

第 3 讲中断、异常和系统调用

第一节：基本概念与原理

向勇、陈渝

清华大学计算机系

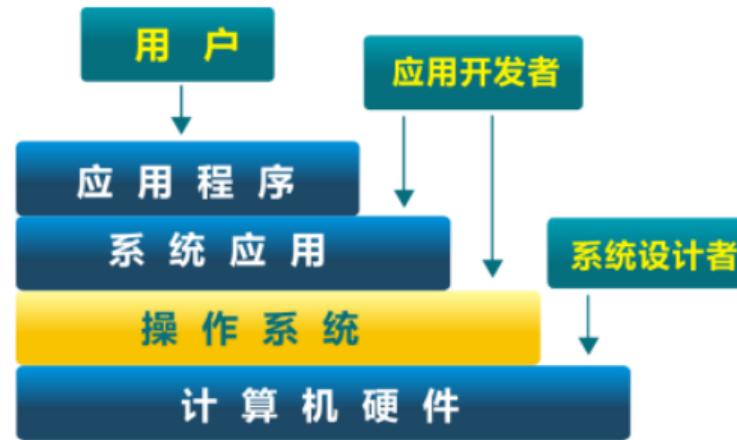
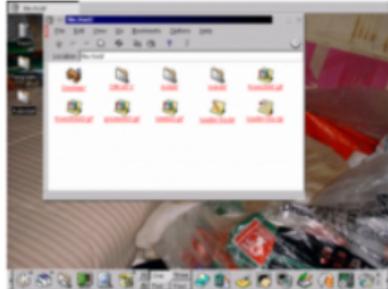
xyong,yuchen@tsinghua.edu.cn

2020 年 5 月 5 日

基本概念与原理

系统调用 (system call)

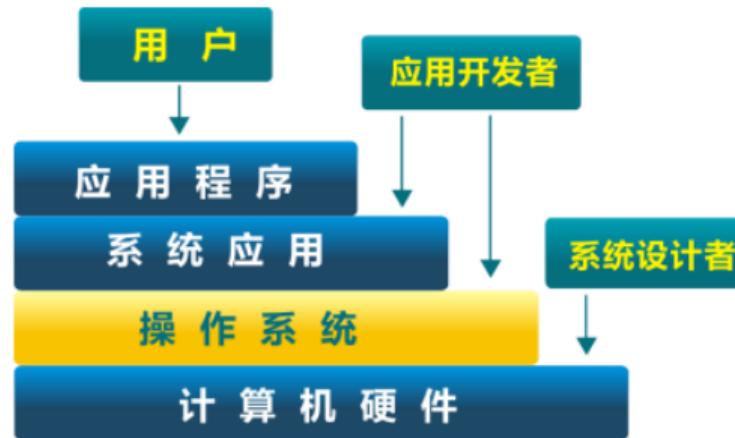
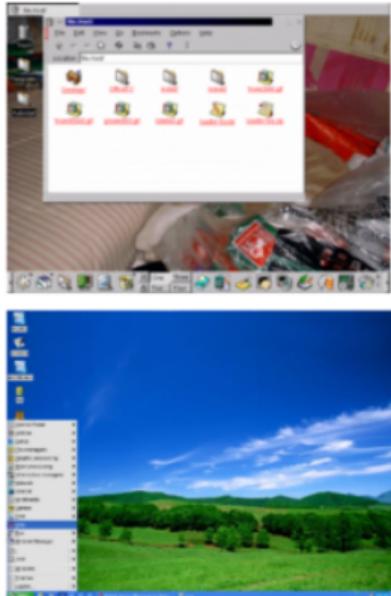
应用程序**主动**向操作系统发出的服务请求



基本概念与原理

异常 (exception)

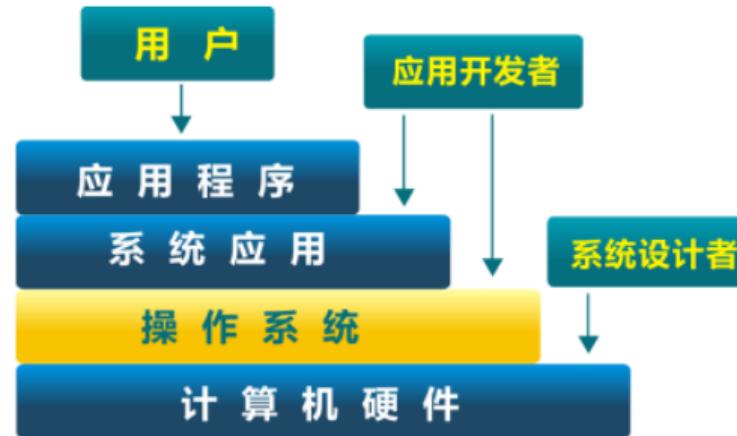
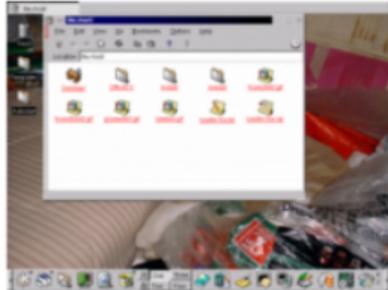
非法指令或者其他原因导致**当前指令执行失败**, (如: 内存出错) 后的处理请求



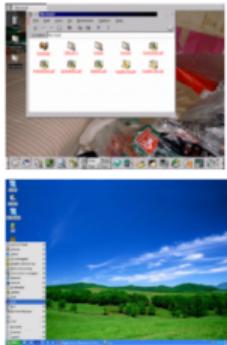
基本概念与原理

中断 (hardware interrupt)

来自硬件设备 (外设, device) 的处理请求

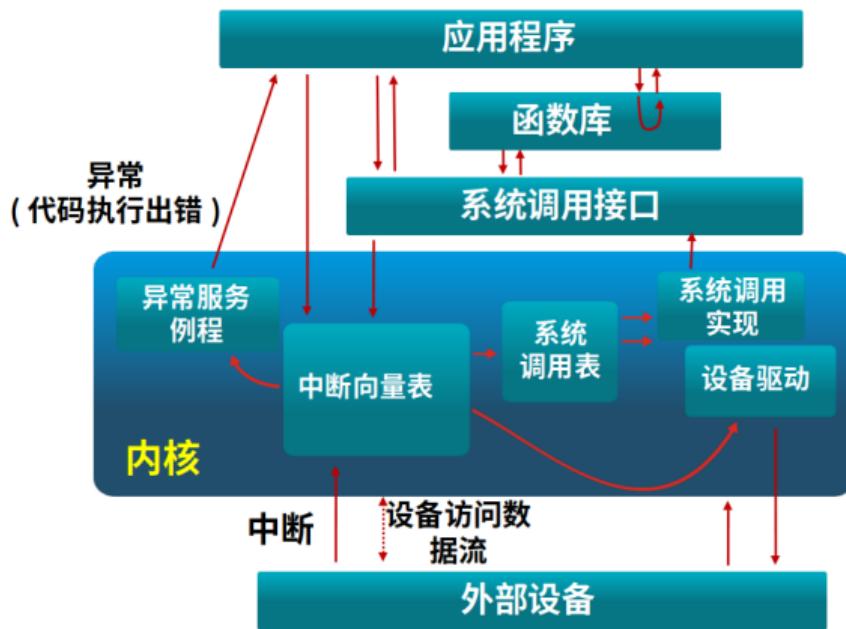


基本概念与原理



- 为什么需要中断、异常和系统调用
 - OS 内核是被信任的第三方
 - OS 内核可以执行特权指令，管理硬件
 - OS 内核提供了各种 Service
- 中断希望解决的问题
 - 当外设连接计算机时，会出现什么现象？
- 异常希望解决的问题
 - 当应用程序处理意想不到的行为时，会出现什么现象？
- 系统调用希望解决的问题
 - 用户应用程序是如何得到系统服务？

