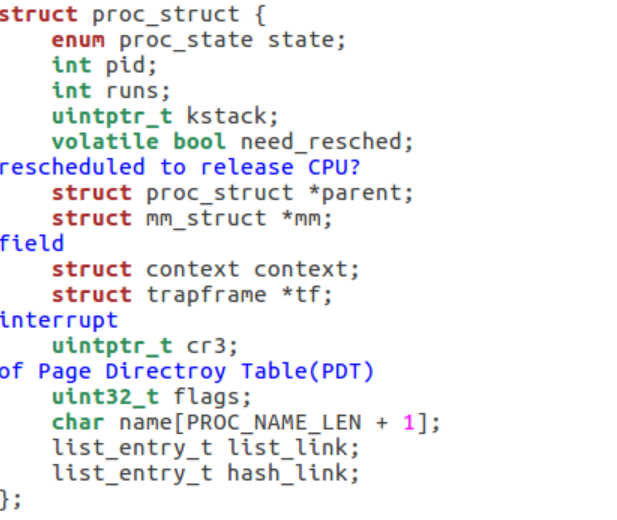
练习1：分配并初始化一个进程控制块

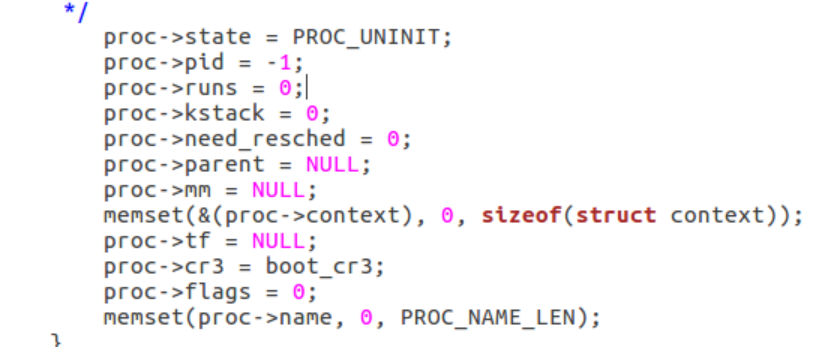
alloc\_proc函数（位于kern/process/proc.c中）负责分配并返回一个新的struct proc\_struct结构，用于存储新建立的内核线程 的管理信息。ucore需要对这个结构进行最基本的初始化，你需要完成这个初始化过程。

