实验 4 基于内核栈切换的进程切换

## 实验目的

* 深入理解进程和进程切换的概念；
* 综合应用进程、CPU 管理、PCB、LDT、内核栈、内核态等知识解决实际问题；
* 开始建立系统认识。

## 实验内容

* 编写汇编程序 switch\_to：
* 完成主体框架；
* 在主体框架下依次完成 PCB 切换、内核栈切换、LDT 切换等；
* 修改 fork()，由于是基于内核栈的切换，所以进程需要创建出能完成内核栈切换的样子。
* 修改 PCB，即 task\_struct 结构，增加相应的内容域，同时处理由于修改了 task\_struct 所造成的影响。
* 用修改后的 Linux 0.11 仍然可以启动、可以正常使用。
* （选做）分析实验 3 的日志体会修改前后系统运行的差别。

## 实验报告

|  |
| --- |
| 问题 1 针对下面的代码片段：  movl tss,%ecx  addl $4096,%ebx  movl %ebx,ESP0(%ecx)  回答问题：   * （1）为什么要加 4096； * （2）为什么没有设置 tss 中的 ss0。  问题 2 针对代码片段：  \*(--krnstack) = ebp;  \*(--krnstack) = ecx;  \*(--krnstack) = ebx;  \*(--krnstack) = 0;   * （1）子进程第一次执行时，eax=？为什么要等于这个数？哪里的工作让 eax 等于这样一个数？ * （2）这段代码中的 ebx 和 ecx 来自哪里，是什么含义，为什么要通过这些代码将其写到子进程的内核栈中？ * （3）这段代码中的 ebp 来自哪里，是什么含义，为什么要做这样的设置？可以不设置吗？为什么？  问题 3 为什么要在切换完 LDT 之后要重新设置 fs=0x17？而且为什么重设操作要出现在切换完 LDT 之后，出现在 LDT 之前又会怎么样？ |
| 1.%ebx指向PCB的起始地址，分配给PCB的大小是4KB,加上4096就指向高位置也就是空栈时内核栈栈顶的位置。  2.内核栈段的ss0永远都是0x10，所以不用修改。 |
| 1.等于0，用来区分父进程和子进程，p->tss.eax = 0（修改前）或者\*(--krnstack) = 0（修改后）。  2.来自sys\_fork压入栈的函数值，就是进入内核态前这两个寄存器的值，为了在返回时恢复现场。  3.来自sys\_fork的pushl %ebp 操作，不能，因为后面copy\_process还要用到%ebp。 |
| 因为如果不重新设置，则fs的隐藏部分存放的就是上一个进程的用户态的基地址和段限长，会产生错误的段属性值。若先设置在切换LDT，就会在切换前的进程中进行操作，导致以后访问空间的之前进程的用户空间。 |

## 实验步骤

1. 重写sched.c 中的schedual函数,注意实验手册中LDT应为\_LDT，注意pnext的初始值，网上的大部分实验报告写的是current，这样不对，应该为&(init\_task.task)

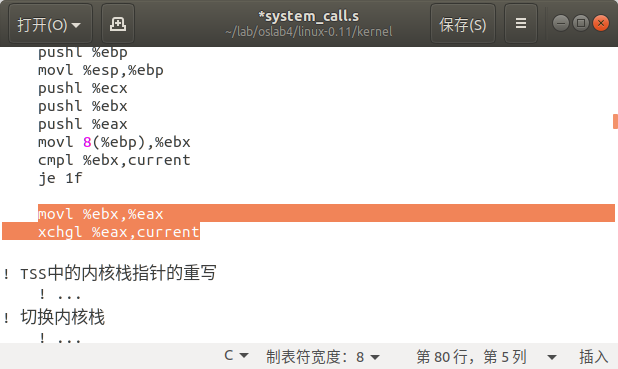
|  |  |
| --- | --- |
| Pnext的值 | 效果 |
| 不设初值 | 无限重启 |
| NULL | 报错如下图  2020-05-08 15-48-21 的屏幕截图 |
| Current | 看起来正常，但无法输入指令 |
| 直接用switch\_to(task[next],\_LDT(next)); | 无限重启 |
| &(init\_task.task) | 正常 |

|  |
| --- |
| 2020-05-08 15-34-51 的屏幕截图2020-05-05 22-53-35 的屏幕截图2020-05-05 22-53-38 的屏幕截图 |