基本概念与原理



• 源头

- 中断：外设
- 异常：应用程序意想不到的行为
- 系统调用：应用程序请求 OS 服务

• 响应方式

- 中断：异步
- 异常：同步
- 系统调用：异步或同步

• 处理机制

- 中断：持续，对应用程序透明
- 异常：杀死或者重新执行
- 系统调用：等待和持续

第 3 讲中断、异常和系统调用

第二节：硬件架构支持

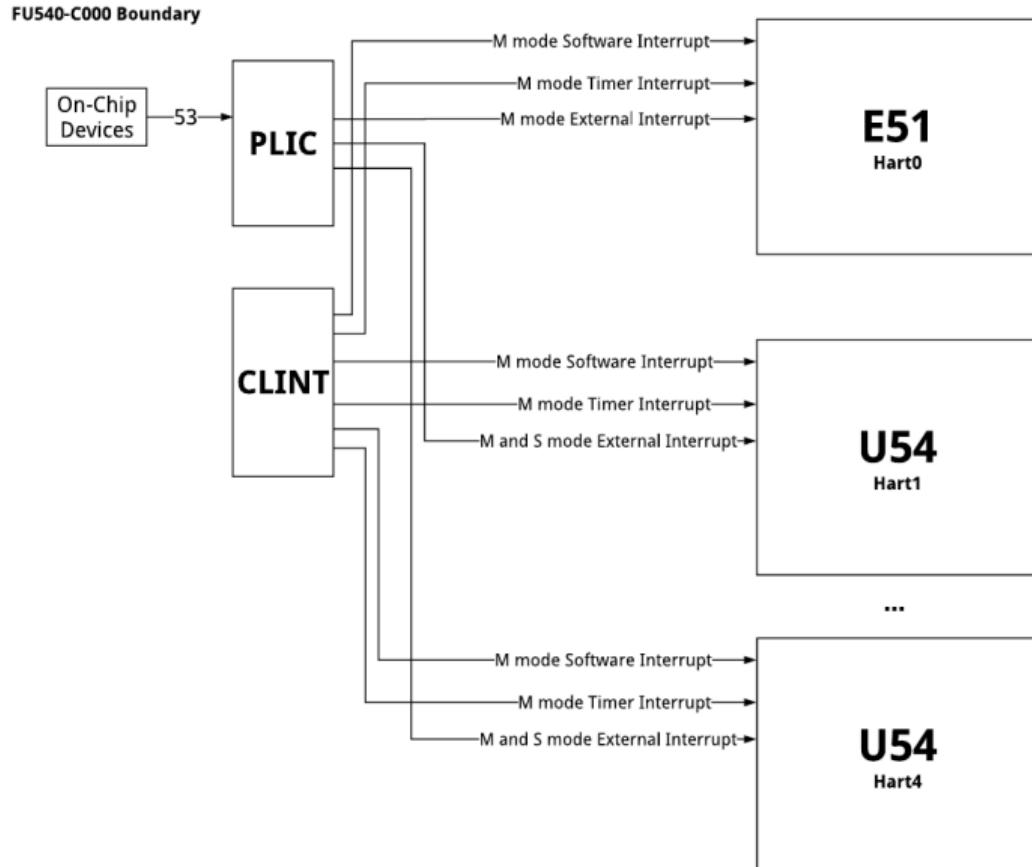
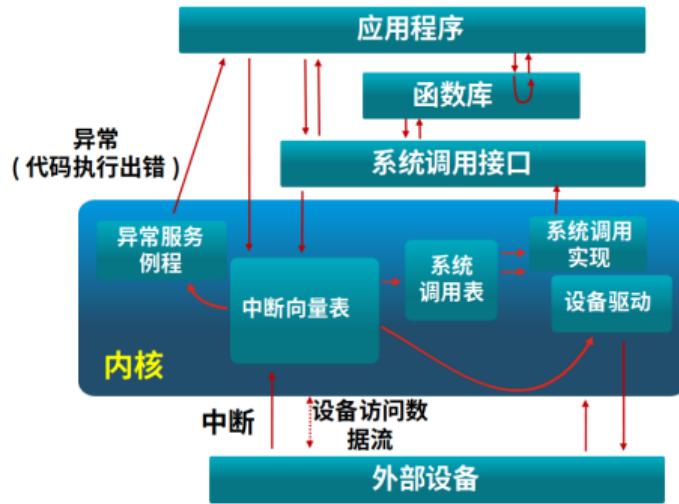
向勇、陈渝

清华大学计算机系

xyong,yuchen@tsinghua.edu.cn

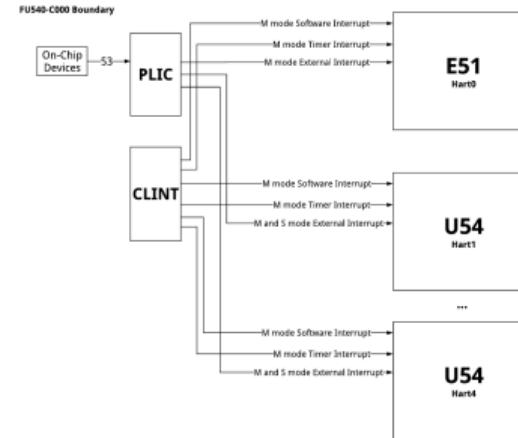
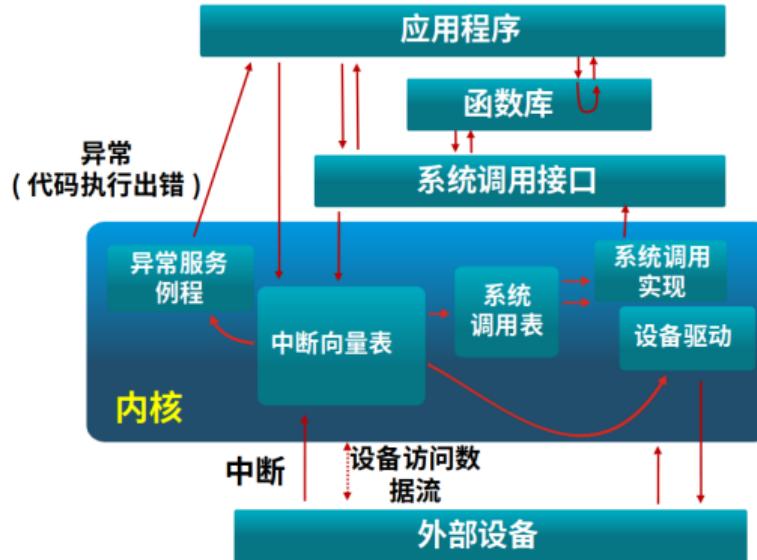
2020 年 5 月 5 日

硬件架构支持



- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

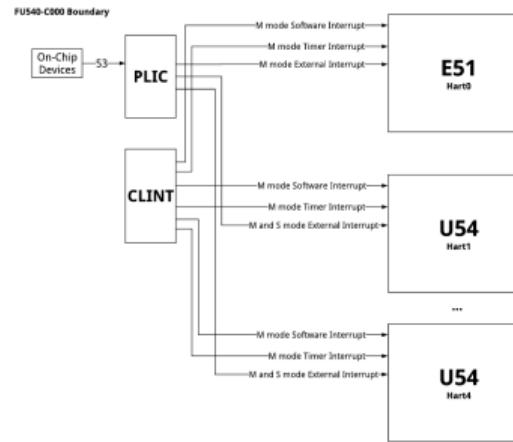
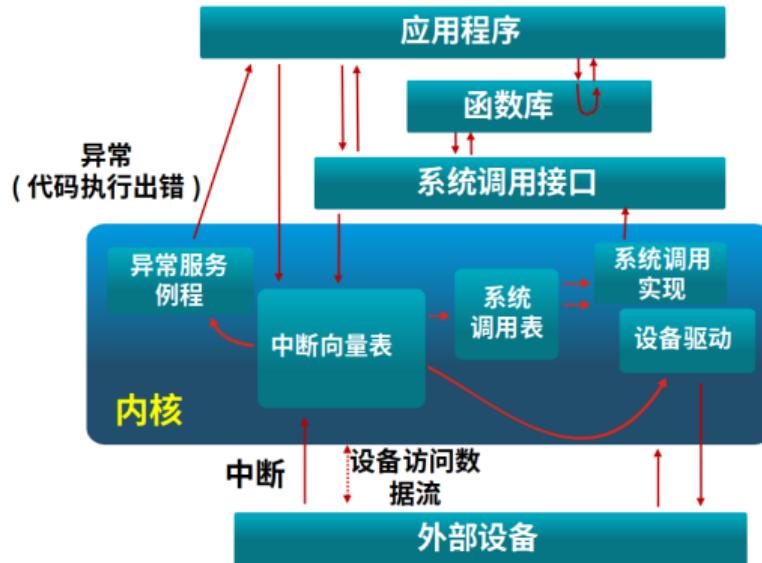
硬件架构支持



三种标准的中断源:

- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)
- 软件中断通过向内存映射寄存器中存数来触发, 如 IPI
- 时钟中断, 如 $stimecmp > stime$
- 由平台级中断控制器引发外部中断

中断处理机制



建立中断机制

- 让 CPU 能响应中断

- sstatus: 保存全局中断使能位
- sie: 指出 CPU 目前能处理或忽略的中断

- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

第 3 讲中断、异常和系统调用

第三节：中断处理机制-Overview

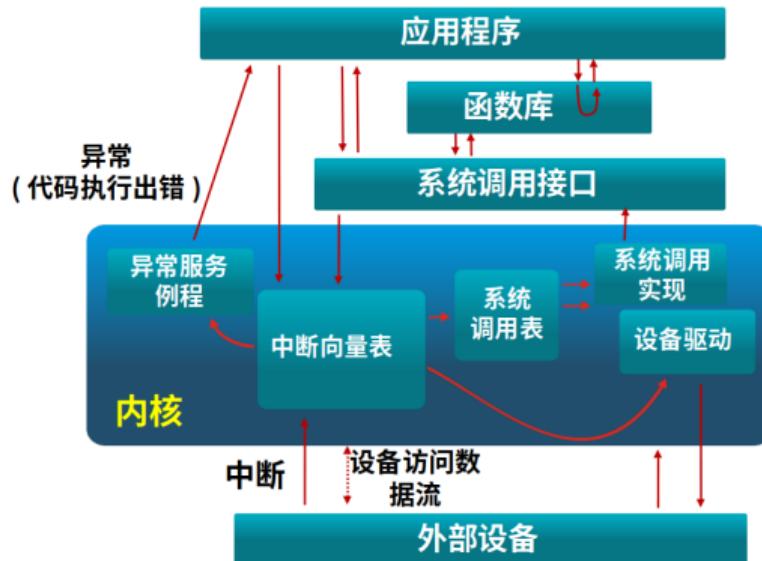
向勇、陈渝

清华大学计算机系

xyong,yuchen@tsinghua.edu.cn

2020 年 5 月 5 日

中断处理机制

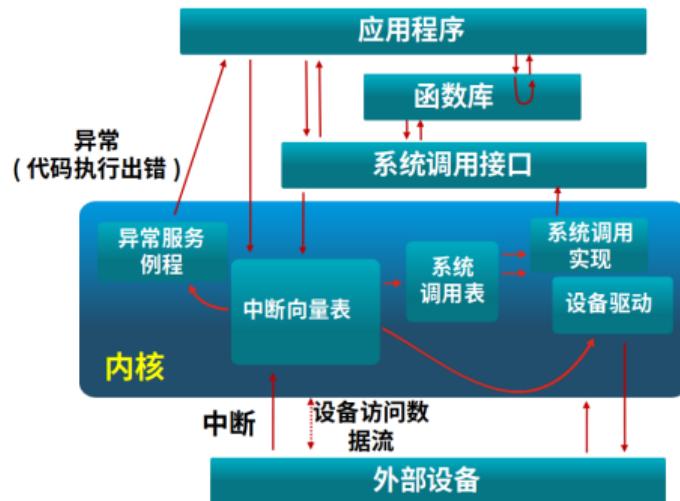


建立中断机制

- 建立中断服务例程
- 让 CPU 能响应中断
- 响应并处理中断
- 保存/恢复现场

- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

中断处理机制



Talk is cheap. Show me the code.

