summer2020 项目申请

项目名称: 完成多任务系统

个人信息

姓名:张俊杰

联系方式: zhangjunjie@ncic.ac.cn

github主页:主页

教育背景:中国科学院计算技术研究所硕士研究生一年级在读，专业为计算机科学与技术,方向为高性能计算/计算机体系结构

简要陈述:linux kernel一直以来是被认为最伟大的项目之一，也是开源世界的鼻祖。kernel的设计十分精细，精确到每个字节每一个比特的使用。恰巧我也是一个追求精致的程序员。我曾阅读过部分linux 0.11源码，进程的创建(fork)以及从系统调用到涉及到的处理流程,对进程上下文切换以及切换时涉及到的中断，寄存器使用，子进程的页面copy-on-write等等机制有过一定的了解。这是我初次参与开源项目的开发，但是我相信我的知识储备，我有能力做好多任务系统的开发。

项目详细方案

- 目前的仓库代码框架已经基本搭建好了，但是进程切换还是有时候会死机。需要先在两个进程情况下fix一些bug，再完善多个进程的创建，切换等等。
- 多任务系统涉及的无非是以下几个模块:
    1. 进程初始化:因为内核刚进入保护模式运行时，还没有进程这一说法，必须要手动创建一个进程，通常pid是0，在`linux 0.11`中即调用`sched_init()`完成。在`simple kernel`由`kernel_thread()`完成
    2. 进程/线程创建:沿用经典的进程间的父子关系，除了0号进程手动初始化，之后所有的进程都由fork()产生，比如0进程创建1号进程。这里涉及到进程`pcb`结构体的分配等等问题。linux 0.11还没支持thread，一个进程只包含了一个thread.
    3. 进程调度:这里涉及到进程的状态(`running zombie等`)，最重要的是进程调度算法和进程的上下文切换(和x86平台相关的)。

进程初始化

进程初始化主要是手工设置进程0的`tss,`和手工分配栈。目前的版本框架已经基本有了，可能存在隐性bug。需要调试。这里基本不会涉及到x86中断机制，但是会涉及到x86的寄存器相关的内容。

进程创建

进程创建发生在`fork()`，涉及到x86的中断机制，软件层面上需要查询是否有足够多的资源创建新的进程，比如是否有新的页面用于创建内核栈。创建进程还需要使用父子继承关系。新的进程可能需要加载可执行运行程序，需要和内存管理，即页面分配，写时复制等打交道。

进程调度

现有的调度算法是轮询进程链表(sched.c)，而且在进程切换之间会发生bug。需要先修复这个bug，之后视情况添加时间片轮转的进程切换算法。这里也会涉及到x86的中断机制和寄存器。

项目开发时间计划：

主要任务分成三个阶段，最后留一周做技术总结。需要做的工作大部分需要和硬件打交道，应该不会涉及源码大幅度改动。

阶段一:热身阶段

这一阶段主要熟悉环境和整理多任务系统的工作机制，并且重新温习一下linux kernel0.11 的机制。同时复习一下嵌入汇编和ia32相关知识（尤其系统调用的机制和用户内核态的切换）。确定一下调试的风格。编写/记录相关的文档。

周次	内容
第1周	熟悉编译过程，熟悉bochs运行kernel以及bochs的调试,阅读内核启动相关知识(grub),查看ia32手册阅读进程切换机制以及内核栈/用户栈切换机制,学习嵌入汇编以便为后面打下基础
第2周	阅读task.c/.h源码，查看相关的结构体(如task_pcb,task_mem等)，明确每一个成员的意义以及是否必要，学习数据结构(如进程的双向链表)。 查看阅读sched.c/.h（这部分代码不是很多）
第3周	阅读内存管理相关的源码(pmm.c,vmm.c,heap.c)，明确物理内存和页面管理/分配机制(malloc和alloc系列函数)，明确进程创建时发生的内存(比如页面分配)事件，查看线程相关的内存事件。
第4周	学习中断相关的函数,明确中断初始化(ldt)和中断发生的硬件机制。学习中断代码intr_s.s。 学习gdt.c,intr.c(idt),cpu.hpp
第5周	学习Debug相关的函数，一系列的printk_*函数，学会调试观察各个寄存器的值，尤其是保护模式下的GDT/LDT的。学习test.c源码，了解调试的具体手段(如何才能跑通一个test) 查看开发日志(devlog目录下，试着还原其中的一些bug)

这一阶段的主要目的就是能快速准确定位bug。

阶段二:修复bug，进程/线程创建

这一阶段主要是针对前面发现的问题进行修复，并且添加代码到现有的仓库中。使得:

1. 0号进程能正常被创建执行
2. 可以从0号进程fork()出第一个真正意义上的进程1，并且完全保证继承0号进程的上下文

总而言之，手工产生进程0，并且0进程可以完整地创建出进程1并且保证进程1的上下文是预期的结果。另外，注意线程的创建和进程的创建有所区别

同时需要编写文档记录详细的实现方案

周次	内容
第5～6周	调试task.c文件,以及x86_64/arch_init.c，观察系统只有0进程时各个寄存器的状态是否正常
第7～8周	调试task.c代码完整实现fork()机制，包括子进程的pcb申请，上下文继承等等 这里可能会涉及内存管理模块，可能需要自己手工写一个简单的内存分配函数。目标是能创建进程1且系统不会崩

这一阶段之后需要进行中期报告，需要整理一下文档和展示部分结果。

阶段三：添加任务切换相关代码

这一阶段的目的是完整实现进程的调度，(或许有时间片轮转算法)。这一阶段会涉及到较多的嵌入汇编知识:主要是对ldt和gdt的访问加载，x86的int指令的工作机制。

首先目标是创建的进程1能继续创建进程2，并且可以根据具体的调度算法进行的进程切换。

周次	内容
第9~10周	调试sched.c，添加schedule模块，根据需求实现进程调度优先级算法，调通两个进程的切换
第11周	创建第三个进程，系统一共有0,1,2个进程，观察进程的切换是否正常，调通
第12周	尝试多进程多线程,观察边界条件，和其他项目的同学同步一下，看看能不能跑通

阶段四(最后一周)

整理全部的技术文档，准备交付，答辩。

为什么选择这个项目

1. 我的方向是高性能计算(计算机体系结构)，对计算机底层有一定的了解，对x86汇编有一定的基础
2. 我很喜欢使用linux，对linux kernel和cpu交互界面十分感兴趣，而且我有linux 0.11源码阅读经历，对进程管理，内存管理有一定的知识积累。
3. linux是开源界的鼻祖，这次项目是我的初次尝试，我希望通过这样一个标志性的项目扎入开源的世界，同时让我对make , git工具的使用更加熟练。