os实验二

实验目的

了解内核线程创建/执行的管理过程 了解内核线程的切换和基本调度过程

实验内容

实验2/3完成了物理和虚拟内存管理，这给创建内核线程（内核线程是一种特殊的进程）打下 了提供内存管理的基础。当一个程序加载到内存中运行时，首先通过ucore OS的内存管理子 系统分配合适的空间，然后就需要考虑如何分时使用CPU来“并发”执行多个程序，让每个运行 的程序（这里用线程或进程表示）“感到”它们各自拥有“自己”的CPU。

本次实验将首先接触的是内核线程的管理。内核线程是一种特殊的进程，内核线程与用户进 程的区别有两个：

内核线程只运行在内核态 用户进程会在在用户态和内核态交替运行 所有内核线程共用ucore内核内存空间，不需为每个内核线程维护单独的内存空间 而用户进程需要维护各自的用户内存空间

相关原理介绍可看附录B：【原理】进程/线程的属性与特征解析。

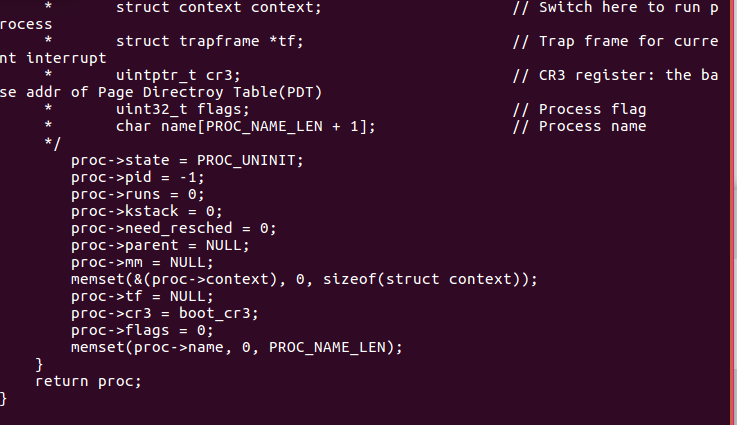
练习1：分配并初始化一个进程控制块（需要编码）

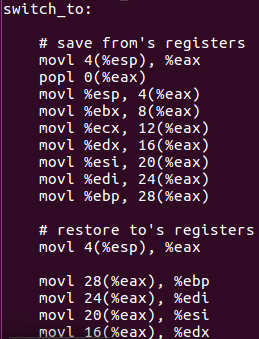
alloc\_proc函数（位于kern/process/proc.c中）负责分配并返回一个新的struct proc\_struct结 构，用于存储新建立的内核线程的管理信息。ucore需要对这个结构进行最基本的初始化，你 需要完成这个初始化过程。

【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）



Proc\_struct中的context用于进程的上下文的进程切换（实际上进行切换的是switch.s里面的代码），proc\_struct中的tf是中断帧的指针，总是指向内核栈的某个位置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让进程继续执行的各寄存器值。

