Lab5 练习三：

理解进程执行 fork/exec/wait/exit 的实 现，以及系统调用的实现

Fork的实现：

1分配并初始化进程控制块(alloc\_proc 函数);

2、分配并初始化内核栈(setup\_stack 函数);

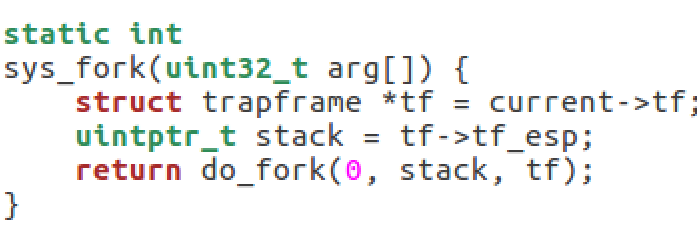
3、根据 clone\_flag标志复制或共享进程内存管理结构(copy\_mm 函数);

4、设置进程在内核(将来也包括用户态)正常运行和调度所需的中断帧和执行上下文(copy\_thread 函数);

5、把设置好的进程控制块放入hash\_list 和 proc\_list 两个全局进程链表中;

6、把进程状态设置为“就绪”态;

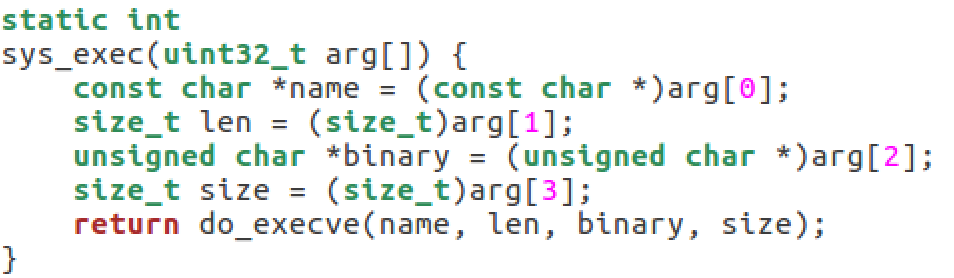
7、设置返回码为子进程的 id 号。



Exec的实现：

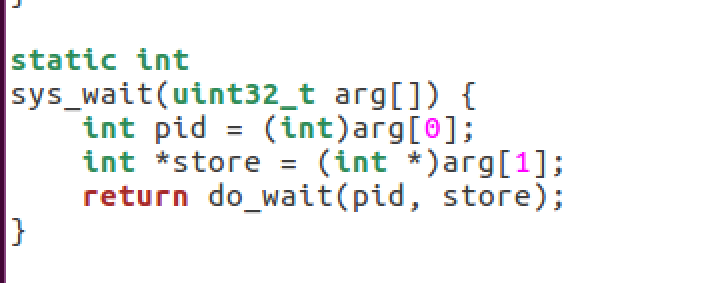
1.为加载新的执行码做好用户态内存空间清空准备

2.加载用程序执行码到当前进程的新创建的用户态虚拟空间中



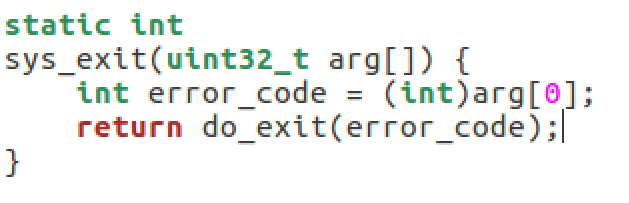
Wait的实现：

用执行wait功能时，会调用SYS\_wait，该系统调用功能由do\_wait实现。完成对子程序的最后收取工作



Exit的实现：

回收当前进程所占的大部分内存资源,并通知父进程完成最后的回收工作



