2.2 中断技术

- 2.2.1 中断概念
- 2.2.2 中断源分类
- 2.2.3 中断和异常的响应及服务
- 2.2.4 中断事件处理原则
- 2.2.5 中断优先级和多重中断

- 中断是现代操作系统实现并发性的基础之一

2.2.1 中断基本概念（1）

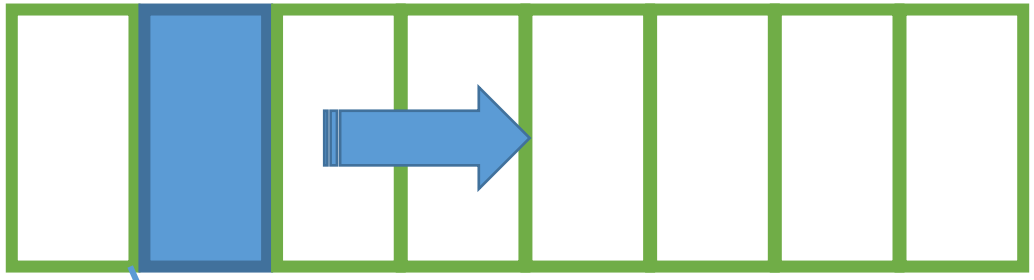
- 目的：请求系统服务，实现并行工作，处理突发事件，满足实时要求，都需要打断处理器正常的工作
- 过程：中断是指程序执行过程中，当发生某个事件时，中止CPU上现行程序的运行，引出处理该事件的程序执行的过程
- 实现方式：在提供中断装置的计算机系统中，在每两条指令之间或某些特殊指令执行期间都检查是否有中断事件发生，若无则立即执行下一条或继续执行，否则响应事件并转去处理中断

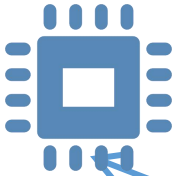
不同操作系统下中断共性

- 当中断事件发生后，它能改变处理器内操作执行的顺序

中断：interrupt

CPU:





中断装置

中断源

2.2.1 中断基本概念（2）

- 中断装置：发现中断源并产生中断的硬件，包括：
  - 中断逻辑线路
  - 中断寄存器
- 中断源：引起中断的事件

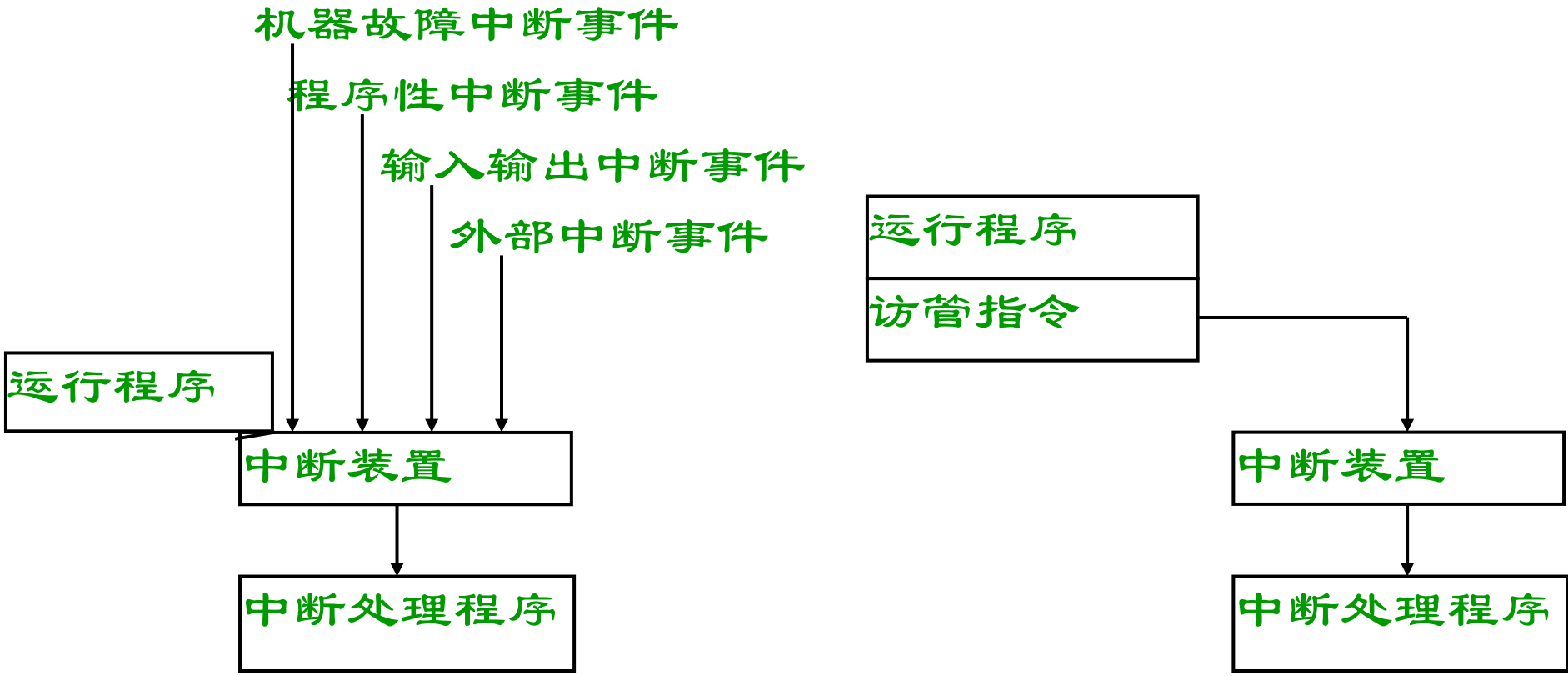
  - 电源故障，主存出错，通路校验错误 …
  - 非法操作码，定点溢出，除数为0
  - 时钟中断，它机中断
  - 数据传输结束，启动失败

