中断与分时多任务

蒋炎岩

南京大学　　计算机科学与技术系　　计算机软件研究所

本讲概述

为什么 while（1）；死循环不会把电脑卡死?

本讲内容

- 中断机制
- 实现分时多任务

中断机制

回到一个经典问题

为什么 while（1）；死循环不会把电脑卡死?

因为有中断

- 相当于在指令之后插入一段代码

```
if (has_interrupt && int_enabled) {
  interrupt_handler();
}
```

- 类似的是异常机制
  - 有时候称为同步中断

x86-64 中断过程

比 “函数调用” 复杂一些

- 函数调用需要保存 PC 到堆栈 (`%rsp`)
  - 但中断不仅要保存 PC (尤其是在有特权级切换的时候)

中断处理

- 自动保存 RIP, CS, RFLAGS, RSP, SS, (Error Code)
- 根据中断/异常号跳转到处理程序 (特权级切换会触发堆栈切换)
  - `int $0x80` 指令可以产生 128 号异常
  - 时钟会产生 32 号中断
  - 键盘会产生 33 号中断
  - ……

为什么死循环不会把电脑卡死？

中断是强制的 (普通的进程不能关闭)

- Ring 3 关闭中断将导致 `Exception(GP(0))` (**手册**)
  - 发生 13 号异常
  - Error Code 0
    - 异常机制进入操作系统代码执行
    - 操作系统发送 SIGSEGV 信号
  - 捕捉 SIGSEGV: **cli.c**

中断发生后，操作系统代码将会切换到另一个进程执行

- “调度”：让进程公平地共享 CPU

实现分时多任务

代码选讲

Abstract Machine

- trap64.S
- cte.c

调试“迷你”操作系统

- **thread-os.c**
  - `-s -S` 启动 QEMU
  - gdb `target remote localhost:1234`
    - **gdbnotes for CSAPP**

总结

操作系统：中断驱动的上下文切换

应用程序视角

- 一组系统调用的 API
  - 进程管理、存储管理、文件管理……

硬件视角

- 操作系统是一个中断处理程序
  - 设备中断：上下文切换；调用设备驱动程序；……
  - 系统调用：操作系统代码执行 (API 实现)
  - 异常：发送信号 (类似系统调用)

End.