操作系统实验二

一．实验目的

1.了解内核线程创建/执行的管理过程

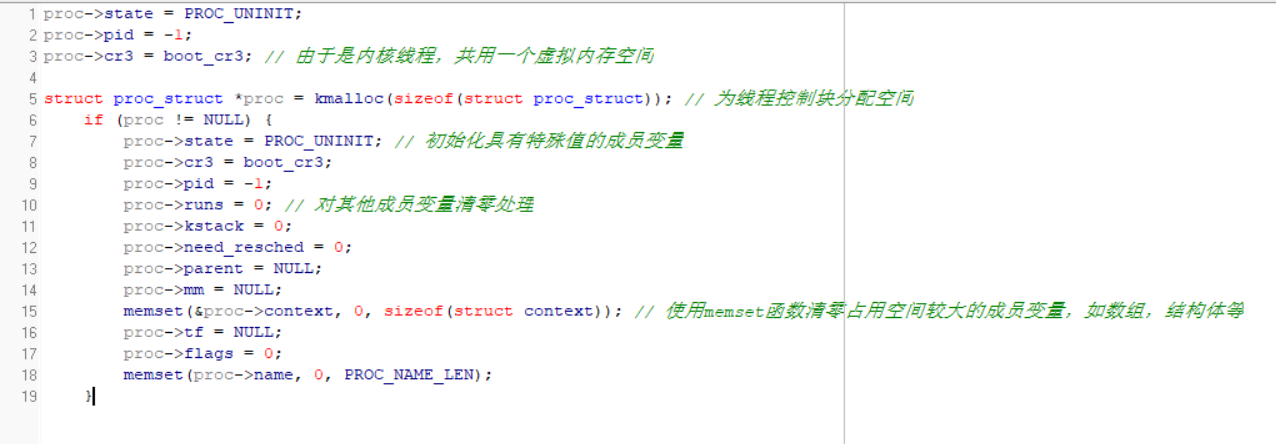
2.了解内核线程的切换和基本调度过程

练习1：分配并初始化一个进程控制块（需要编码）

alloc\_proc函数（位于kern/process/proc.c中）负责分配并返回一个新的structproc\_struct结 构，用于存储新建立的内核线程的管理信息。ucore需要对这个结构进行最基本的初始化，你 需要完成这个初始化过程。

【提示】在alloc\_proc函数的实现中，需要初始化的proc\_struct结构中的成员变量至少包 括：state/pid/runs/kstack/need\_resched/parent/mm/context/tf/cr3/flags/name。

请在实验报告中简要说明你的设计实现过程。



请回答如下问题：

请说明proc\_struct中 struct context context 和 struct trapframe \*tf 成员变量含义和在 本实验中的作用是啥？（提示通过看代码和编程调试可以判断出来）

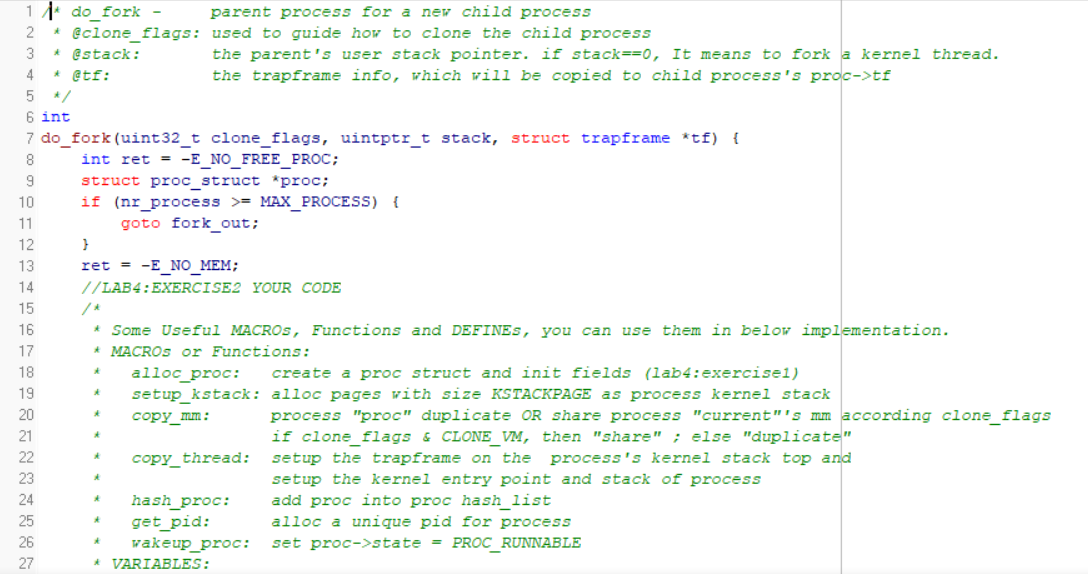
答：context变量的意义是保存的线程运行的上下文信息，其作用是内核线程之间进行切换的时候，将原先的线程运行的上下文保存下来。tf变量的意义是指针。tf变量的作用在于在构造出了新的线程的时候，如果要将控制权交给这个线程，是使用中断返回的方式进行的，因此需要构造出一个伪造的中断返回现场，也就是trap frame，使得可以正确地将控制权转交给新的线程。