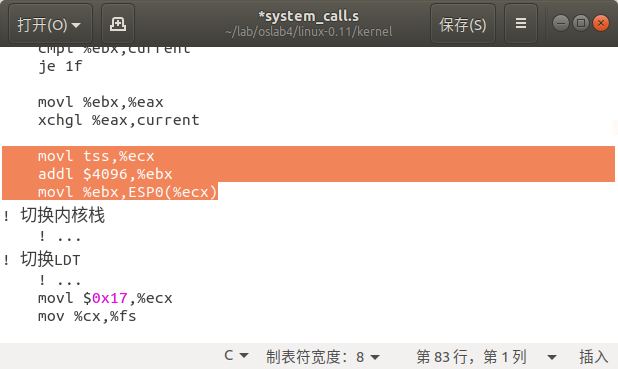
1. 注释原有的switch\_to函数

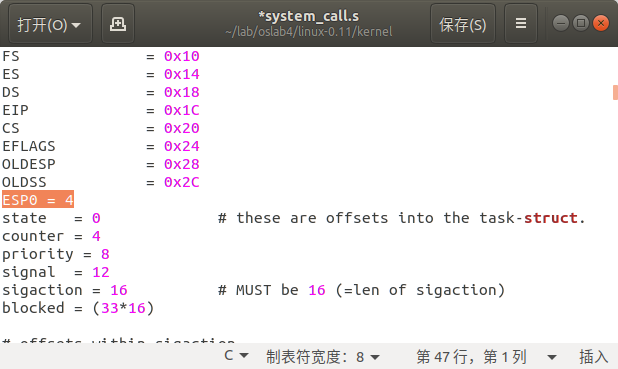
|  |
| --- |
| 2020-05-05 22-54-42 的屏幕截图 |

