OS 进程管理与调度

0 概述

能运行多任务是操作系统的基本要求之一。为了方便统筹每个任务的运行，需要有包含该任务的资源、状态的数据结构，即进程控制块(Processing Control Block, PCB)。对于英特尔 x86 架构处理器来说，进程的调度涉及了中断的管理、代码特权级的转移等等内容。

1 进程控制块

进程控制块包含最基本的资源是可以存放当前寄存器状态的内存空间，个人起别名寄存器帧。其中进程控制块中的 EIP 和 ESP 分别存放进程函数的地址和可供其使用的栈区地址。若某进程即将被调度至运行状态，该进程控制块的寄存器帧中的记录将恢复到现场，并通过 EIP 进行跳转。

2 TSS

任务进程不应该运行在内核态。当代码的转移涉及特权级别的变化时，使用的堆栈也要相应改变。英特尔开发 TSS 表的初衷是让每一个进程都拥有一个 TSS 表记录，里面有所有寄存器的记录和三个 RING 的堆栈的信息。当进程发生调度时该进程的 TSS 表复制到系统 TSS 的相应表项中，恢复所有寄存器的值并根据进程的特权等级设置相对应的 SS 和 ESP。然而在 Linux 内核中和进程相关的寄存器都存储在 PCB 结构体中，并只记录了进程当前 RING 的 SS 与 ESP。当进程调度时只需改变 SS 和 ESP 的值，而其他寄存器的状态保存于相关进程的 PCB 寄存器帧中，无需拷贝。

进程调度时涉及的堆栈有三项，个人称为（进程）运行栈、（进程）状态存储栈和内核栈。进程运行中使用的是本身所拥有的栈，当发生调度时（通常是由中断触发）时硬件将会把 TSS 表中相应 RING 的 SS 和 ESP 设置为当前寄存器中，此时的 ESP 是状态存储栈。所谓的状态储存栈也就是指向进程表中寄存器帧尾，这样当中断发生时可以通过一系列的入栈操作将寄存器状态保存到当前进程的寄存器帧中。而中断处理中使用的是内核栈。

3 中断

对于多进程的系统来说，让一个任务持续占领着 CPU 资源是件不合理的事情。合理的做法之一是给每一个任务进程一定的时间片，当该进程用完当前时间片时挂起，可在时钟中断的处理函数中进行进程的调度。中断有两个重要概念，中断向量表和可编程控制器（例如 8259A 芯片）。

3.1 中断的初始化

中断的初始化步骤：

- 设置可编程控制器，设置各个中断的向量地址
- 完成中断向量表

其中，第二步在 x86 保护模式下是以中断门的形式实现。构造一个中断描述子表(IDT)，每一号中断对应一个中断门。中断门中记录着中断处理函数的地址。

3.2 中断重入

中断函数响应时，有可能下一个中断也会发生。如何处理待续。

3.3 中断中调度

中断处理中调度进程，将切换到 TSS 表中记录的堆栈中。先确保当前 ESP 指向当前进程的寄存器帧尾，将当前所有寄存器入栈。此时可切换至内核栈中进行相关处理，其中应清除中断相关标志位。最后将 ESP 指向下一个进程的寄存器帧头，所有寄存器值出栈并跳转。