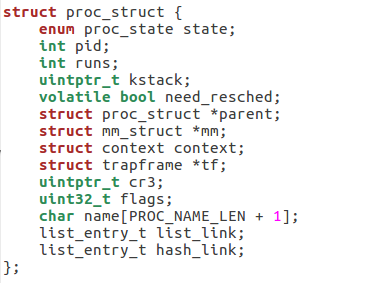
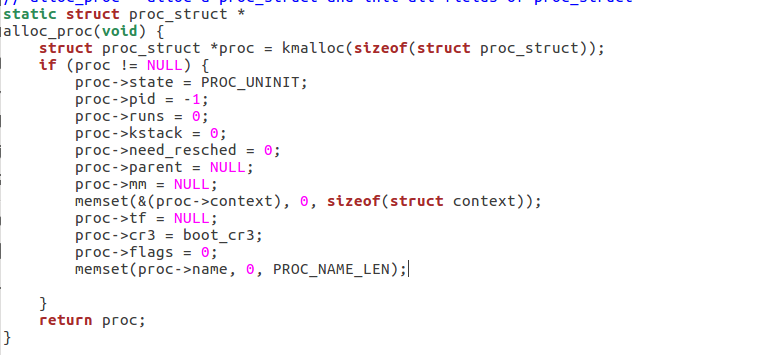
## 练习1，分配并初始化一个进程控制块

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,编码完成这个初始化过程；

首先在pro.h中定义定义进程控制块的数据结构：



然后在pro.c中申请空间，大小为以上定义的数据结构的大小，并对数据成员进行赋值，开始初始化进程控制块



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?

context该结构在定义的时候都是寄存器号，在文件中查找有关该结构的代码，发现只有switch.c中也用了这个数据结构，根据Swtich中代码的语义，可以确定context变量的意义就在于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。

同样在代码中寻找对tf变量进行了定义的地方，最后可以发现在copy\_thread函数中对tf进行了设置，在这个函数中，同时也对context变量的esp和eip进行了设置，前者设置为tf变量的地址、后者设置为forkret这个函数的指针，接下来观察forkret函数，发现这个函数最终调用了\_\_trapret进行中断返回。 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trapframe，使得可以正确地将控制权转交给新的线程；具体切换到新的线程的做法为，调用switch\_to函数，然后在该函数中进行函数返回，直接跳转到forkret函数，最终进行中断返回函数\_\_trapret，之后便可以根据tf中构造的中断返回地址，切换到新的线程了；

## 练习2:为新创建的内核线程分配资源(需要编码)

在pro.c中找到do\_fork函数，对其进行编码：

图片包含 屏幕截图

描述已自动生成

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。

新进程的id是通过get\_pid来获得的，通过查看该函数代码，发现它通过last\_pid和next\_safe两个变量来取得一个正确的id，我们可以发现这样进出比较和++处理之后，返回的last\_

Pid都是唯一的没有重复的。

## 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

对proc\_run函数的分析如下：

图片包含 屏幕截图

描述已自动生成

在本实验中共运行了两个线程，两个函数的作用是关闭中断。

运行过程：

图片包含 文字, 屏幕截图

描述已自动生成