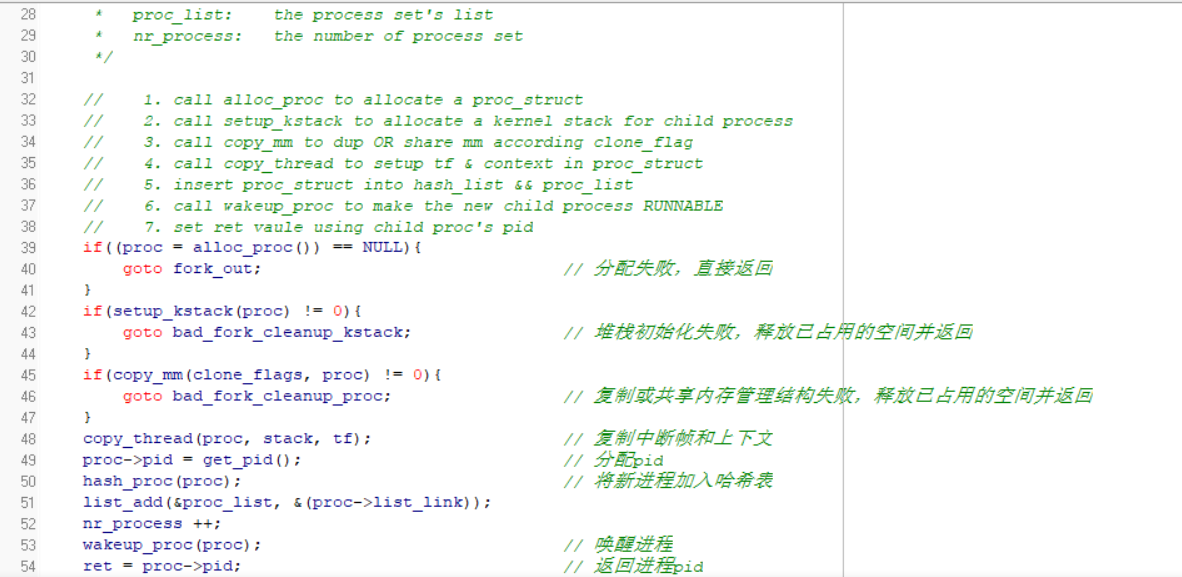
练习2：为新创建的内核线程分配资源（需要编码）

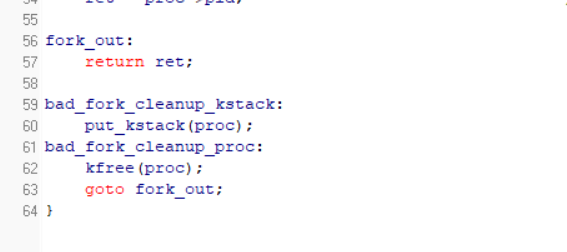
创建一个内核线程需要分配和设置好很多资源。kernel\_thread函数通过调用do\_fork函数完成 具体内核线程的创建工作。do\_kernel函数会调用alloc\_proc函数来分配并初始化一个进程控 制块，但alloc\_proc只是找到了一小块内存用以记录进程的必要信息，并没有实际分配这些资 源。ucore一般通过do\_fork实际创建新的内核线程。do\_fork的作用是，创建当前内核线程的 一个副本，它们的执行上下文、代码、数据都一样，但是存储位置不同。在这个过程中，需 要给新内核线程分配资源，并且复制原进程的状态。你需要完成在kern/process/proc.c中的 do\_fork函数中的处理过程。它的大致执行步骤包括：

调用alloc\_proc，首先获得一块用户信息块。 为进程分配一个内核栈。 复制原进程的内存管理信息到新进程（但内核线程不必做此事） 复制原进程上下文到新进程 将新进程添加到进程列表 唤醒新进程 返回新进程号

请在实验报告中简要说明你的设计实现过程。







请回答如下问题：

请说明ucore是否做到给每个新fork的线程一个唯一的id？请说明你的分析和理由。

答：可以。ucore中为新的fork的线程分配pid的函数为get\_pid，

* 在该函数中使用到了两个静态的局部变量next\_safe和last\_pid，根据命名推测，在每次进入get\_pid函数的时候，这两个变量的数值之间的取值均是合法的pid（也就是说没有被使用过），这样的话，如果有严格的next\_safe > last\_pid + 1，那么久可以直接取last\_pid + 1作为新的pid（需要last\_pid没有超出MAX\_PID从而变成1）；
* 如果在进入函数的时候，这两个变量之后没有合法的取值，也就是说next\_safe > last\_pid + 1不成立，那么进入循环，在循环之中首先通过if (proc->pid == last\_pid)这一分支确保了不存在任何进程的pid与last\_pid重合，然后再通过if (proc->pid > last\_pid && next\_safe > proc->pid)这一判断语句保证了不存在任何已经存在的pid满足：last\_pid<pid<next\_safe，这样就确保了最后能够找到这么一个满足条件的区间，获得合法的pid；

练习3：阅读代码，理解 proc\_run 函数和它调用的函数如何完成 进程切换的。（无编码工作）

请在实验报告中简要说明你对proc\_run函数的分析。并回答如下问题：

在本实验的执行过程中，创建且运行了几个内核线程？ 语句 local\_intr\_save(intr\_flag);....local\_intr\_restore(intr\_flag); 在这里有何作用?请说明理由

答：总共创建了两个内核线程，分别为：

* idleproc: 最初的内核线程，在完成新的内核线程的创建以及各种初始化工作之后，进入死循环，用于调度其他线程；
* initproc: 被创建用于打印"Hello World"的线程；

该语句的作用是：

* 该语句的左右是关闭中断，使得在这个语句块内的内容不会被中断打断，是一个原子操作；这就使得某些关键的代码不会被打断，从而不会一起不必要的错误；比如说在proc\_run函数中，将current指向了要切换到的线程，但是此时还没有真正将控制权转移过去，如果在这个时候出现中断打断这些操作，就会出现current中保存的并不是正在运行的线程的中断控制块，从而出现错误。