# 第 5 章 中断与系统调用

## 阅读要求

1. 启动部分： Bootloader：bootasm.S bootmain.c xv6 初始化模块：main.c bootother.S 2. 中断与系统调用部分： trap.c trapasm.S vectors.S & vectors.pl syscall.c sysproc.c proc.c 以及相关其他文件代码

## 讨论总结

**用户态和内核态**

在操作系统中，内核态是指运行时操作系统内核的状态。在这种状态下，内核程序可以访问任何现有硬件并执行任何现有指令。用户态是指用户进程执行时系统的状态。在这种状态下，用户进程只能执行部分指令，访问硬件并根据操作系统提供的系统调用与其他进程交互。

将内核态与用户态隔离是为了提高系统的整体安全性和健壮性，并防止恶意进程和错误进程损坏系统。

**中断和系统调用**

中断是一种允许操作系统响应外部硬件的机制。例如，当用户进程执行并加载了另一个用户进程请求的磁盘文件时，需要设计一个中断信号来通知操作系统暂停当前用户进程。让操作系统处理这个中断事件;系统调用是一种机制，它使用户进程能够进入内核状态并请求某个系统服务，例如使用系统提供的syscall指令进入内核，并完成需要内核的输入和输出任务进程的权限。然后返回用户模式，进程继续执行。

当计算机运行时，它通过CPU中某些寄存器的权限位知道它是处于内核状态还是用户状态。例如，在x86系统中，CPU通过检查％cs寄存器中的CPL位来检查当前指令的执行权限级别。在XV6系统中，CPL0代表内核模式，CPL3代表用户模式。如果指令的执行权限与CPL位的值不匹配，则生成一般保护错误。

**计算机开始运行阶段就有中断吗？XV6 的中断管理是如何初始化的？**

计算机开始运行阶段就有BIOS支持的中断。由于xv6在开始运行阶段没有初始化中断处理程序，于是xv6在bootasm.S中用cli命令禁止中断发生。 picinit()方法和oapicinit()方法初始化可编程中断控制器；consoleinit()方法和uartinit()方法设置I / O和设备端口的中断。 然后，tvinit()调用trap.c中的代码来初始化中断描述符表，关联vectors.S中的中断IDT条目，调用idtinit()以在调度开始之前设置第32个时钟中断，最后在调用中调用它。调度()。 Sti()启动中断并完成中断管理初始化。

**XV6 是如何实现内核态到用户态的转变的？XV6 中的硬件中断是如何开关的？**

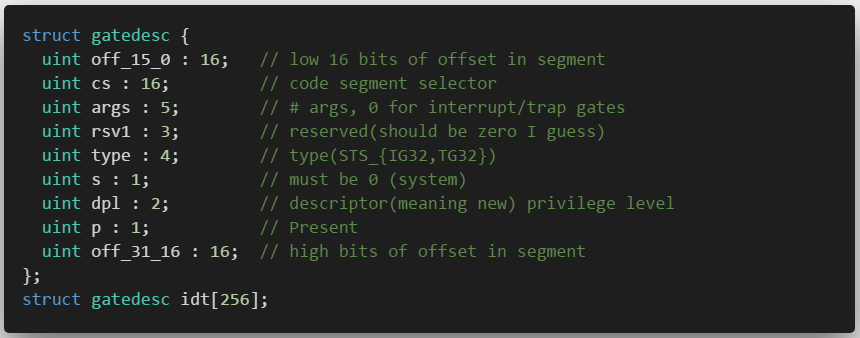
从内核态到用户态的转变主要是靠设置程序状态字，程序状态字存储在%cs寄存器的低3位中。XV6系统的硬件中断由可编程中断控制器进行设置和管理，比如通过调用ioapicenable控制IOAPIC中断。处理器可以通过设置 eflags 寄存器中的 IF 位来控制自己是否想要收到中断：通过命令cli关中断, 通过命令sti开中断。

**实际的计算机里，中断有哪几种？**

1. 外部设备请求中断。一般的外部设备（例如键盘，打印机，A / D转换器等）在完成其自己的操作之后，向CPU发出中断请求，请求CPU为他服务。由计算机硬件异常或故障引起的中断，也称为内部中止;
2. 故障被迫中断。计算机在一些关键部件中具有自动故障检测设备。如操作溢出，内存读取错误，外部设备故障，电源故障等报警信号等，这些设备的报警信号可以中断CPU并执行相应的中断处理;
3. 实时时钟请求中断。在控制中遇到定时检测和控制，并且经常使用外部时钟电路（可编程）来控制其时间间隔。当需要定时时，CPU发出命令以启动时钟电路。一旦达到指定的时间，时钟电路发出中断请求，CPU就完成检测和控制工作;
4. 数据通道被中断。数据通道中断也称为直接存储器访问（DMA）操作，例如磁盘，磁带驱动器或CRT，它们是与存储器直接交换数据所必需的。
5. 程序自愿中断。 CPU执行特殊指令（陷阱指令）或由硬件电路引起的中断是程序自愿中断。它指的是当用户调试程序时程序使用的检查方法，程序主动中断检查中间结果或查找错误，例如断点中断。单步中断等。

