lab4实验

实验题目：

实验四：内核线程管理

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

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。

enum proc\_state state：表示进程状态，该进程的初始化状态未完成

uintptr\_t kstack:内核堆栈起始地址

struck mm\_struck \*mm:内存管理，初始值为null

struck trapfrname \*tf：初始化中断帧

context：进程的切换

代码：

proc->state = PROC\_ UNINIT;

proc->pid = -1;

proc->runs = 0;

proc->kstack = 0;

proc->need\_ resched = 0;

proc->parent = NULL;

proc->mm = NULL;

memset(&(proc->context)，0，sizeof(struct context));//初始化进程. 上

下文

proc->tf = NULL; //初始化中断帧，用于记录进程发生中断前的状态

proc->cr3 = boot\_ cr3; //因为是内核线程，所以CR3=boot\_ cr3

proc->flags = 0;

memset(proc->name, 0, PROC\_ NAME\_ LEN);

练习2：为新创建的内核线程分配资源

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。

这个函数调用了do\_ \_fork ()函数完成具体内核线程的创建工作，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。

1.调用alloc\_ proc，首先获得一块用户信息块

2.为进程分配一个内核栈

3.复制原进程的内存管理信息到新进程

4.复制原进程的上下文到新进程.

5.将新进程添加到进程列表

6.唤醒新进程

7.返回新进程

实验代码：

if（（proc = alloc\_proc())==NULL)

goto fork\_ out;

}

proc->parent = current;//设置父进程

if (setup. \_kstack(proc) != 0) {//分配了2页内核栈

goto bad\_ \_fork\_ cleanup\_ proc;

}

if (copy\_ \_mm(clone\_ flags, proc) != 0) {

goto bad\_ \_fork\_ cleanup\_ kstack;

coPy\_ \_thread(proc, stack, tf); //设置中断帧

bool intr\_ \_flag;

1ocal\_ intr. save(intr\_ \_f1ag) ;

proc->pid = get\_ pid();

hash\_ proc(proc);//将新进程加入hash\_ list

list\_ add(&proc\_ l1ist, &(proc->list\_ link));//将新进程加入

proc\_ list

nr\_ process ++;

}

1ocal\_ intr\_ restore(intr. \_flag);

wakeup\_ proc(proc);//唤醒进程，等待调度

ret = proc->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。



