**实验四**

**171491125 吴俊**

1. **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：**

**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？ 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）**

**答：**

**分析：**

1.fork函数是通过do \_ fork实现的，在每次需要创建进程的时候，都会调用它来创建一个新的进程。

2.然后在通过exec通过do \_ execve函数实现。它将内容装入的刚刚创建的进程中。它可以取代调用进程的内容，当fork完后调用exec便会进行新进程的进行，这也是为何fork复制出来的子进程执行之后与父进程不一样的根本原因所在。

3.wait通过do \_ wait来完成。主要完成了等待指定的一个或任意一个处于僵死状态的子进程，并且释放掉其占用的空间。

4.exit通过do \_ exit来完成。主要完成了退出当前进程并回收资源，然后唤醒其父进程的工作。

**问题：**

当想创建新的进程时，系统便会调用fork函数，复制一个和父进程一样的进程。之后会使用exec取代那个内容，所以子进程和父进程便有区别了。当该进程所需的资源都已经准备好了，便会进入就绪状态。当给它分配了处理机资源后，它便会进入到运行状态。若在时间片内完成该进程，便会通过exit函数来回收资源。若时间片用光或是有高优先级进程已经处于就绪状态（可抢占的），便会将现在的进程通过wait函数将其送到等待队列。这时也会使用wait函数收回其原来占用的资源。我认为，只要有进程释放了处理机资源，就需要使用exit函数。



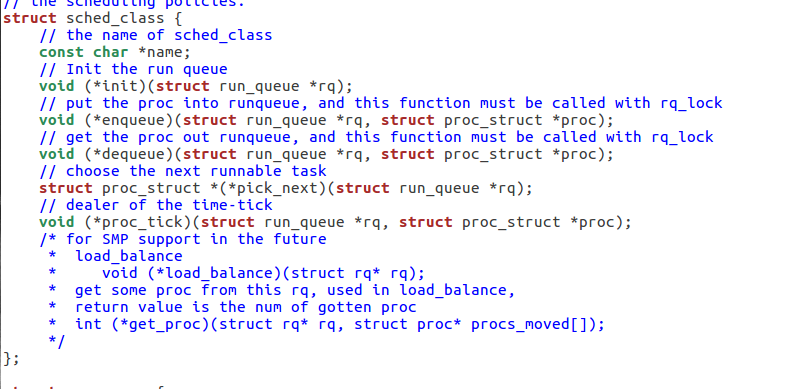
**2.练习1:使用Round Robin调度算法（不需要编码）请在实验报告中完成：**

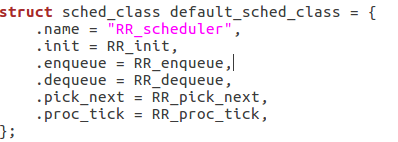
1. **请理解并分析sched\_calss中各个函数指针的用法。**

**答：**

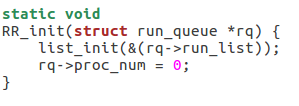
**指针用法：**

根据代码可知：

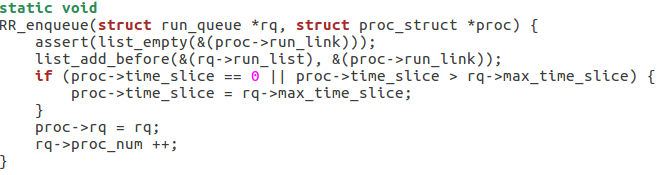




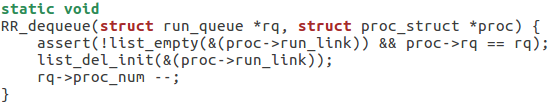
**函数功能：**



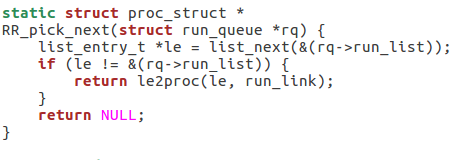
* init：初始化进程队列（RR\_init函数），初始化进程队列，用于初始化。



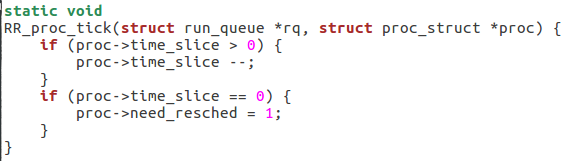
* enqueue：将进程加入就绪队列（RR\_enqueue函数）。它把进程的进程控制块指针放入到rq队列末尾，且如果进程控制块的时间片为0或者进程的时间片大于分配给进程的最大时间片，则需要把它重置为max\_time\_slice。然后在依次调整rq和rq的进程数目加一。



* dequeue：将进程从就绪队列中移除（RR\_dequeue函数）。先确定当前进程控制块指针非空并且进程在就绪队列中，然后将进程控制块指针从就绪队列中删除，最后将就绪进程数减一。

