练习**0**：填写已有实验

本实验依赖实验1/2/3。请把你做的实验1/2/3的代码填入本实验中代码中有“LAB1”,“LAB2”,“LAB3”的注释相应部分。

练习**1**：分配并初始化一个进程控制块（需要编码）

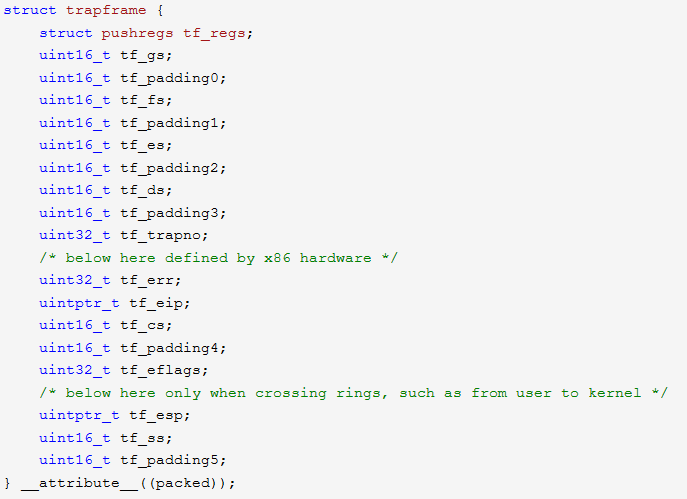
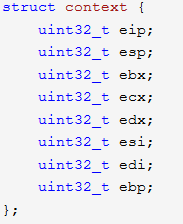
alloc\_proc函数（位于kern/process/proc.c中）负责分配并返回一个新的struct proc\_struct结构，用于存储新建立的内核线程

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。

【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的作用：进程的上下文，用于进程切换。

\*tf的作用：指向中断前的指针，当进程从用户空间跳到内核空间时，指针指向进程在被中断前的状态。当内核需要跳回用户空间时，恢复让进程继续执行的各寄存器值。

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：首先，本实验不提供线程释放的功能，意味着pid只分配不回收。当fork的线程总数小于MAX\_PID时，每个线程的pid是唯一的。当fork的线程总数大于MAX\_PID时，后面fork的线程的pid可能与前面的线程重复。

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。

答：

1.首先判断要切换到的进程是不是当前进程，若是则不需进行任何处理。

2.调用local\_intr\_save和local\_intr\_restore函数去使能中断，避免在进程切换过程中出现中断。

3.更新current进程为proc

4.更新任务状态段的esp0的值

5.重新加载cr3寄存器，使页目录表更新为新进程的页目录表

6.上下文切换，把当前进程的当前各寄存器的值保存在其proc\_struct结构体的context变量中，再把要切换到的进程的proc\_struct结构体的context变量加载到各寄存器。

7.完成上下文切换后，CPU会根据eip寄存器的值找到下一条指令的地址并执行。根据copy\_thread函数可知eip寄存器指向forkret函数

8.forkrets函数的实现如下。首先是把输入变量current->tf复制给%esp，此时栈上保存了tf的值，亦即各寄存器的值。然后在trapret函数中使用popal和popl指令将栈上的内容逐一赋值给相应寄存器。最后执行iret，把栈顶的数据（也就是tf\_eip、tf\_cs和tf\_eflags）依次赋值给eip、cs和eflags寄存器。

并回答如下问题：

1. 在本实验的执行过程中，创建且运行了几个内核线程？

答：本实验创建且运行了两个内核线程，分别是idle和init线程。

2.语句 local\_intr\_save(intr\_flag);....local\_intr\_restore(intr\_flag); 在这里有何作用?请说明理由

完成代码编写后，编译并运行代码：make qemu 如果可以得到如 附录A所示的显示内容（仅供参考，不是标准答案输出），则基本正确。

答：避免在进程切换过程中处理中断。