1. 在system\_call.s中编写switch\_to
   1. 编写部分:交换PCB指针



* 1. 设置为空栈时的栈顶指针



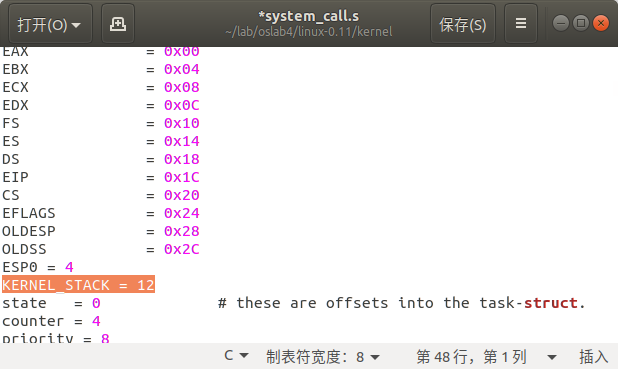


* 1. 定义全局变量



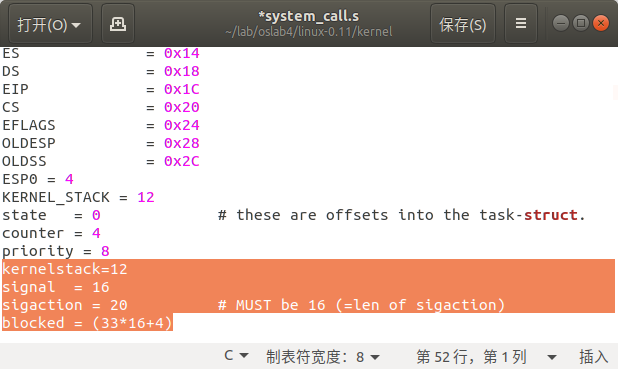
* 1. 切换内核栈指针





* 1. 修改结构体、位置及其初值







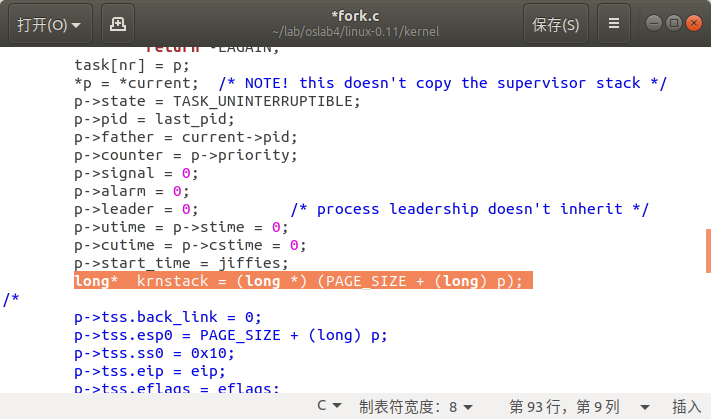
* 1. 切换LDT



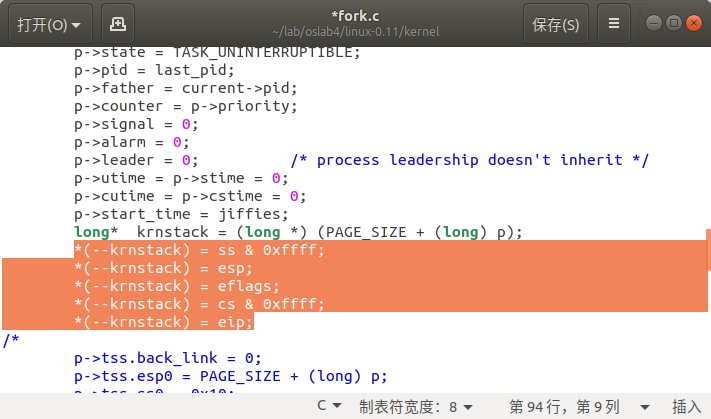
1. 修改fork
   1. 不再需要tss相关操作



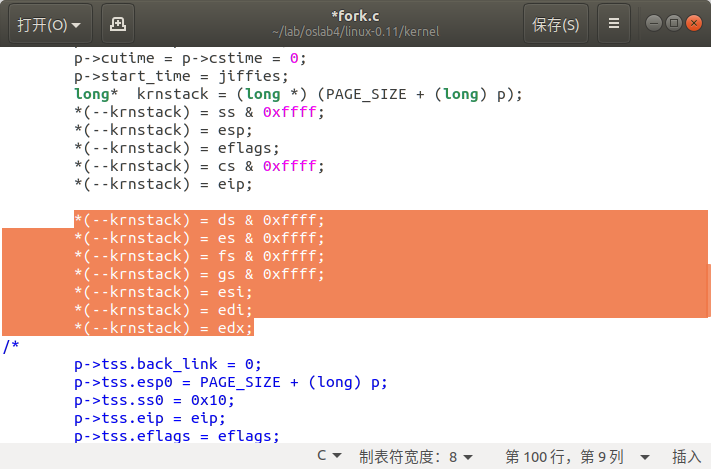
* 1. krnstack 初始化
     1. 找到内核栈位置



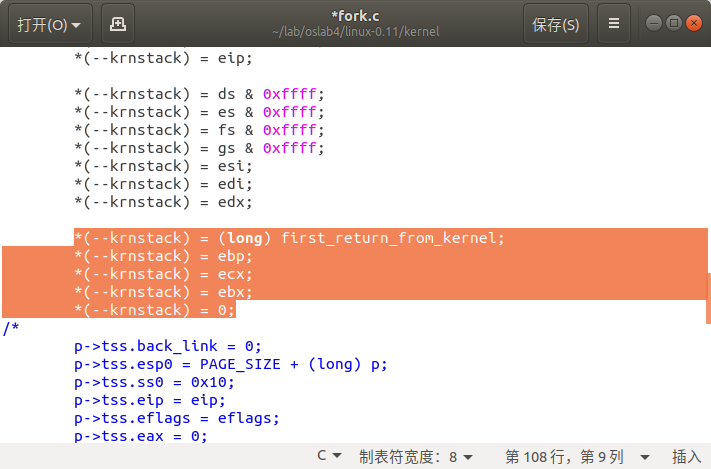
* + 1. 压入first\_return\_from\_kernel中 iret需要的参数