(Linus Torvalds)

- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

第 3 讲中断、异常和系统调用

第三节：中断处理机制-Detail

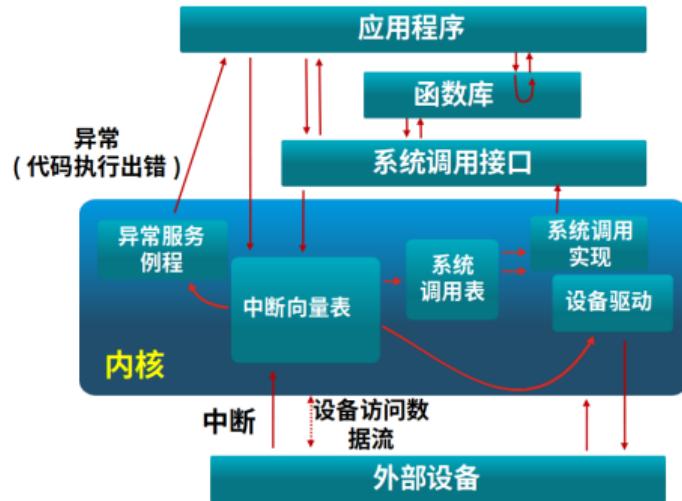
向勇、陈渝

清华大学计算机系

xyong,yuchen@tsinghua.edu.cn

2020 年 5 月 5 日

中断处理机制-让 CPU 能响应中断

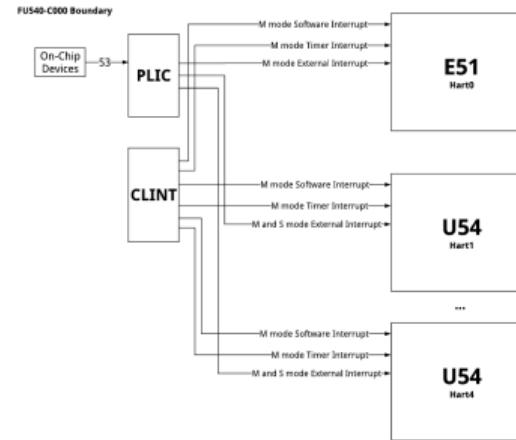
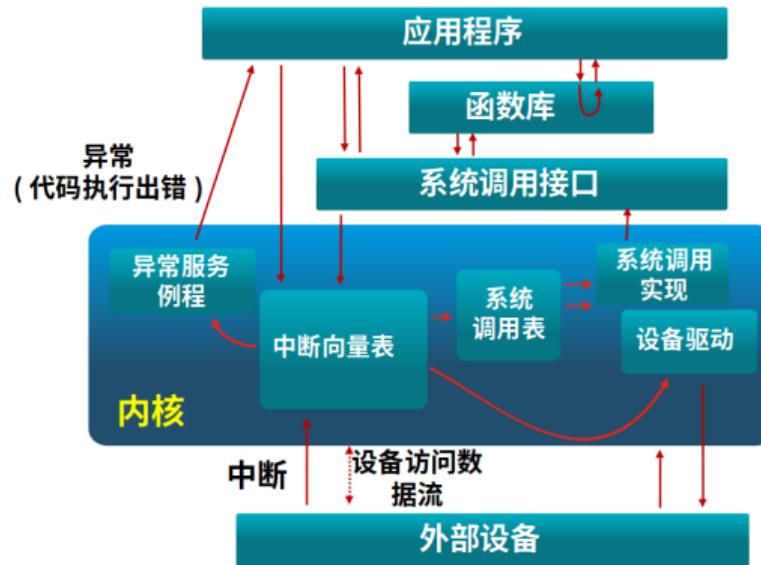


Talk is cheap. Show me the code.

(Linus Torvalds)

- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

中断处理机制-让 CPU 能响应中断



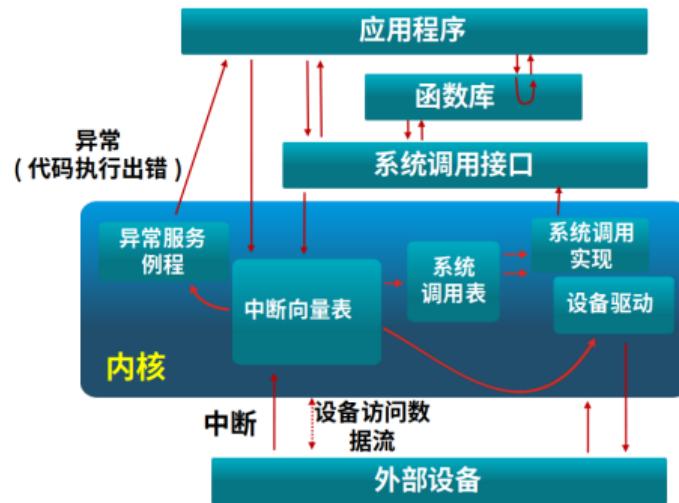
建立中断机制

- 让 CPU 能响应中断

- 硬件: sstatus: 保存全局中断使能位
- 硬件: sie: 指出 CPU 目前能处理 | 忽略的中断
- 硬件: stvec: 中断入口地址

- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

中断处理机制-让 CPU 能响应中断



- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

- 初始化：设置 sie 的 TI 使能 STIE 位
- 初始化：设置 sstatus 的使能中断 SIE 位
- 初始化：实现中断服务总控函数
- 初始化：设置 stvec 指向中断服务总控函数的入口地址