Alloc\_proc函数定义了一个结构体如下：

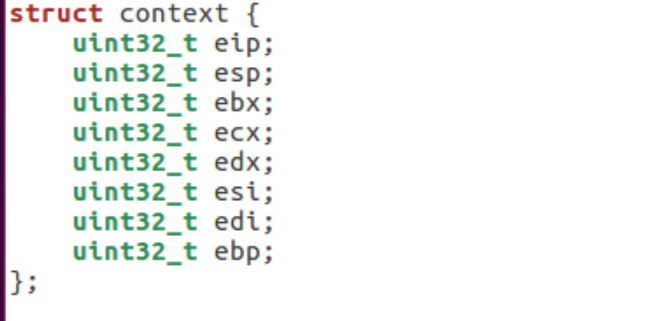


初始化过程对一个该结构对象初始化，分配内核线程。

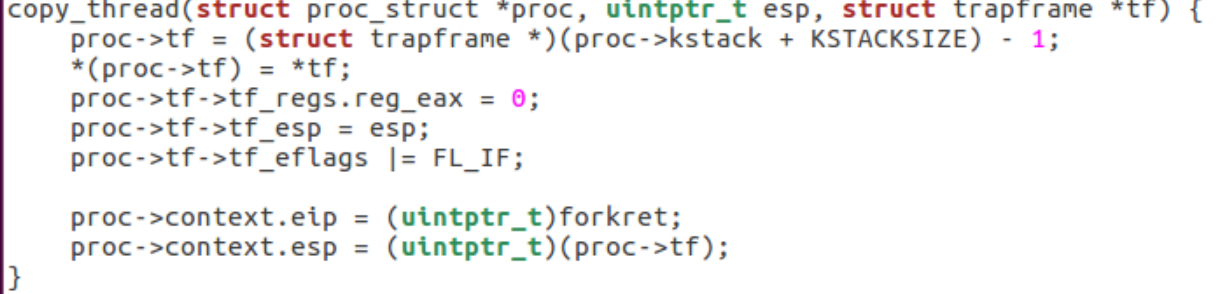
主要的初始化过程如下：



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？



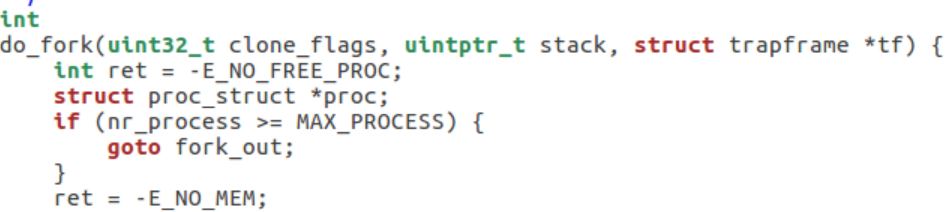
Struct context出现在proc.c中



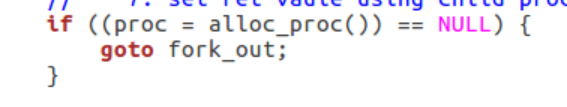
根据函数可知 context用于将原先的线程运行的上下文保存

Copy\_thread也对tf’进行了设置。将context变量的esp设置为tf变量的地址，将eip设置为forkret的指针

练习2：为新创建的内核线程分配资源



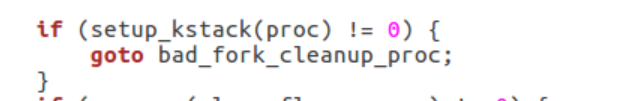
调用alloc\_proc，首先获得一块用户信息块。



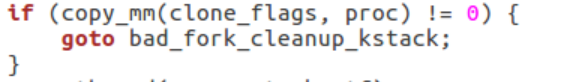
将子进程的父节点设置为当前进程



调用setup\_stack()函数，为内核分配一个进程



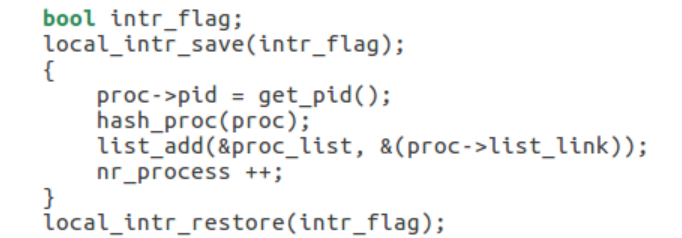
调用copy\_mm()函数，复制父进程内存信息到子进程



调用copy\_thread()函数，复制父进程的中断帧和上下文信息



将新进程加到进程列表

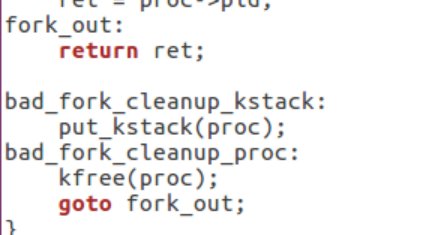