**什么是中断描述符，中断描述符表？在 XV6 里是用什么数据结构表示的？**

中断描述符代表并描述一个特定的中断。中断描述符表是用于在X86架构中以受保护模式存储中断服务例程信息的数据结构。 在XV6数据结构中，涉及的数据结构如下：

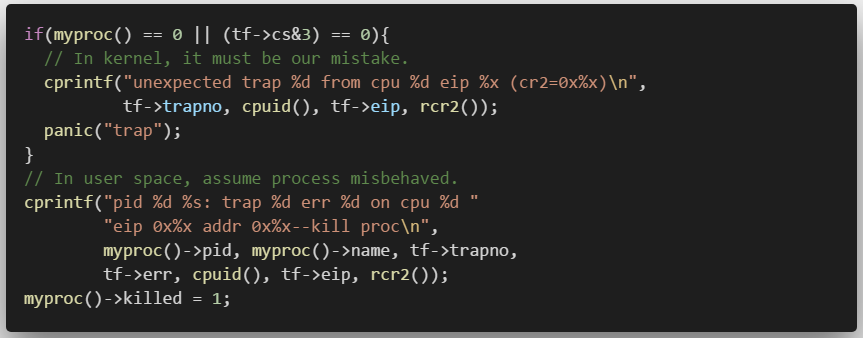
其中，struct gatedesc的格式与X86架构所需的格式完全相同。对于第i个中断描述符，CS寄存器存储内核代码段的段号SEG\_KCODE，偏移部分存储vector [i]的地址。在XV6系统中，所有vector [i]地址都指向trapasm.S中的alltraps函数。

**请以某一个中断为例，详细描述 XV6 一次中断的处理过程。**

以零除错误为例。 在XV6指令执行期间遇到除零错误时，CPU硬件首先发现此错误并触发中断处理机制。 在中断处理机制中，硬件执行以下步骤：

1. 从中断描述符表中获得对应的中断描述符。
2. 检查CS的域 CPL <= DPL，即操作是否合法，DPL 是描述符中记录的特权级。
3. 如果目标段选择符的 PL < CPL，就在 CPU 内部的寄存器中保存ESP和SS的值。
4. 从一个任务段描述符中加载SS和ESP。
5. 将用户信息段SS/ESP、EFLAGS寄存器、CS寄存器、EIP寄存器依次压栈。
6. 清除EFLAGS的一些位。
7. 设置CS和EIP为描述符中的值。

此时，由于CS已设置为描述符中的值（SEG\_KCODE），系统已进入内核态，并且EIP指向trapasm.S中alltraps函数的开头。 在alltrap函数中，系统按下用户寄存器，构建一个Trap Frame，并将数据寄存器段设置为内核数据段，然后跳转到trap.c中的trap函数。 在陷阱功能中，首先检查中断呼叫号码，发现这不是系统调用，也不是外部硬件中断，因此请输入以下代码段：

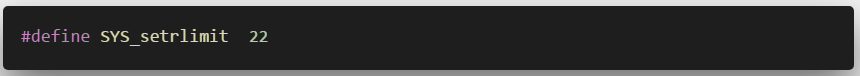


根据系统程序或用户进程是否触发中断来执行不同的处理。 如果用户进程错误，系统将终止用户进程; 如果内核进程错误，则在输出错误消息后，整个系统进入无限循环。

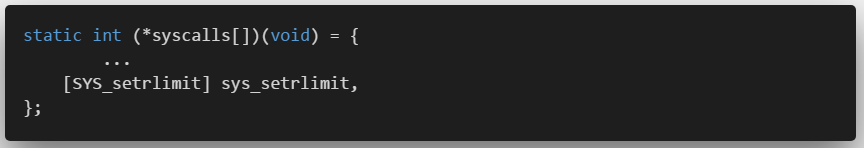
如果是一个可以修复的错误，比如页错误，那么系统会在处理完后返回trap()函数进入trapret()函数，在这个函数中恢复进程的执行上下文，让整个系统返回到触发中断的位置和状态。

**请以系统调用** **setrlimit为例，叙述如何在 XV6 中实现一个系统调用。**

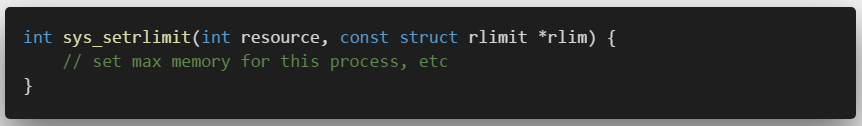
1. 首先，在syscall.h中添加一个新的系统调用号



1. 其次，在syscall.c中增加入口函数中添加setrlimit的系统调用处理函数的地址



1. 同时，在sysproc.c中声明并实现setrlimit的系统调用处理函数

4. 最后，在user.h中声明setrlimit()这个函数系统调用函数的接口，并在usys.S中添加有关的用户系统调用接口。