第 3 讲中断、异常和系统调用

第三节：中断处理机制-Detail

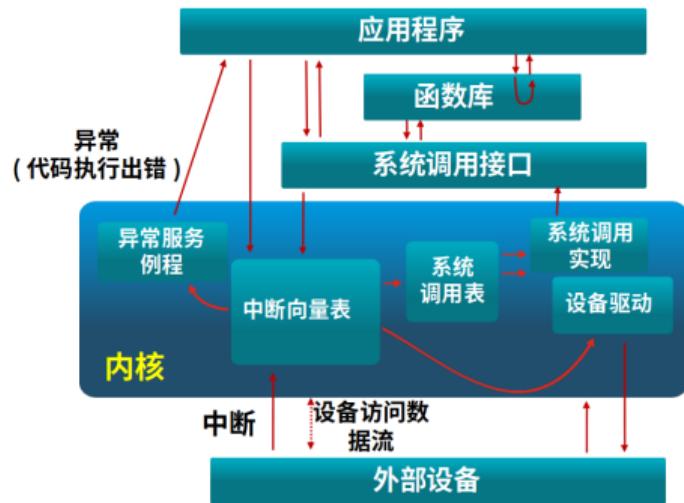
向勇、陈渝

清华大学计算机系

xyong,yuchen@tsinghua.edu.cn

2020 年 5 月 5 日

中断处理机制-建立中断服务例程

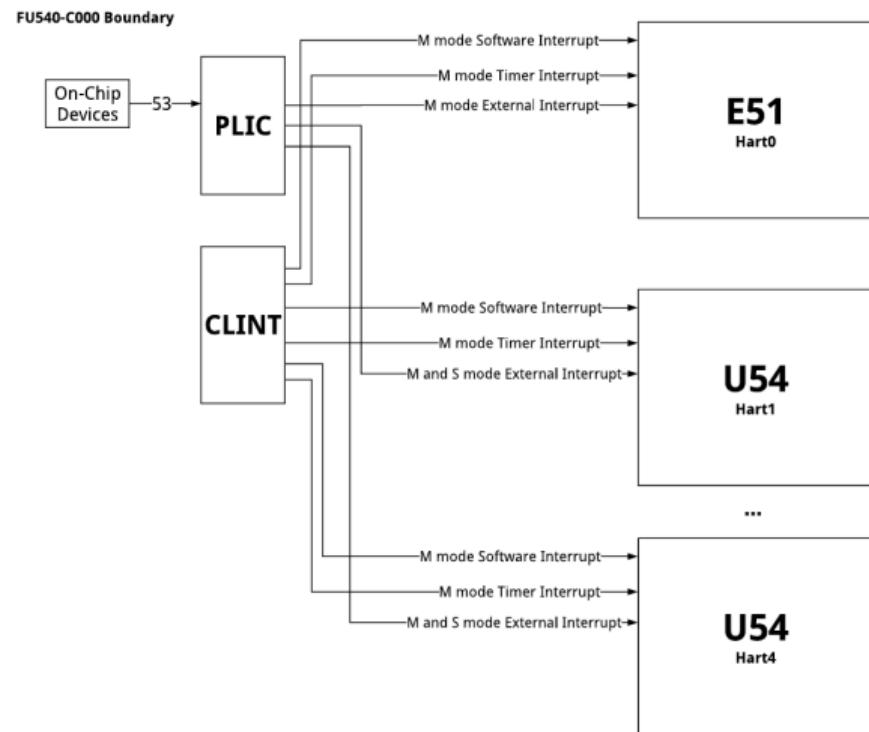
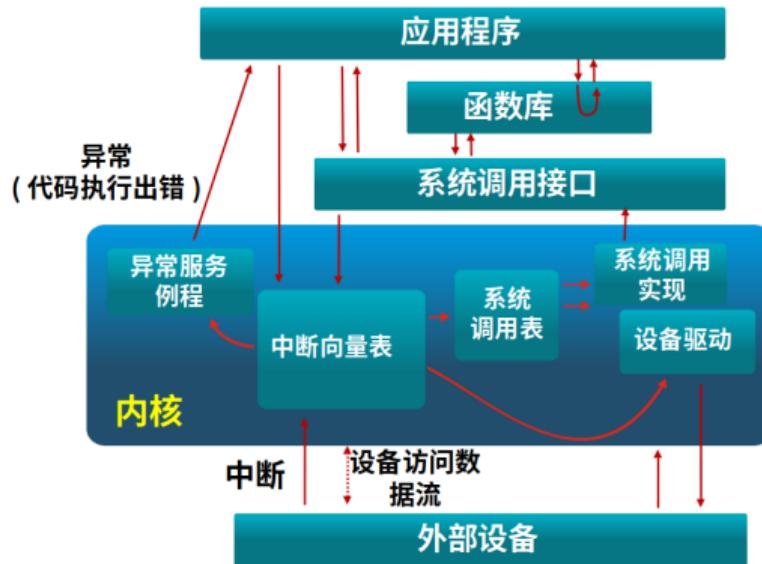


Talk is cheap. Show me the code.

(Linus Torvalds)

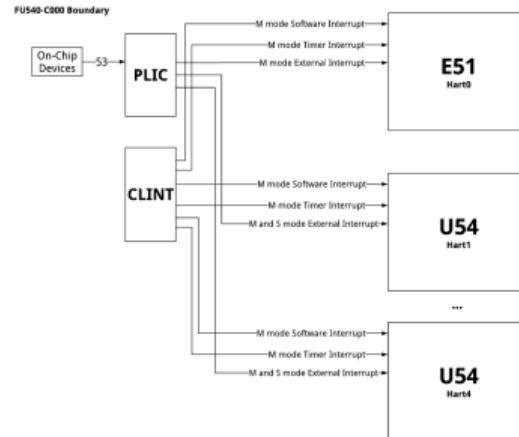
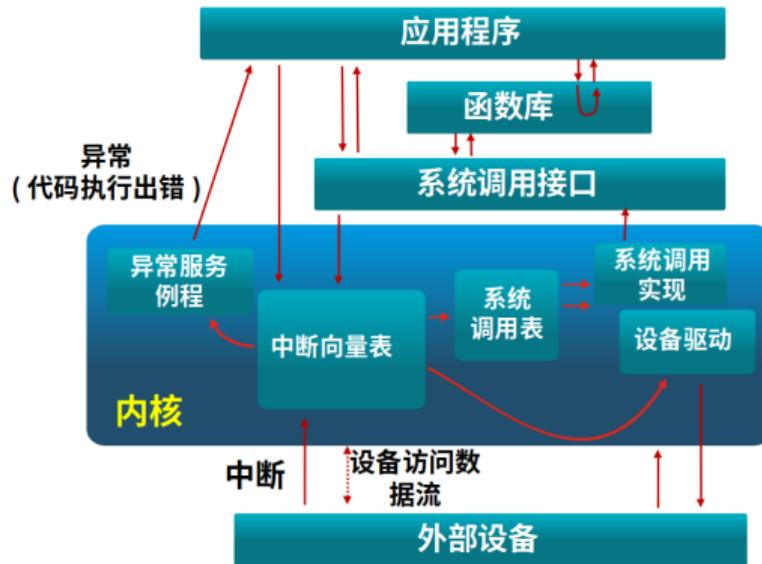
- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

中断处理机制-建立中断服务例程



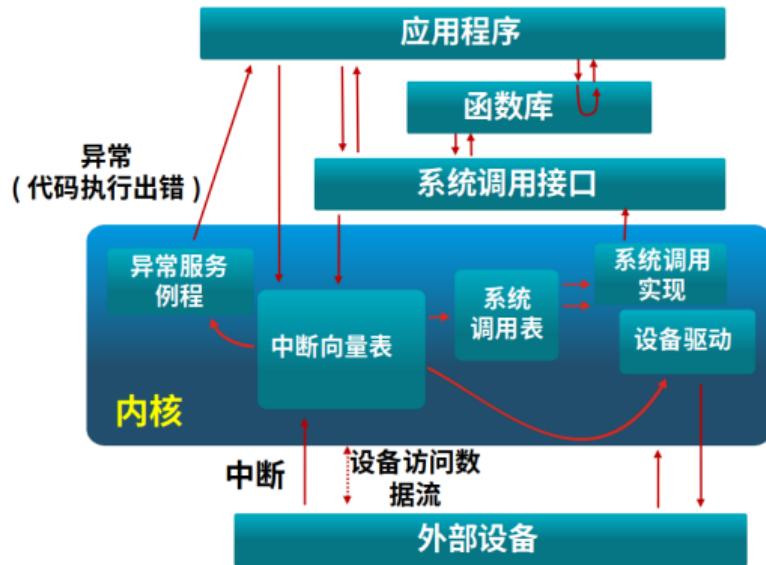
- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

中断处理机制-建立中断服务例程



- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

中断处理机制-建立中断服务例程



- 初始化：设置时钟中断触发次数
- 初始化：设置 sie 的 TI 使能 STIE 位
- 服务例程：调用 OpenSBI 提供的接口设置下次时钟中断触发时间

- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

第 3 讲中断、异常和系统调用

第三节：中断处理机制-Detail

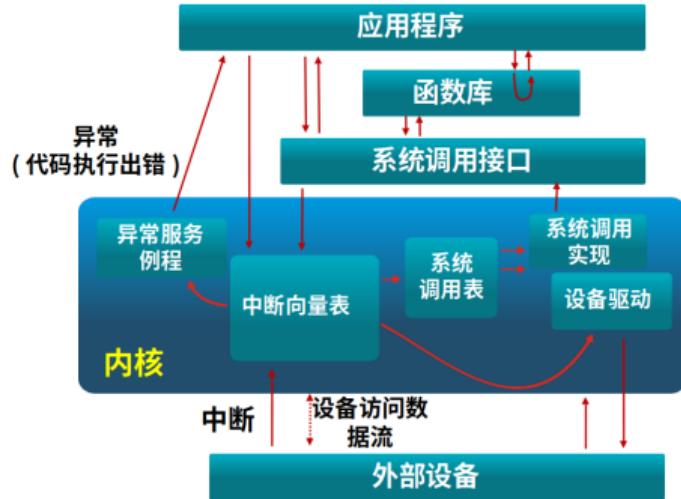
向勇、陈渝

清华大学计算机系

xyong,yuchen@tsinghua.edu.cn

2020 年 5 月 5 日

中断处理机制-保存恢复现场

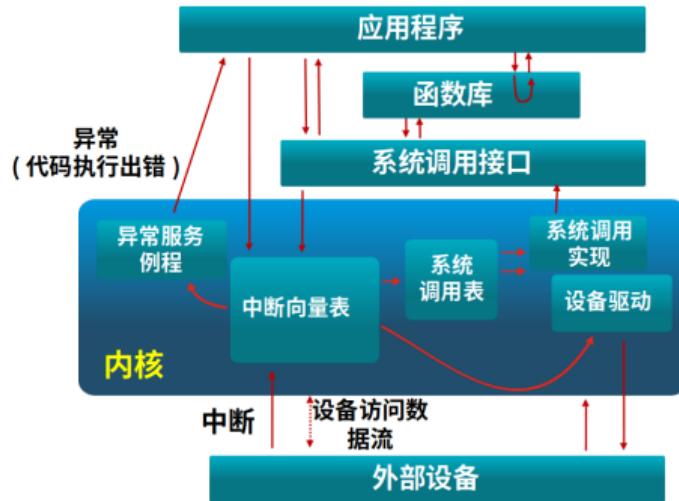


Talk is cheap. Show me the code.