唤醒子进程

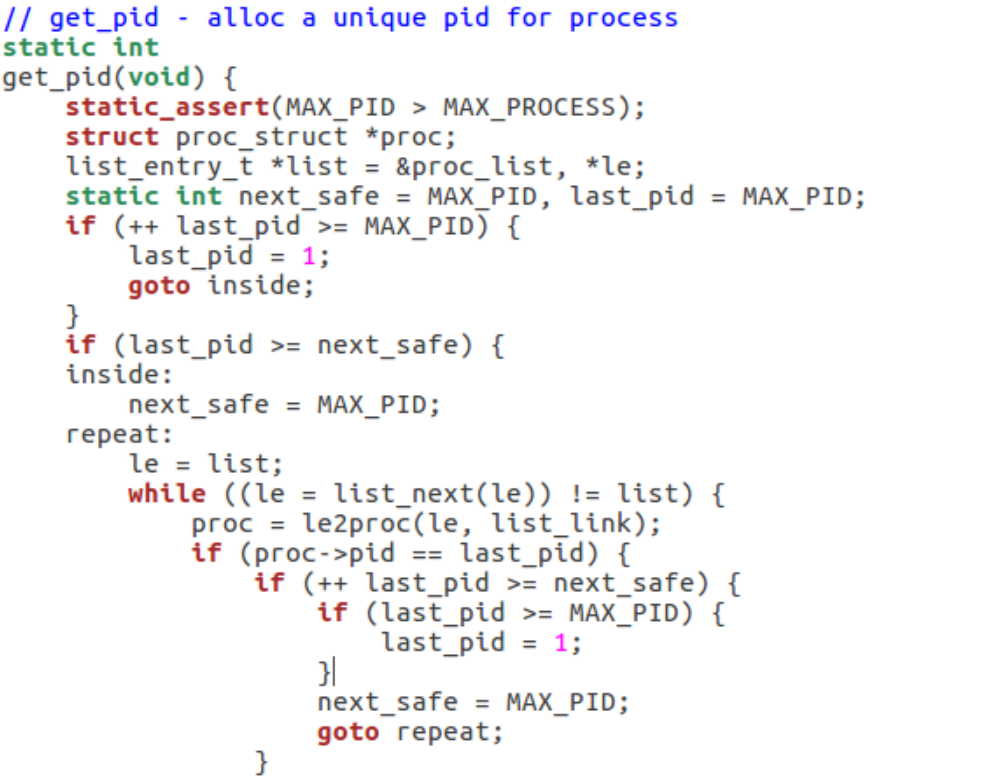


返回子进程pid





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。

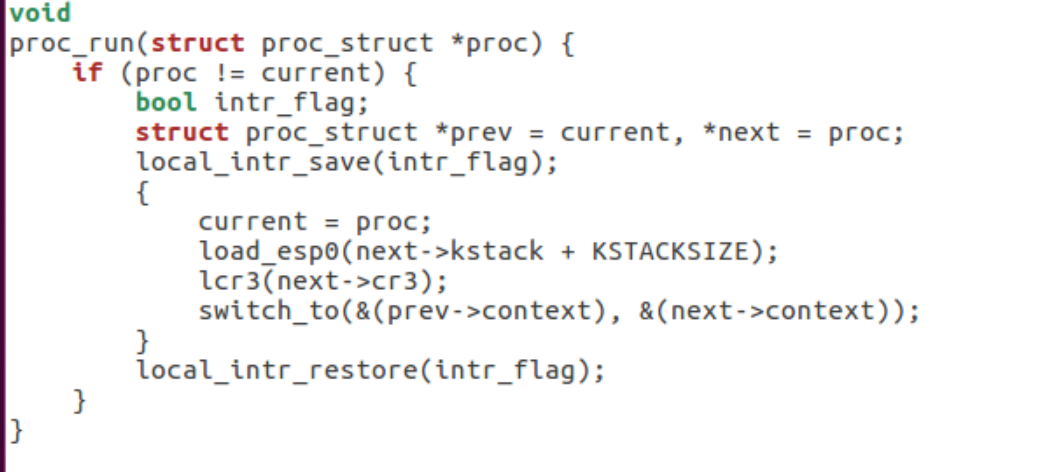


可以。ucore中为新的fork的线程分配pid的函数为get\_pid

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。



proc\_run中首先进行了TSS以及cr3寄存器的设置，然后调用到了swtich\_to函数来切换线程，根据上文中对switch\_to函数的分析可以知道，在调用该函数之后，首先会恢复要运行的线程的上下文，然后由于恢复的上下文中已经将返回地址（copy\_thread函数中完成）修改成了forkret函数的地址(如果这个线程是第一运行的话，否则就是切换到这个线程被切换出来的地址)，也就是会跳转到这个函数，最后进一步跳转到了\_\_trapsret函数，调用iret最终将控制权切换到新的线程；作者：AmadeusChan  
在本实验的执行过程中，创建且运行了几个内核线程？

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；

语句 local\_intr\_save(intr\_flag);....local\_intr\_restore(intr\_flag); 在这里有何作用?

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；

Make qemu