* + 1. first\_return\_from\_kernel中其他pop的内容（实验指导未给出）



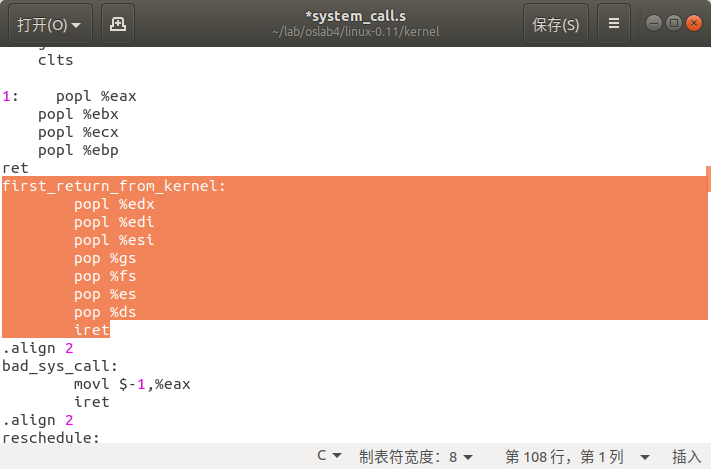
* + 1. Switch\_to 四次弹栈和ret的参数，0为fork产生的子进程返回值



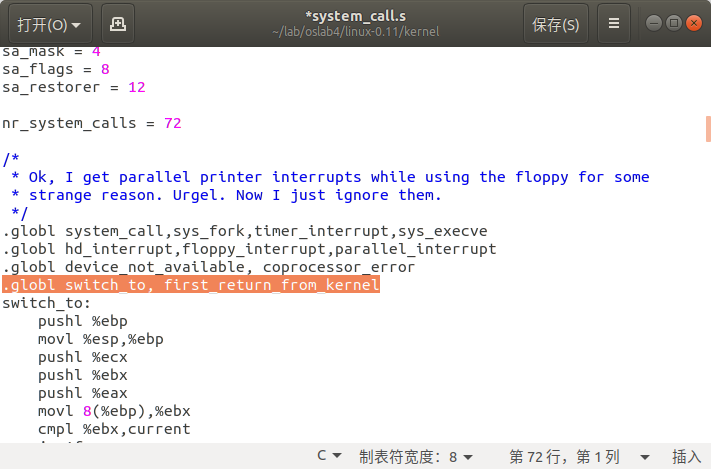
* + 1. 初始化栈顶位置

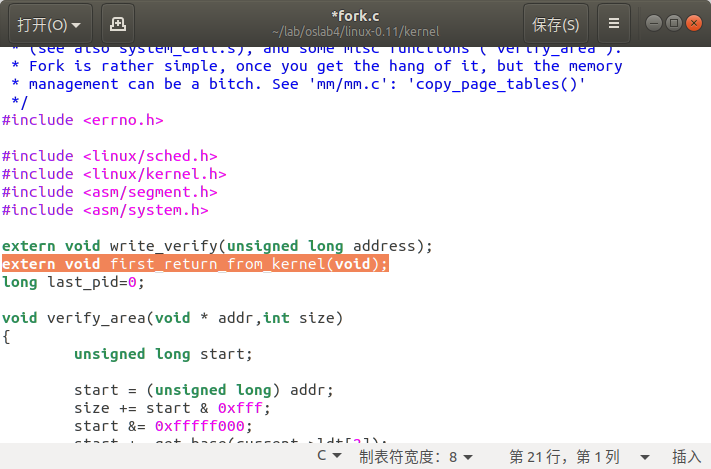


* 1. first\_return\_from\_kernel实现返回至用户态



* 1. 添加全局声明





1. 编译运行，可正常使用