(Linus Torvalds)

- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

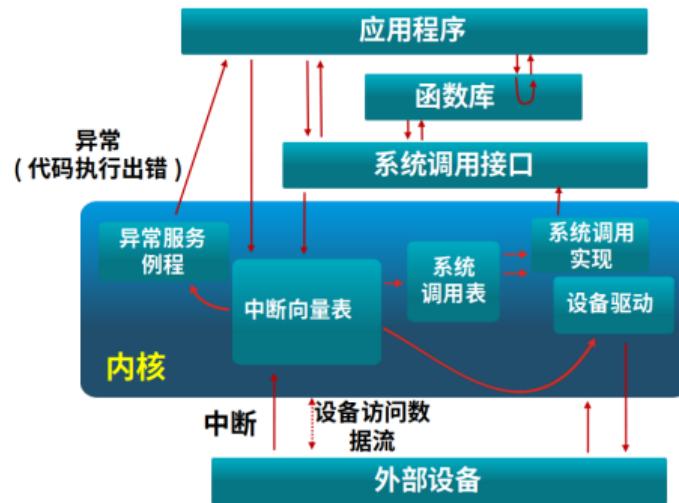
中断处理机制-保存恢复现场



- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

- 还需为被中断的程序保存和恢复当时程序运行时的上下文：
- SAVE_ALL 寄存器

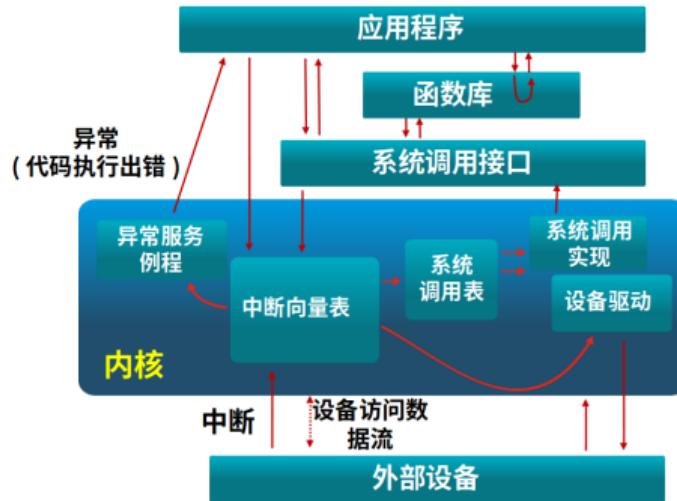
中断处理机制-保存恢复现场



- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

- 要保存和恢复当时程序运行时的上下文：
- x[0–32]：通用寄存器
- sstatus：系统系状态
- sepc：触发异常/中断的指令地址
- scause：指示发生异常/中断的种类
- stval：保存了发生异常/中断的附加信息

中断处理机制-还原整个过程



- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

产生中断后：

- 硬件设置：

- sepc：保存中断的指令地址
- pc：设置为 stvec
- scause：设置中断的来源
- sstatus：SIE 位置零以禁用中断
- stval：保存了中断相关的附加信息

- 软件保存：

- x[0–32]：通用寄存器
- pc：设置为 stvec
- scause：设置中断的来源
- sstatus：SIE 位置零以禁用中断
- stval：保存了中断相关的附加信息

第 3 讲中断、异常和系统调用

第三节：中断处理机制-Summary

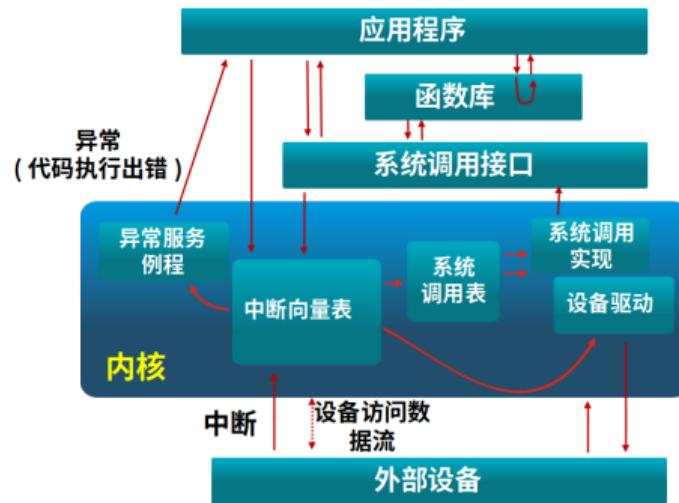
向勇、陈渝

清华大学计算机系

xyong,yuchen@tsinghua.edu.cn

2020 年 5 月 5 日

中断处理机制-中断执行过程

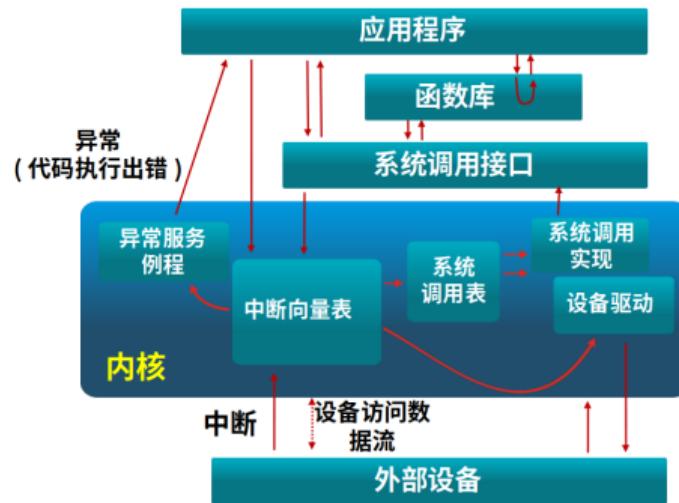


- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

产生中断后：

- 硬件设置：
 - sepc：保存中断的指令地址
 - pc：设置为 stvec
 - scause：设置中断的来源
 - sstatus：SIE 位置零以禁用中断
 - stval：保存了异常相关的附加信息
- 软件保存被打断现场：
 - x[0–32]：通用寄存器
 - sepc：保存中断的指令地址
 - scause：设置中断的来源
 - sstatus：SIE 位置零以禁用中断
 - stval：保存了异常相关的附加信息

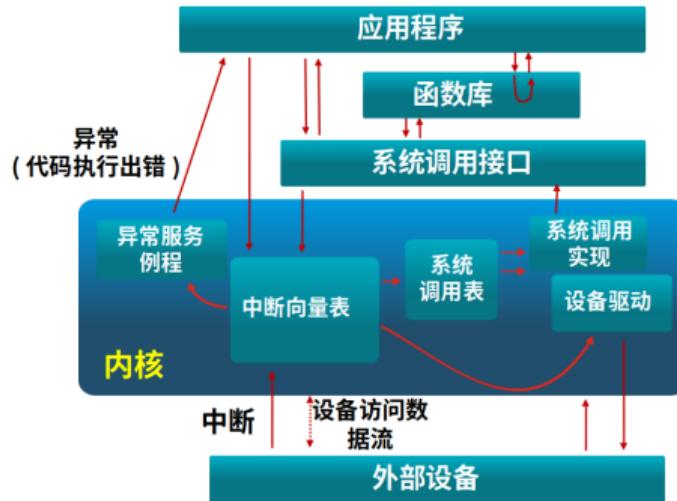
中断处理机制-中断执行过程



产生中断后：

- Core Local Interruptor (CLINT)
 - Platform-Level Interrupt Controller (PLIC)
- 硬件设置
- 软件保存被打断现场
- 执行软件实现的中断服务例程
- 软件恢复被打断现场
- 继续执行

中断嵌套



- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

- 硬件中断服务例程可被打断
 - 不同硬件中断源可能在硬件中断处理时出现
 - 中断请求会保持到 CPU 做出响应
 - 硬件中断服务例程中需要临时禁止中断请求

第 3 讲中断、异常和系统调用

第三节：中断处理机制-Detail

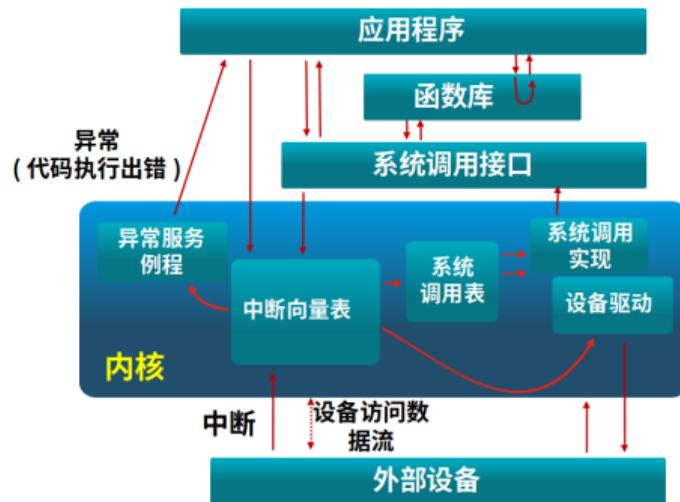
向勇、陈渝

清华大学计算机系

xyong,yuchen@tsinghua.edu.cn

2020 年 5 月 5 日

中断处理机制

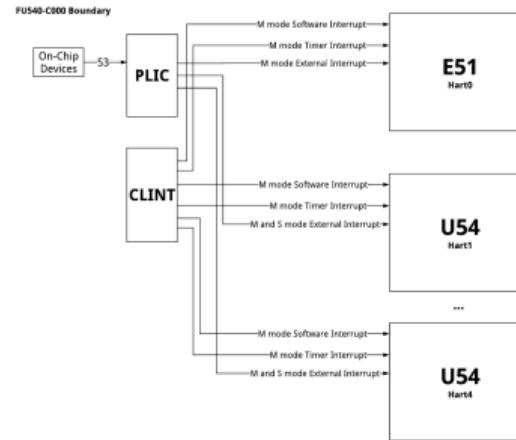
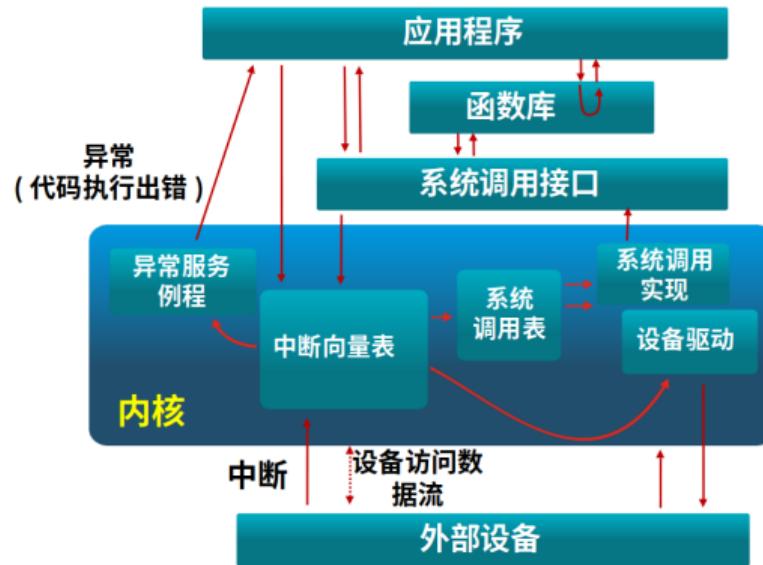


Talk is cheap. Show me the code.

