**操作系统实验报告2**

171491221 姜世廷

练习1：分配并初始化一个进程控制块（需要编码）

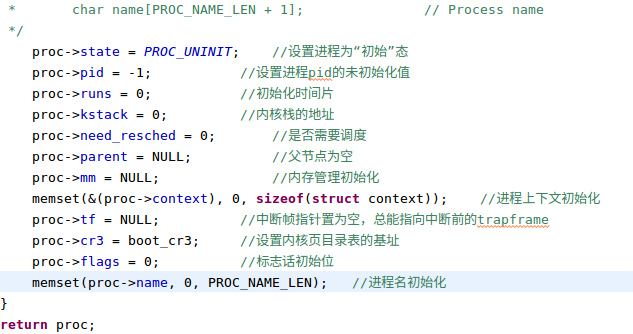
alloc\_proc函数（位于kern/process/proc.c中）负责分配并返回一个新的struct proc\_struct结

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：

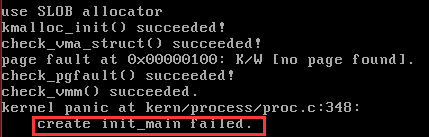
关键数据结构struct proc\_struct：



在alloc\_proc函数下填写代码后：



运行结果：

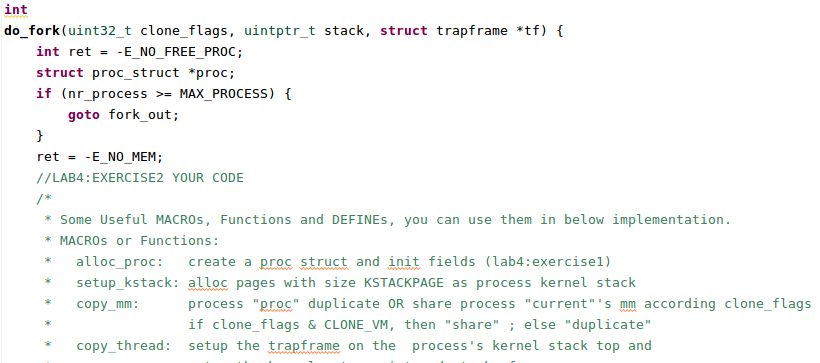


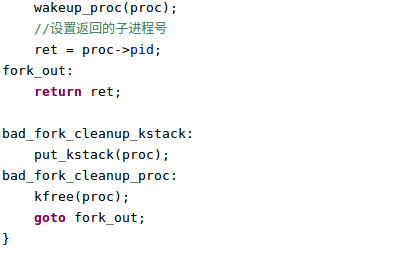
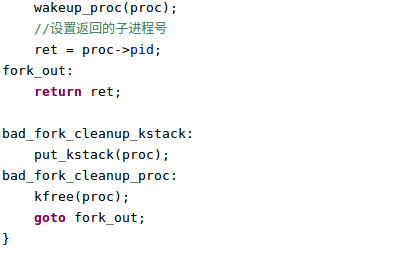
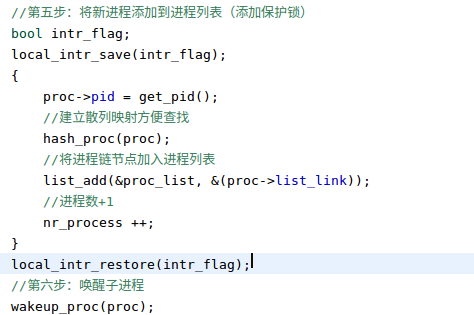
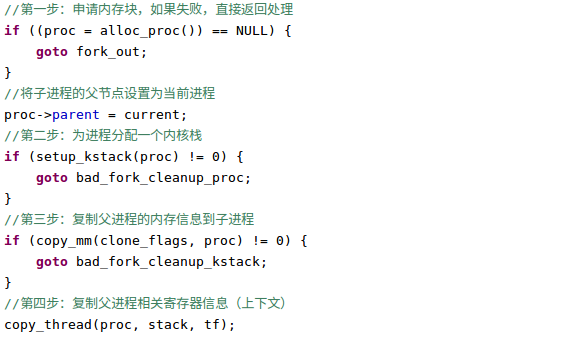
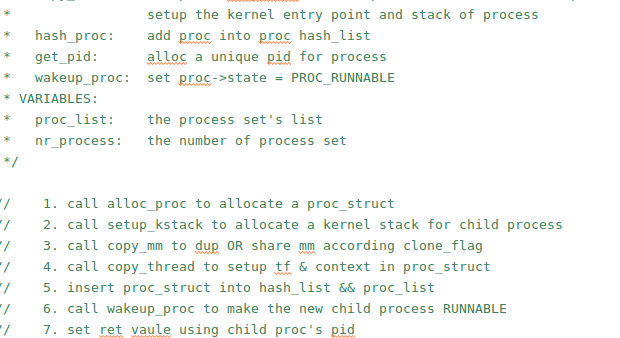
context和\*tf的作用：

1. context：进程的上下文，用于进程切换。起到保存现场的作用，实际利用context进行上下文切换的函数是在kern/process/switch.S中定义switch\_to。
2. \*tf:中断帧的指针，总是指向内核栈的某个位置。当进程从用户空间跳到内核空间时，中断帧记录了进程在被中断前的状态。当内核需要跳回用户空间时，需要调整中断帧以恢复各寄存器值。此外，ucore内核允许嵌套中断。因此为了保证嵌套中断发生时\*tf总是指向当前的tf，在内核栈上维护了tf的链。

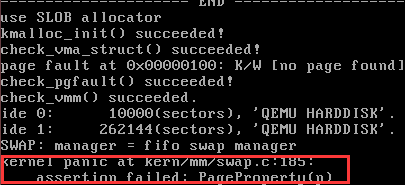
练习2：为新创建的内核线程分配资源（需要编码）

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。





运行结果：



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：在使用fork或clone系统调用时产生的进程均会由内核分配一个新的唯一的PID值。就是在分配PID时，设置一个保护锁，暂时不允许中断，保证了ID的唯一性，真正完成了资源分配的工作，与第一步的工作明显不同。do\_fork只是创建当前进程的副本，它们执行的上下文、寄存器和代码都是一样的。