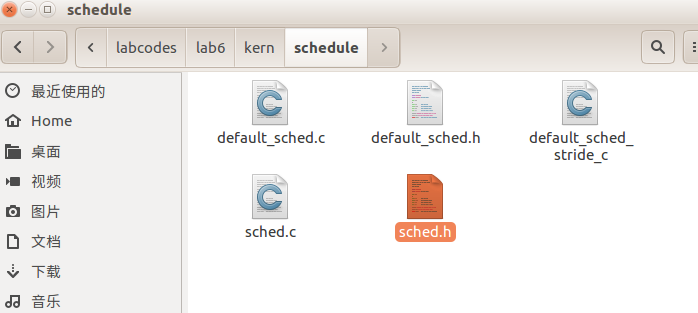


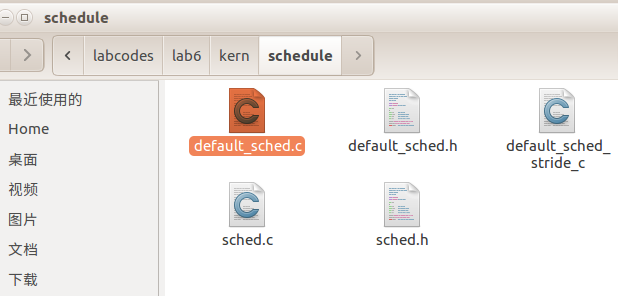
* pick\_next函数：选择下一调度进程（RR\_pick\_next函数）先选取就绪进程队列rq中的队头队列元素，并把队列元素转换成进程控制块指针。最后返回就绪进程。



* proc\_tick函数：时间片（RR\_proc\_tick函数）即每一次时间片到时的时候，当前执行进程的时间片time\_slice便减一。如果time\_slice降到零，则设置此进程成员变量need\_resched标识为1，这样在下一次中断来后执行trap函数时，会由于当前进程程成员变量need\_resched标识为1而执行schedule函数，从而把当前执行进程放回就绪队列末尾，而从就绪队列头取出在就绪队列上等待时间最久的那个就绪进程执行。

位置：





1. **并接合RoundRobin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。**

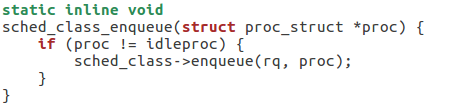
**答：**

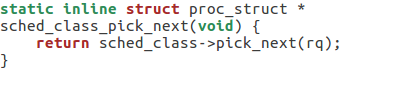
**时间片轮转算法：**

在时间片轮转算法中，系统根据先来先服务策略，将所有的就绪进程排成一个就绪队列，并可设置每隔一定时间间隔及产生一次中断，激活系统中的进程调度程序，完成一次调度，将CPU分给对手进程，令其执行当该进程的时间片耗尽或运行完毕时，系统再次将CPU分给新的队首进程，令其执行。当该进程的时间片耗尽或运行完毕时，系统再次将CPU分配给新的队首进程（或新到达的紧迫进程），由此可保证就绪队列中的所有进程在一个确定的时间段内都能获得一次CPU执行。在时间片轮转算法中，切换分为两种情况，1.若一个时间片尚未用完，正在运行的进程便已经完成，就立即激活调度程序，将它从就绪队列中删，除再调度就绪队列中队首的进程运行，并启动一个新的时间片。2.在一个时间片用完时，计时器中断处理程序被激活，如果进程尚未运行完毕，调度程序将把它送往就绪队列的末尾。

**ucore的调度过程：**

ucore通过schedule()函数进行调度，在这个函数中，当需要进行线程的切换时，他会先将当前的线程通过sched\_class\_enqueue()函数加入队列，然后通过sched\_class\_pick\_next()函数选择下一个要执行的线程并将其从队列中删除。





**多级反馈队列调度算法：**

1. 设置多个就绪队列。在系统中设置多个就绪队列,并为每个队列赋予不同的优先级。第一个队列的优先级最高,第二个次之,其余队列的优先级逐个降低。该算法为不同队列中的进程所赋予的执行时间片的大小也各不相同,在优先级愈高的队列中,其时间片就愈小。例如第二个队列的时间片要比第一个的时间片长一倍,……,第i+1个队列的时间片要比第i个的时间片长一倍。
2. 每个队列都采用先来先服务算法，当新进程进入内存后，首先将它放入第一队列的末尾，按先来先服务原则等待调度。当轮到该进程执行时，如果他能在该时间片内完成，便可撤离系统，否则，即他在一个时间片结束时尚未完成，调度程序将其转入第二队列的末尾等待调度，如果他在第二队列中运行一个时间片后仍未完成，再依次将它放入第三队列，以此类推，当进程，最后被降到第n队列后，再第n队列中便采取按RR方式运行。
3. 按队列优先级调度，调度程序首先调度最高优先级队列中的诸进程运行。仅当第一队列空闲时才调度第二队列中的进程运行，换言之，仅当第1~（i-1）所有队列均空时，才会调度第i队列中的进程运行。如果处理机正在第i队列中为某进程服务时，又有新进程进入任一优先级较高的队列，此时，须立即把正在运行的进程放回到第i队列的末尾，而把处理机分配给新到的高优先级进程。