(Linus Torvalds)

- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

中断处理机制-让 CPU 能响应中断



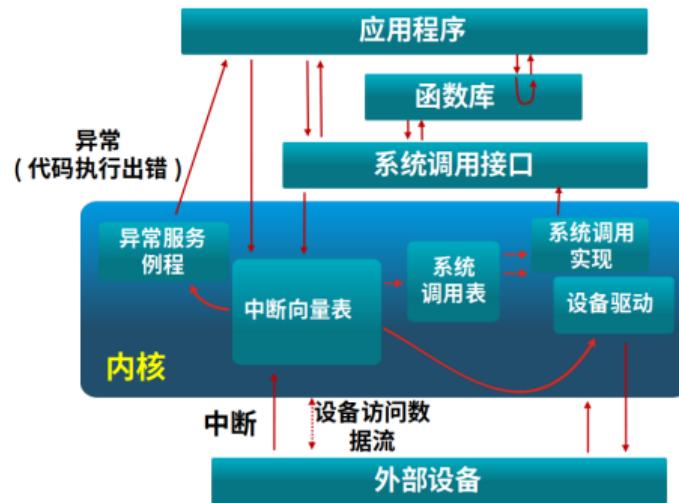
建立中断机制

- 让 CPU 能响应中断

- 硬件: sstatus: 保存全局中断使能位
- 硬件: sie: 指出 CPU 目前能处理 | 忽略的中断
- 硬件: stvec: 中断入口地址

- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

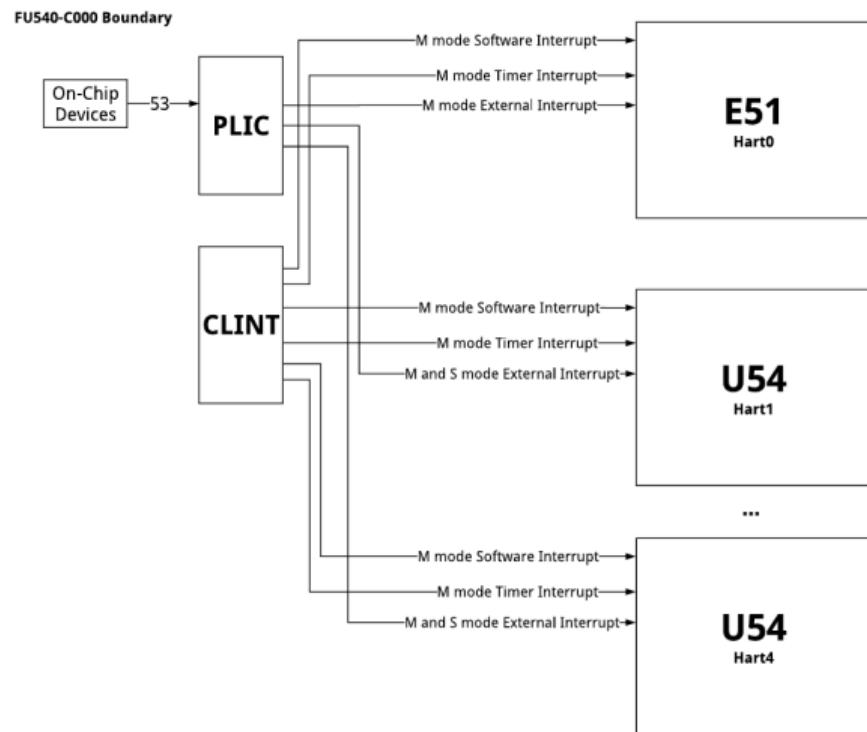
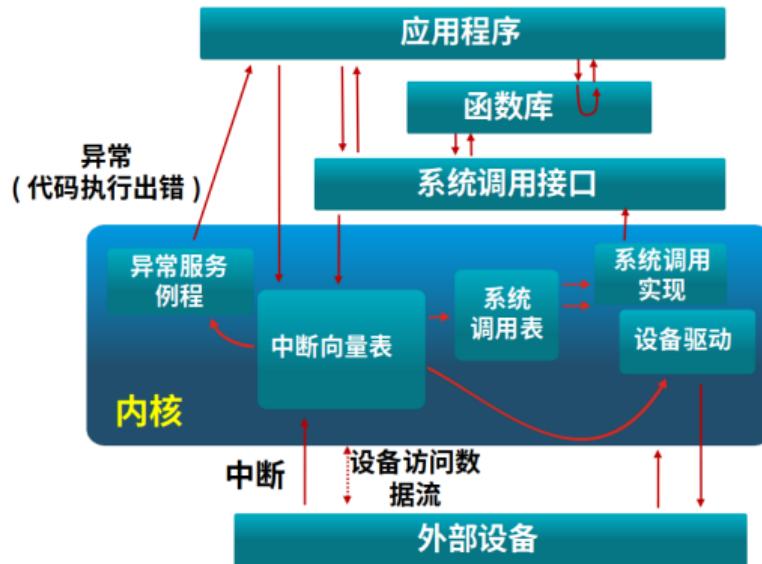
中断处理机制-让 CPU 能响应中断



- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

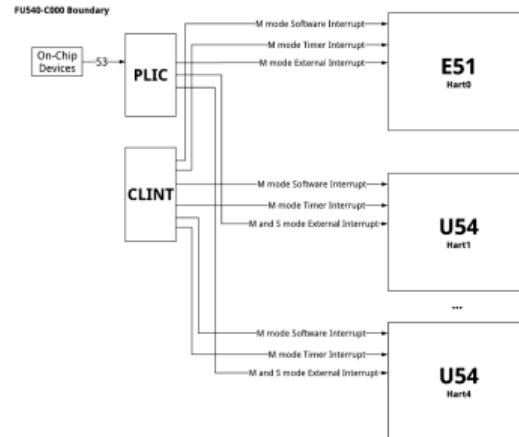
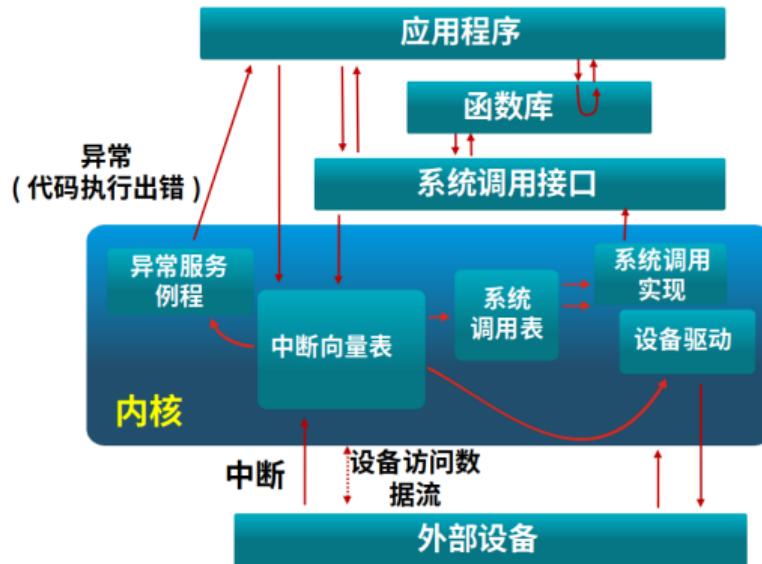
- 初始化：设置 sie 的 TI 使能 STIE 位
- 初始化：设置 sstatus 的使能中断 SIE 位
- 初始化：实现中断服务总控函数
- 初始化：设置 stvec 指向中断服务总控函数的入口地址

中断处理机制-建立中断服务例程



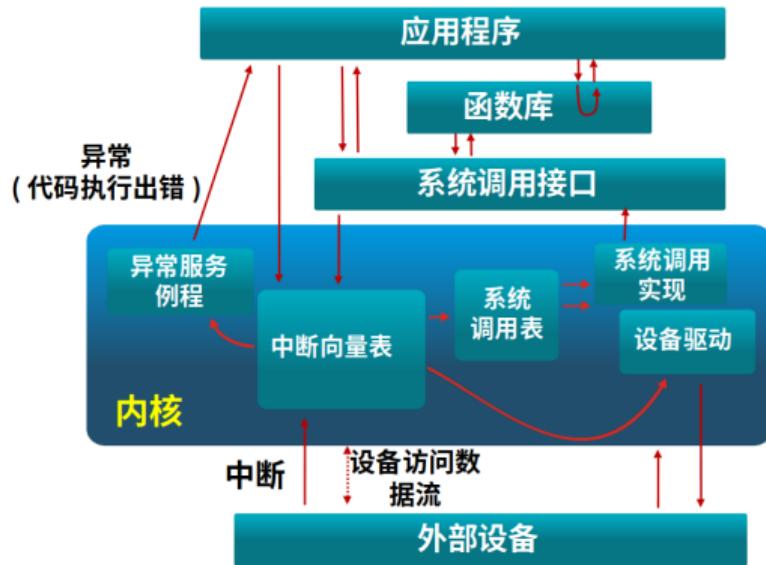
- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

