目 录

1 进程

1.1 形成进程的必要考虑

CPU 的个数通常总是小于进程的个数，所以我们需要进程调度，使得系统总有“正在运行的”和“正在休息的”进程。

为了让“正在休息的”进程在重新醒来时记住自己挂起之前的状态，我们需要一个数据结构记录一个进程的状态。

还需要考虑的是，进程和进程切换运行在不同层级上。

还有一点需要考虑，就是进程自己不知道什么时候被挂起，什么时候又被启动，我们需要知道诱发进程切换的原因不只一种，比如发生了时钟中断。

1.2 最简单的进程

首先介绍一下进程切换的情形：

```
1  一个进程正在运行着，此时时钟中断发生。
2  特权级从ring1跳到ring0，开始执行时钟中断处理程序。
3  中断处理程序调用进程调度模块，指定下一个应该运行的进程。
4  中断处理程序结束时，下一个进程准备就绪并开始运行，特权级从ring0跳回ring1。
```

从上述过程得知，我们需要完成几个部分：

1. 时钟中断处理程序。
2. 进程调度模块。
3. 两个进程。

1.2.1 简单进程中的关键技术

需要考虑保存进程的状态，用于恢复进程，所以我们要把寄存器的值统统保存起来。

一般使用 push 或 pushad 保存大多寄存器的值，并且把它写在时钟中断例程的最顶端，以便中断发生时马上被执行。

当恢复进程时，使用 pop 来恢复寄存器的值，虽然执行指令 iretd 回到原先的进程。

1.2.2 进程表