出去，直到有对应的子进程结束来唤醒这个进程为止；
* exit：在执行了exit系统调用之后，会执行正常的中断处理流程，最终将控制权转移给syscall，之后根据系统调用号执行sys\_exit函数，进一步执行了的do\_exit函数，首先将释放当前进程的大多数资源，然后将其标记为ZOMBIE态，然后调用wakeup\_proc函数将其父进程唤醒（如果父进程执行了wait进入SLEEPING态的话），然后调用schedule函数，让出CPU资源，等待父进程进一步完成其所有资源的回收；

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；
* exec不会影响当前进程的执行状态，但是会修改当前进程中执行的程序；
* wait系统调用取决于是否存在可以释放资源（ZOMBIE）的子进程，如果有的话不会发生状态的改变，如果没有的话会将当前进程置为SLEEPING态，等待执行了exit的子进程将其唤醒；
* exit会将当前进程的状态修改为ZOMBIE态，并且会将父进程唤醒（修改为RUNNABLE），然后主动让出CPU使用权；

lab6练习一：