中断处理机制-建立中断服务例程



- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

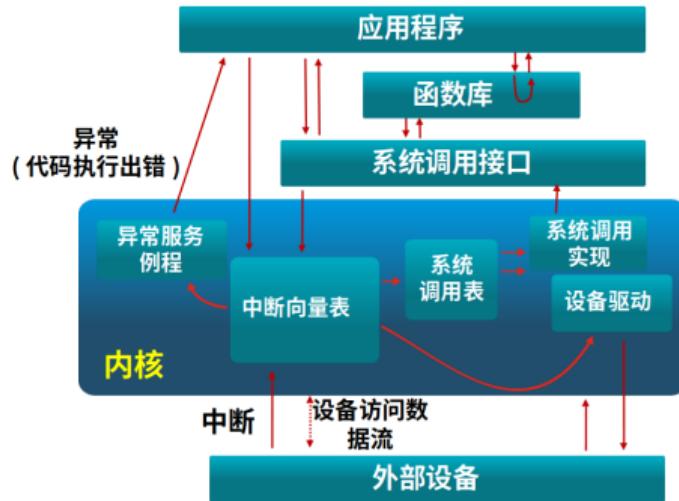
中断处理机制-建立中断服务例程



- 初始化：设置时钟中断触发次数
- 初始化：设置 sie 的 TI 使能 STIE 位
- 服务例程：调用 OpenSBI 提供的接口设置下次时钟中断触发时间

- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

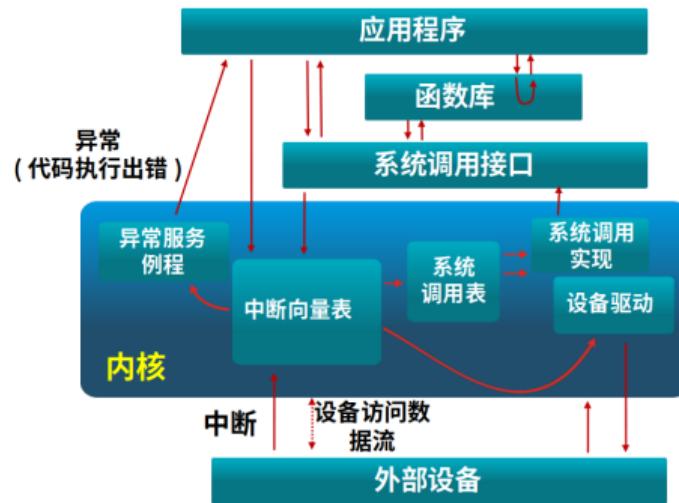
中断处理机制-保存恢复现场



- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

- 还需为被中断的程序保存和恢复当时程序运行时的上下文：
- SAVE_ALL 寄存器

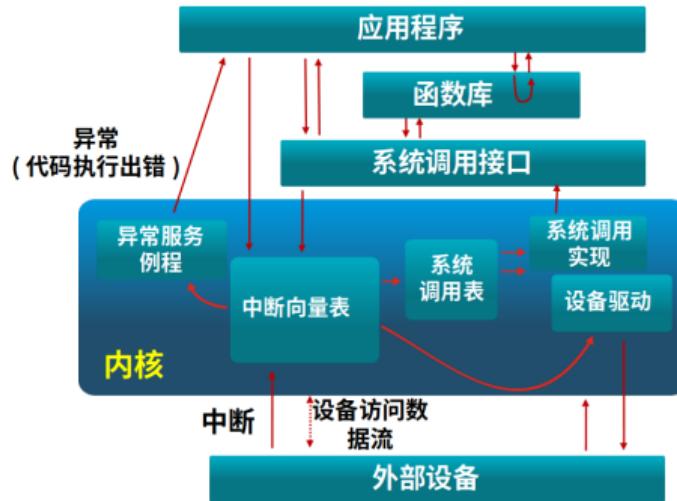
中断处理机制-保存恢复现场



- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

- 要保存和恢复当时程序运行时的上下文：
- $x[0-32]$: 通用寄存器
- sstatus: 系统系状态
- sepc: 触发异常/中断的指令地址
- scause: 指示发生异常/中断的种类
- stval: 保存了发生异常/中断的附加信息

中断处理机制-还原整个过程

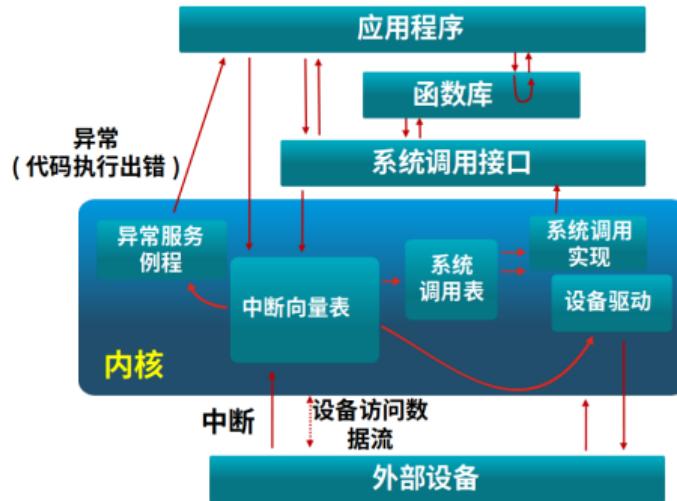


- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

产生中断后：

- 硬件设置：
 - sepc：保存中断的指令地址
 - pc：设置为 stvec
 - scause：设置中断的来源
 - sstatus：SIE 位置零以禁用中断
 - stval：保存了中断相关的附加信息
- 软件保存：
 - x[0–32]：通用寄存器
 - pc：设置为 stvec
 - scause：设置中断的来源
 - sstatus：SIE 位置零以禁用中断
 - stval：保存了中断相关的附加信息

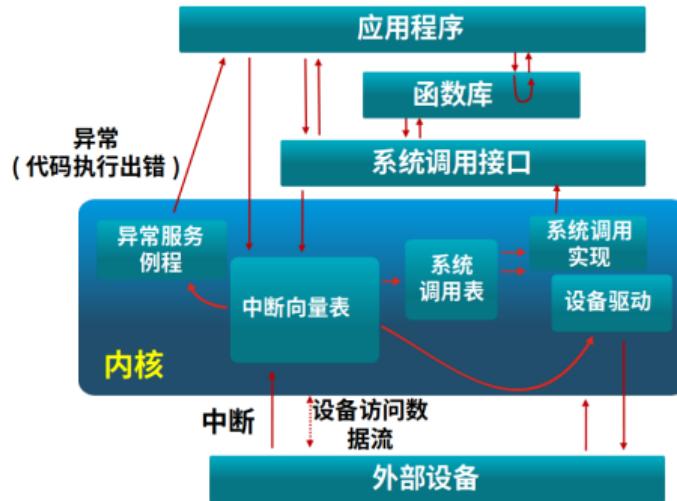
中断处理机制-还原整个过程



产生中断后：

- 硬件设置
 - 软件保存被打断现场
 - 执行软件实现的中断服务例程
 - 软件恢复被打断现场
 - 继续执行
-
- Core Local Interruptor (CLINT)
 - Platform-Level Interrupt Controller (PLIC)

中断嵌套



- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)

- 硬件中断服务例程可被打断
 - 不同硬件中断源可能在硬件中断处理时出现
 - 中断请求会保持到 CPU 做出响应
 - 硬件中断服务例程中需要临时禁止中断请求

