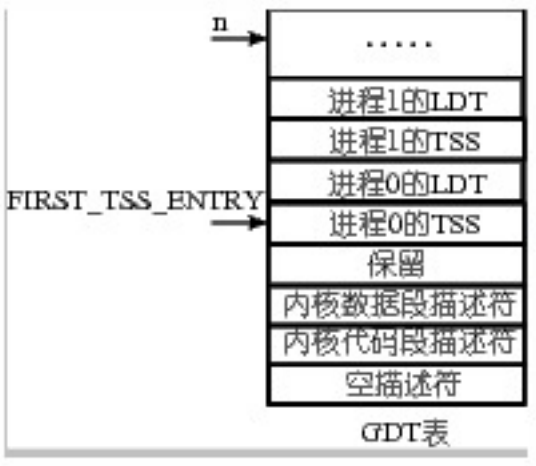
yasusu L69 2016-08-27 20:12 Linux 计算机专业课

135 1

实验代码

“操作系统原理与实践”实验报告

基于内核栈切换的进程切换



GDT表内容如上，_TSS宏给出了进程n的TSS位置：

#define _TSS(n) ((((unsigned long) n)<<4)+(FIRST_TSS_ENTRY<<3))

3、在sched.h中去掉switch_to宏。加上函数的声明

4、第一个问题，加4096刚好是1页的大小（4K），加上之后ebx就指向这个进程使用的内存页的末尾，即该进程的堆栈，这样就将其内核栈清空了。ss0不必设置，因为都在内核数据段中，段寄存器的值相同。

5、完成内核栈的切换也非常简单，将寄存器esp（内核栈使用到当前情况时的栈顶位置）的值保存到当前PCB中，再从下一个PCB中的对应位置上取出保存的内核栈栈顶放入esp寄存器。

6、注意：schedule中pnext必须要初始化

7、注意修改system_call.s中的与task_struct相关的常量，如signal、sigaction、blocked。

8、注意fork中堆栈的初始化。

9、重置fs段寄存器的问题

> fs的作用前面已提及，关于段寄存器参考网上的资料：
>
> 为了避免在每次存储器访问时，都要访问描述符表而获得对应的 段描述符，从80286 开始每个段寄存器都配有一个高速缓冲寄存器，称之为段描述符高速缓冲寄存器或描述符投影寄存器，对程序员而言它是不可见的。每当把 一个选择子装入到某个段寄存器时，处理器自动从描述符表中取出相应的描述符，把描述符中的信息保存到对应的高速缓冲寄存器中。此后对该段访问时，处理器都 使用对应高速缓冲寄存器中的描述符信息，而不用再从描述符表中取描述符在保护模式下段寄存器中只能存储段号（segment selector，也译作“段选择符”），再由段号映射到存在内存中的GDT（global segment descriptor table，全局段号记录表），读取段的信息。
>
> 所以这里将17置给fs，即可完成用户段段基址的修改。

copy_process注意其一众参数，及调用途径： 系统调用fork()->int 0x80 （ss、esp、eflags、cs、eip入栈，对应函数的后5个参数）->跳转到system_call（ds、es、fs、edx、ecx、ebx入栈）->调用（返回地址入栈，对应函数参数none）sys_fork（gs、edi、esi、ebp、eax入栈）

switch_to切换完进程后，先要pop4个寄存器，再ret返回。 所以先压栈4个寄存器，返回地址（first_return_from_kernel），first_return_from_kernel要从堆栈中弹出一系列寄存器，因此copy_process时应进行准备。

几条线： 1、fork新建进程：fork->int->system_call->sys_fork->copy_process 2、切换至新进程线：int->schedule->switch_to，要保证copy_process产生和下面情况相同的现场 3、切换至已有进程线：int->schedule->switch_to，所有函数正常返回

0