1. 中断和异常的护理与抢占式多任务

1140310606 张茗帅

**一些储备的基础知识：**

1.中断包括硬件中断和软中断。

硬件中断是由外围硬件设备发出的中断信号引发的，完全随机，与处理器执行不同步，当中断发生时，处理器会先执行完当前的指令，再在对中断进行处理。

软中断 int n，n是中断号。

1. 异常分三种：

程序错误异常，处理器检测到程序错误。

软件引发的异常，into、int3、bound。允许在当前点上检查实施异常处理条件是否满足。

机器检查异常，与处理器型号相关。

按照严重性分三种：

故障，可以纠正，压栈的返回地址指向引起故障的那条指令。

陷阱，常用于调试目的，压栈的返回地址是陷阱截获指令的下一条指令地址。

终止，最严重，典型的8号双重故障，重大错误，将任务从系统中抹去。

1. NMI和异常的向量是有处理器自动给出的，硬件中断向量是IO中断控制芯片送给处理器的，软中断的向量是由指令中的操作数给出的。
2. 实模式下使用中断向量表ITV，将中断向量乘以4，作为表内偏移访问中断向量表，从中获得段地址和偏移地址。

保护模式下，使用IDT表，里面可以存放中断门、陷阱门和任务门描述符。IDT的第一位有效。线性基地址和限长存在IDTR中。处理器用中断向量乘以8的结果去访问IDT，得到描述符，用其中的目标代码段选择子得到目标代码段描述符，再从目标代码段描述符中取出段基地址，加上门描述符中的偏移量，得到中断处理子程序的线性地址（任务门不完全这样）。若开启分页，则转化为物理地址。当处理器中断向量访问IDT超出了界限，产生常规保护异常。

1. 中断和异常处理程序的保护

检查当前代码段描述符的特权级DPL（可以从门描述符中的段选择子，从GDT、LDT中找到）需要CPL>=目标代码段的DPL（目标代码段的描述符特权级别高），否则GP异常。

但是不检查RPL（进入中断或异常处理程序或者通过任务门发起任务切换时）。

软中断int n 和int3 、into时当前特权级CPL需要CPL<=门描述符的DPL。（当前特权级别高）

转换栈，根据处理程序的特权级，从当前任务TSS中取得斩断选择子和栈指针，旧栈的选择子和站指针也会压入新栈。EFLAGS、cs、eip的当前状态压入新栈，有错误代码的异常，错误代码压入新栈，紧挨着eip。旧栈中也存放这4项。

中断门和陷阱门区别在于IF位，中断门禁止嵌套的中断，陷阱门允许其他中断优先处理。（仅影响硬件中断）

1. 中断任务

IDT表中找到任务门，引发任务切换，把发生故障的任务从系统中抹去。

中断和异常引发任务切换不再保存CS、EIP，切换任务完成后，如果有错误码错误码会压入新栈。任务不可重入，iretd之前必须关中断。任务门只有在int3、int n和into时才会检查特权级，而且不检查代码段的特权级，只需要CPL<=任务门的DPL。

1. 错误代码

iret或者iretd指令时，错误代码需要执行前弹出。

外部异常、int n不会压入错误代码。

**代码部分**，感觉17章的代码与linux0.00十分神似。只不过添加了分页机制等等复杂的表示方式，更加具体。

整体来讲，代码的特色就是运行在平坦模式下，没有过多的段的区分。主引导程序中，完成了将内核程序加载到0x40000地址和加载GDT的工作，值得说明的是，这里还进行了分页机制的开启，并按照16章的方式将内核程序映射到高端全局空间，栈顶同样进行了映射。

**①跳转到内核程序的过程时，已经是在分页机制下进行的**



0x80040004，即1000000000 0001000000 000000000100->0x200 0x040 0x004

从CR3取出页目录表物理地址0x00020000，加0x200\*4得到页目录项0x00020800



得到页表的物理地址0x00021000，加0x040\*4得到页表项0x00021100



得到页表物理地址0x00040000，加0x004得到页表对应的物理地址0x00040004



得到内核程序物理地址0x80040d9f



**②调用函数make\_gate\_descriptor**



段及地址0x0平坦模式，偏移地址0x8004017f也是线性地址，页部件计算之后得到的物理地址为0x4017f，里面存放着机器指令



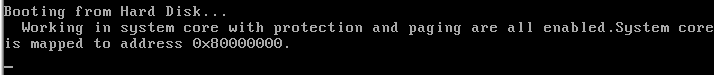
**③用定时中断进行任务切换**

与linux0.00中的任务切换方式相同，程序调用 rtm\_0x70\_interrupt\_handle过程，时钟中断，它放在全局空间中，由RTC芯片触发。

这里进行切换时，用到了TCB链表，每个任务的TCB的0x00处放的的下一个任务TCB的基地址，只要得到TCb链表的头，就是第一个任务TCB，就可以使用类似链表的增删查改的方式进行抢占式任务切换。查找时，可以用TCB的0x04位置显示任务状态的字段判断其状态来确定是否进行切换，每一个被切换的任务，其TCB都会转移到TCB链表的尾部，最后一个TCB的0x0处存放的为0.

**④sti**

第954行开中断，显示

 下面就允许进行抢占式任务切换了。开中断之后进行内核任务的创建和用户任务的创建，并将任务TCB链表建立起来，它是抢占式任务切换的前提。

一直在说抢占式，简单来说，就是通过时钟中断，通过TCB链表进行调度，进行的任务切换。

**⑤时钟中断中的任务切换指令**



这与linux0.00中基本一致。切换回来时再次发生时钟中断，从下一条开始执行。

任务切换就是在屏幕上交替打印不同的显示信息。