第 3 讲中断、异常和系统调用

第四节：系统调用

向勇、陈渝

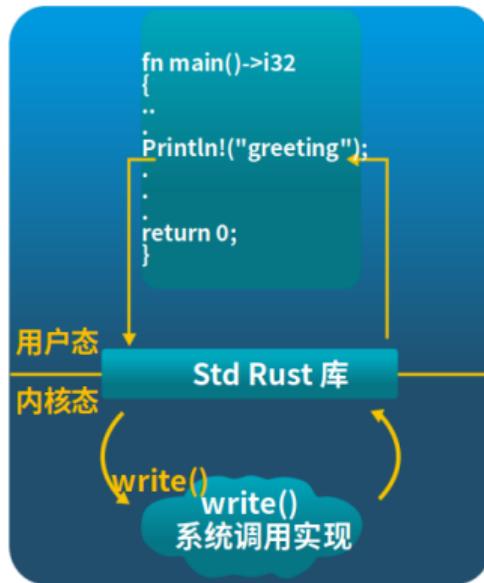
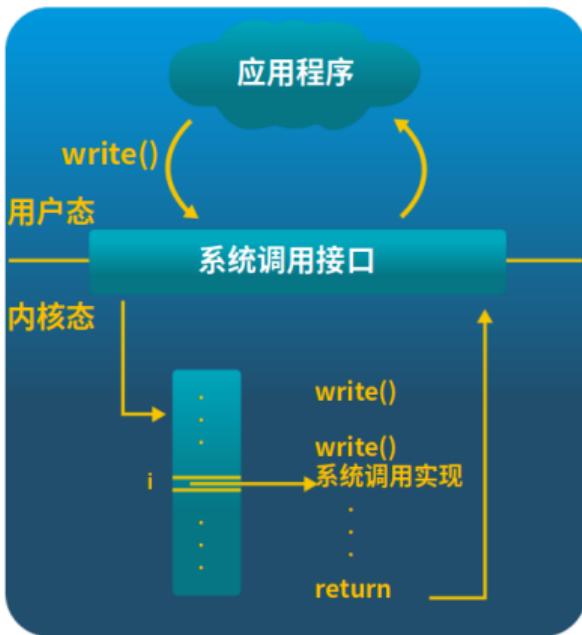
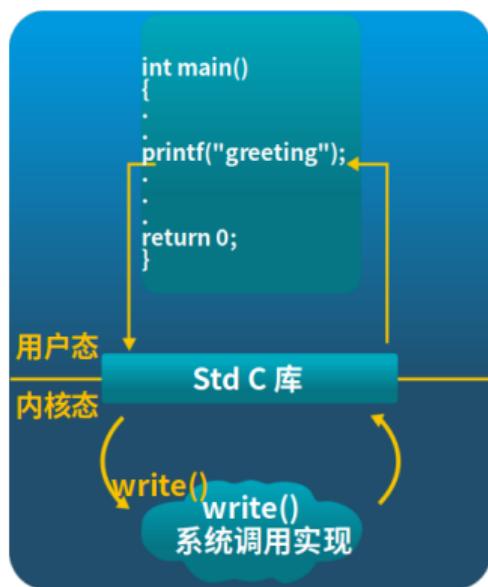
清华大学计算机系

xyong,yuchen@tsinghua.edu.cn

2020 年 5 月 5 日

系统调用概貌

- 应用程序要输出字符串时，会触发系统调用 write()



系统调用概貌

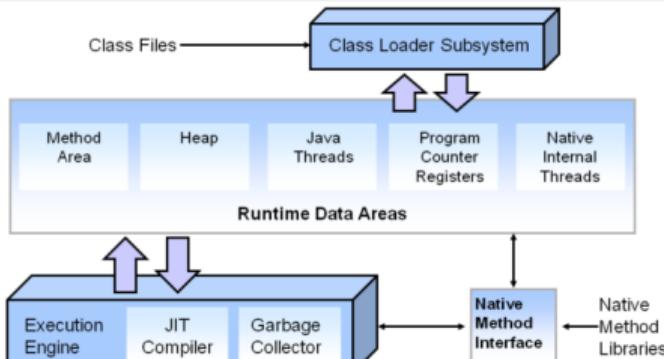
Win32 API

```
32 void GameActivate(HWND hWnd)
33 {
34     HDC hDC;
35     RECT rect;
36 }
```

POSIX

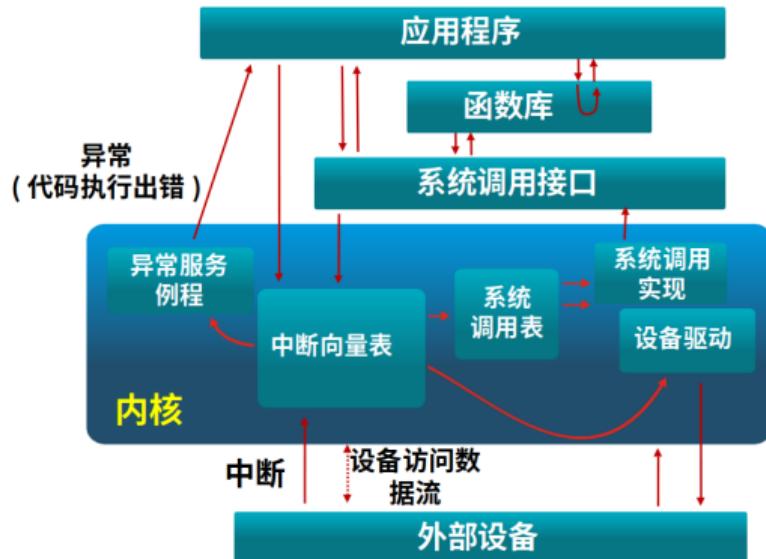
stands for

Portable Operating System Interface based on UNIX



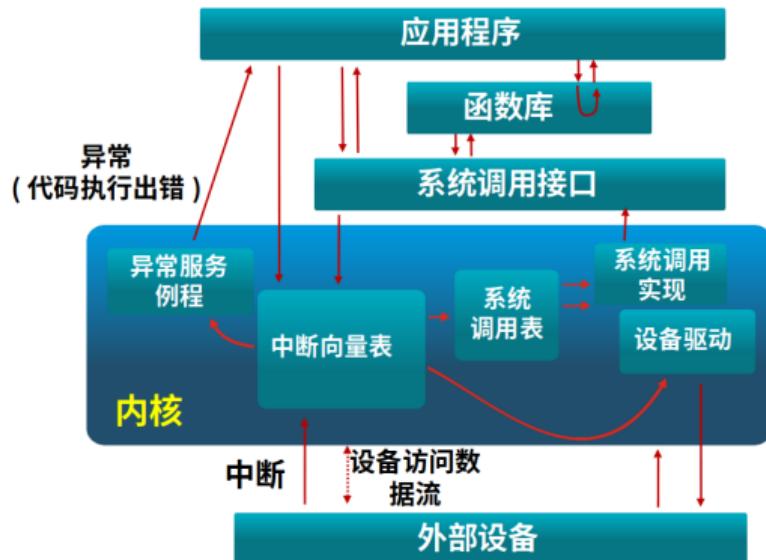
- 操作系统服务的应用编程接口
 - 通常由高级语言编写 (C、go 和 rust 等)
 - 程序通常访问高层次的 API 接口
- 三种最常用的应用程序编程接口
 - Win API: 用于 Windows 32/64
 - POSIX API: 用于 POSIX-based OS
 - Java API: 用于 JAVA 虚拟机 (JVM)

系统调用的实现概述



- 用户态: 通过运行库来管理
- 内核态: 对应系统调用号
- 内核态: 实现系统调用功能

函数调用和系统调用的不同处



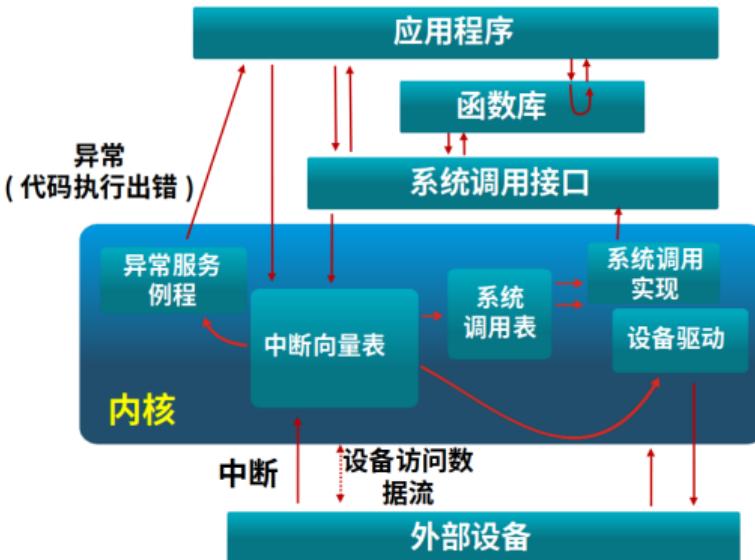
- 系统调用

- RISC-V 中，ecall 和 sret 指令用于系统调用
- 堆栈切换和特权级的转换

- 函数调用

- RISC-V 中，call 和 ret 指令用于系统调用
- 无堆栈切换和特权级的转换

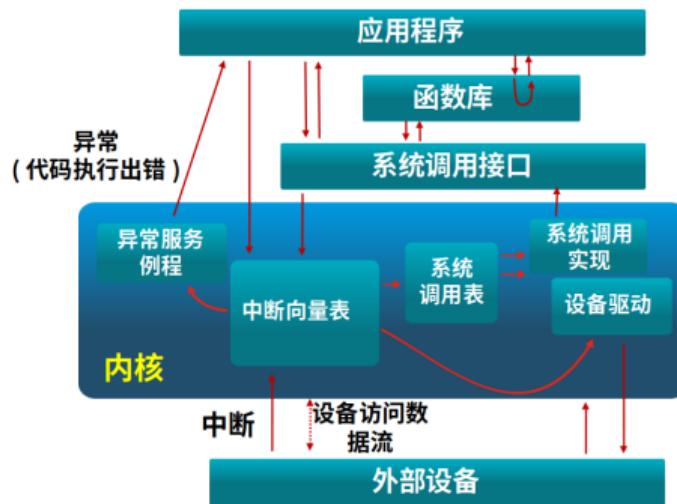
系统调用的开销



- 开销超过函数调用
- 开销:

- 切换内核堆栈
- 验证参数
- 可能切换页表
- 需要拷贝数据

系统调用机制-具体实现



应用发起请求

- std lib 发出系统调用请求
 - 发出设置系统调用号和参数，发出 ecall
- 硬件设置
 - sepc：保存请求后的指令地址
 - pc：设置为 stvec
 - scause：设置为 ecall from u-mode
 - sstatus：SIE 位置零以禁用中断
 - stval：保存了相关的附加信息
- 软件保存被打断现场
- 执行软件实现的中断服务例程
- 软件恢复现场
- 应用继续执行

- Core Local Interruptor (CLINT)
- Platform-Level Interrupt Controller (PLIC)