（proc.h找不到defs.h不知道是不是之前的，proc.c运行出错，只能直接看代码理解）

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：

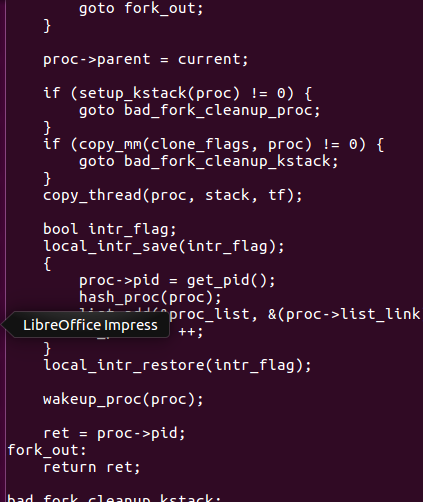
练习

211

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

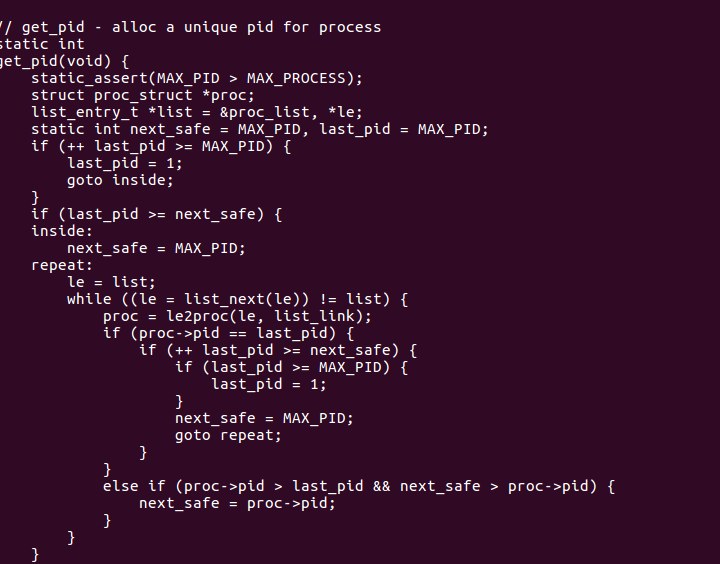
请在实验报告中简要说明你的设计实现过程。请回答如下问题：

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。



可以保证每个线程的ID唯一

理由：如图，函数get\_pid



见上图第一句的assert可以确保进程数一定不会多于可以分配的进程标识号的数目。然后函数将会扫描所有进程，找到一个当前没有被使用的进程号，储存在last\_pid中，作为新进程的进程号。具体来说，循环扫描每一个当前进程：当一个现有进程号和last\_pid相等时。则将last\_pid+1，当现有的进程号大于last\_pid时，这意味着在已经扫面的进程中[last\_pid,min(next\_safe,proc->pid)]这段进程号尚未被占用，继续扫描这样可以保证返回的新进程号一定能够没有被占用，即具有唯一的id

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）

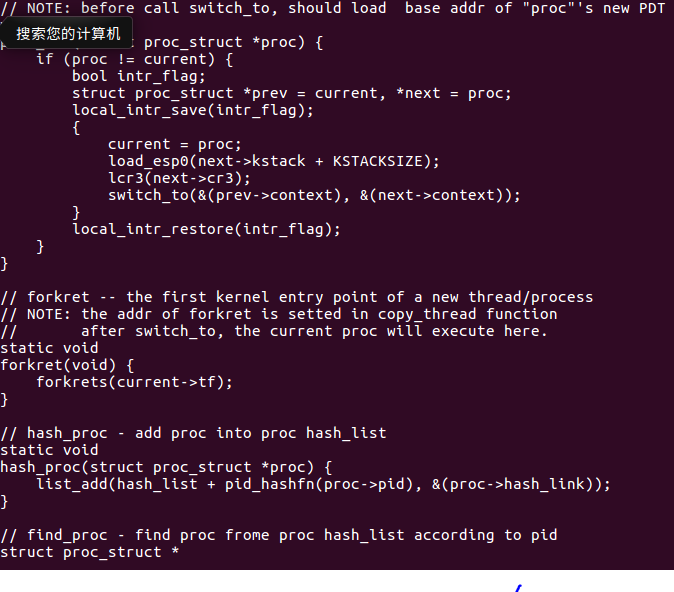
uCore Lab Documents

168练 习

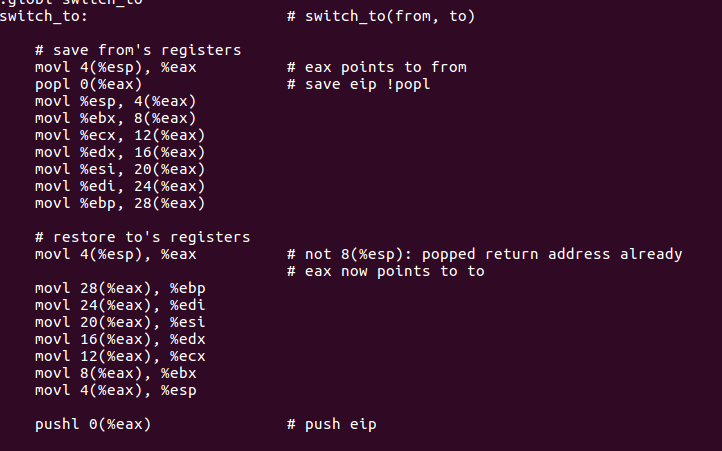
在本实验的执行过程中，创建且运行了几个内核线程？

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。



保存寄存器和恢复待调度进程的寄存器部分代码在switch\_to中



本实验在执行过程中，创建且运行了两个线程

idleproc：ucore第一个内核进程，完成内核中各个子系统的初始化，之后立即调度，执行其他进程。

initproc：用于完成实验的功能而调度的内核进程

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

它会屏蔽中断和打开中断，从而保护进程切换不会被中断，以免进程切换时其他进程再进行调度。