Lab5、Lab6实验报告

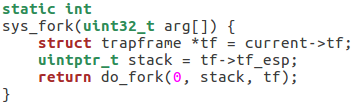
班级：计算机科学与技术2班

姓名：许金曼

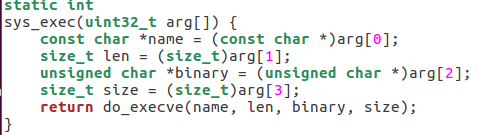
学号：171491220

Lab5 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。

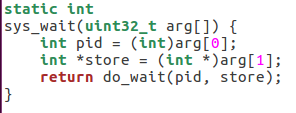
1. 请分析fork/exec/wait/exit在实现中是如何影响进程的执行状态的？



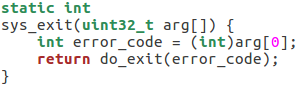
sys\_fork将创建新的子进程，将子进程的状态由UNINIT状态变为RUNNABLE状态，不改变父进程的状态。sys\_fork主要是由do\_fork完成的。



sys\_exec完成用户进程的创建工作，同时使用户进程进入执行，不改变进程状态。sys\_exec是通过do\_execve实现的。



sys\_wait完成子进程资源回收，如果有已经结束的子进程或者没有子进程，那么调用会立刻结束，不影响进程状态；否则，进程需要等待子进程结束，进程从RUNNING状态变为SLEEPING态。调用的功能主要由do\_wait实现。



sys\_exit回收当前所占的大部分内存资源，并通知父进程完成最后的回收，进程从RUNNING态变为ZOMBIE态，通过do\_exit实现。

2. 请给出ucore中一个用户态进程的执行状态生命周期图

创建

就绪

运行

等待

退出

进程调度

进入就绪队列

运行结束

时间片用完

exit（）

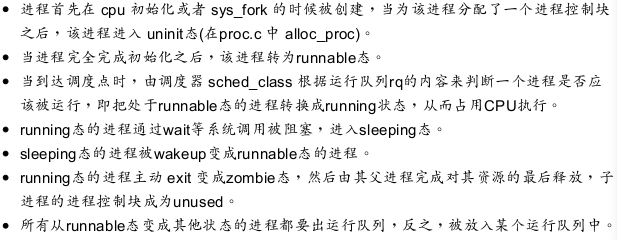
事件发生

事件请求

exit()

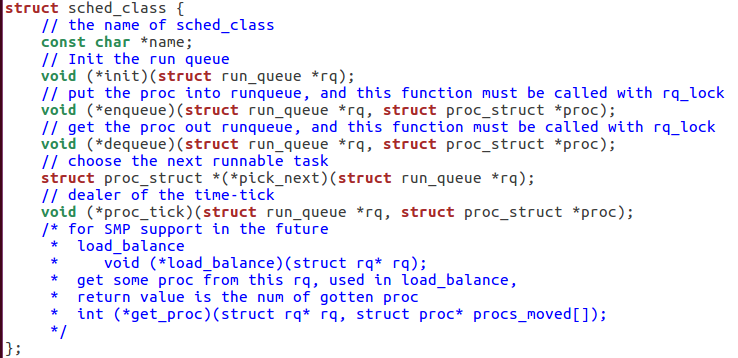
wait()

fork()

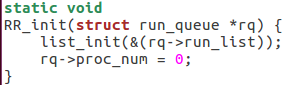


Lab6 使用Round Robin调度算法

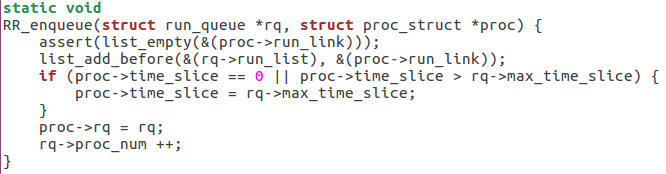
1. 请理解并分析sched\_class中各个函数指针的用法，并结合Round Robin调度算法描述ucore的调度执行过程



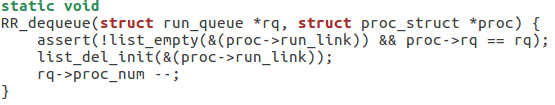
Round Robin算法的实现：



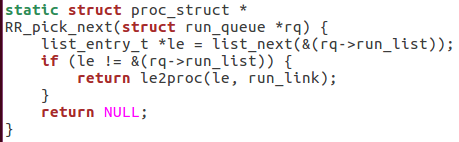
RR\_init完成对进程队列的初始化，将进程数置0



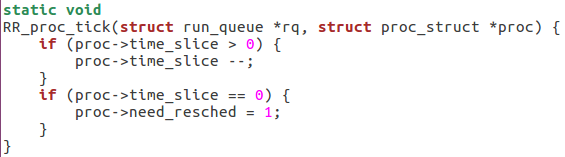
RR\_enqueue完成一个进程入队的操作，将进程的进程控制块加入队列末尾，如果进程控制块的时间片为0或者进程的时间片大于分配给进程的最大时间片，则把它置为max\_time\_slice；并更新队列的进程数量



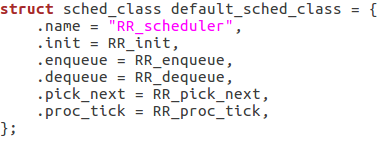
RR\_dequeue将进程从就绪队列中移除，先确定当前进程控制块指针非空且处于就绪队列，就将其从就绪队列中删除，将就绪进程数减1。



RR\_pick\_next选择下一个调度进程，调用list\_next函数，选择一个进程，表示当前要去执行该进程，如果选不出来则返回NULL。



每次时间片到时时，时间片数量减1；如果时间片用完了，则将need\_resched置为1，表示它必须放弃占有CPU，需要调度别的进程来执行。



Schedule初始化的时候，需要填写一个初始化信息

2. 简要说明如何设计实现“多级反馈队列调度算法”

设有N个队列，其中各个队列的优先级是不一样的，一般来说Q1>Q2>…>QN。

进程在进入待调度的队列等待时，首先进入优先级最高的Q1等待。

首先调度优先级高的队列中的进程，如果高优先级的队列中已经没有进程，则调度次优先级队列中的进程。

对于同一个队列中的各个进程，按照FCFS分配时间片调度。比如在Q1队列的时间片为N，那么Q1中的作业在经历了N个时间片后，若还没有完成，则进入Q2队列等待，若Q2的时间片用完作业还不能完成，一直进入下一级队列，直至完成。

在最后一个队列QN中的各个进程，按照时间片轮转分配时间片调度。