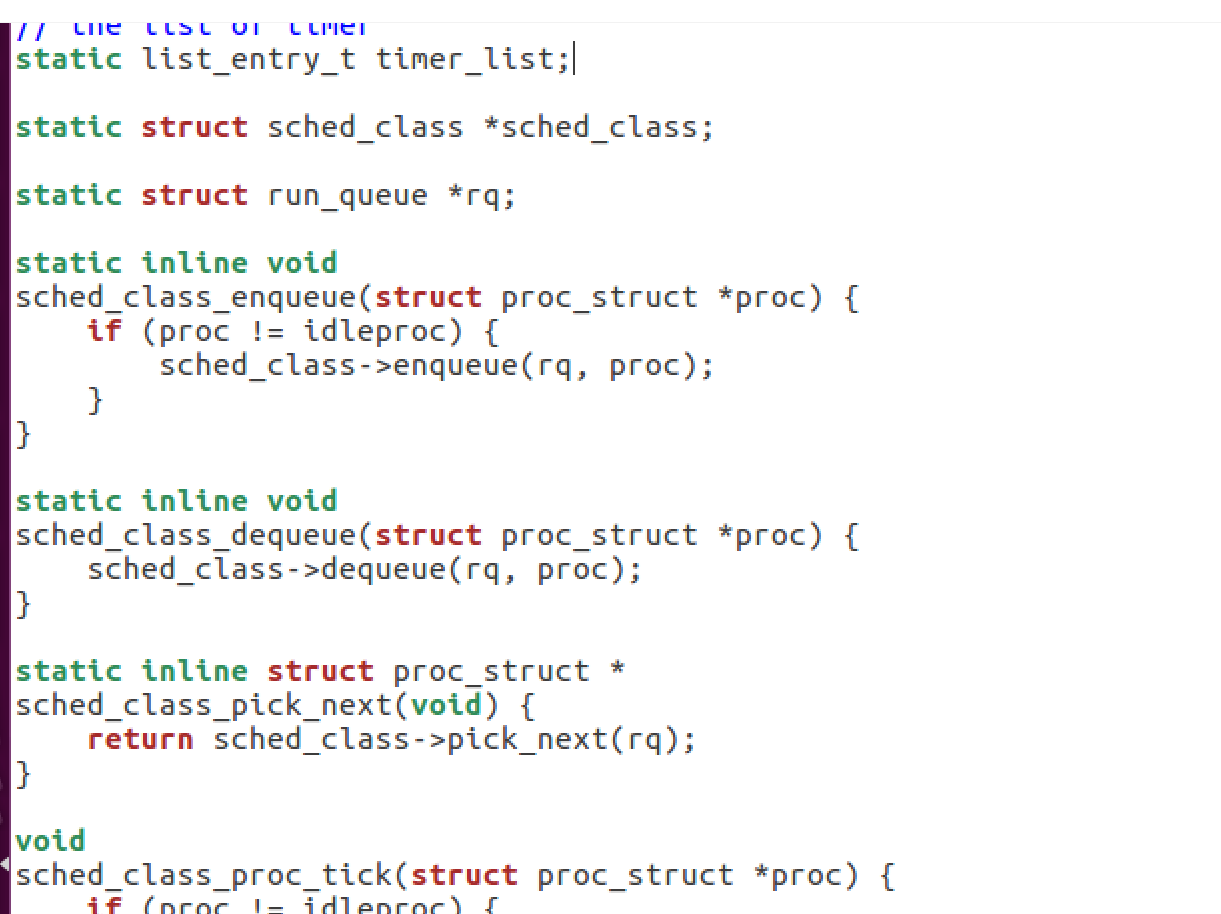
Lab6

实验目的：理解操作系统的调度管理机制

熟悉 ucore 的系统调度器框架，以及缺省的Round-Robin 调度算法

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

练习：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



1. init函数：完成对进程队列的初始化
2. enqueue函数：该函数的功能为将指定的进程的状态置成RUNNABLE，并且放入调用算法中的可执行队列中，被封装成sched\_class\_enqueue函数，可以发现这个函数仅在wakeup\_proc和schedule函数中被调用，前者为将某个不是RUNNABLE的进程加入可执行队列，而后者是将正在执行的进程换出到可执行队列中去；
3. dequeue函数：该函数的功能为将某个在队列中的进程取出，其封装函数sched\_class\_dequeue仅在schedule中被调用，表示将调度算法选择的进程从等待的可执行的进程队列中取出进行执行；
4. pick\_next函数：该函数的封装函数同样仅在schedule中被调用，功能为选择要执行的下个进程；
5. proc\_tick函数：该函数表示每次时钟中断的时候应当调用的调度算法的功能，仅在进行时间中断的ISR中调用；