  强迫性中断

  - 程序执行“访管指令”

  自愿性中断

2.2.2 中断源分类（1）



按照中断性质与激活方式分类（IBM大型机）

2.2.2 中断源分类（2）

- 按中断事件性质和激活方式 （IBM 大型机）
  - 强迫性中断 / 自愿性中断

- 按中断事件来源和实现手段 （Intel x86 PC）
  - 硬中断
    - 外中断（异步中断/中断）
    - 内中断（同步中断/异常）
  - 软中断
    - 信号
    - 软件中断

2.2.2（3）硬中断

- 由硬件设施来产生的中断请求
  - 外中断（异步中断/中断）：来自处理器和主存之外
    - 与当前程序无关，不可确定事件的发生时间
    - 细分为 可屏蔽中断和不可屏蔽中断，具体不同优先级
    - 时钟中断，键盘中断，它机中断，设备中断 …
  - 内中断（同步中断/异常）：来自处理器内部的
    - 在程序执行过程中产生
    - 不可屏蔽，需要立刻响应
    - 硬件故障：电源失效，奇偶校验错误 …
    - 程序性异常：非法操作，地址越界，除数为0 …
    - 访管中断：执行系统调用 …

2.2.2 (3) 硬中断

- 中断（外中断）与 异常（内中断）的区别
  - 中断是由与现行指令无关的中断信号触发
    - 系统不能确定中断事件的发生时间
    - 中断与CPU是异步的
    - CPU对中断的响应完全是被动的
    - 中断的发生于CPU模式无关
    - 通常在两条指令之间才能响应中断
    - 通常中断处理程序所提供的服务不是当前进程所需要的

2.2.2 (3) 硬中断

- 中断（外中断）与 异常（内中断）的区别
  - 异常是由处理器控制单元产生，源于现行程序执行指令过程中检测到例外
    - 异常与CPU是同步的
    - 一条指令执行期间允许响应异常，而且允许多次响应异常
    - 异常处理程序提供的服务是为当前进程所用的
    - 大部分异常发生在用户态
    - 异常会嵌套中断，但中断不会嵌套异常

- 产生异常时，硬件并不清除中断标志位，此时还允许外部硬件中断
- 产生中断时，硬件将立即清除中断标志位，以禁止其他硬件中断

2.2.2 (3) 硬中断

- Linux 系统中异常的分类与区别
  - 编程异常（Programmed exception）
    - 实现系统调用 （主动进行核心态） 执行当前下一条指令
  - 故障(fault)
    - 系统捕获的可恢复错误 （页面故障）返回执行当前指令
  - 终止（Abort）
    - 致命的不可恢复错误 （奇偶校验错误）
  - 陷阱(trap)
    - 特定的调试指令（断点）执行当前下一条指令

2.2.2（4）软中断

- 不必由硬件产生中断源而引发的中断
- 软件对硬中断机制的模拟，实现宏观的异步执行

  - 信号机制 （进程间通信）
    - 信号发送者相当于中断源
    - 信号接收者则是另一个进程

  - 软件中断 （线程调度，延迟调用和异步执行）
    - Linux bottom half (softirq)
    - Windows Dispathch/DPC, APC

2.2.2（5）软硬中断区别

- 中断（硬件，异步）：
  - 用于外部设备对CPU的中断，转向中断处理程序执行
- 异常（硬件，同步）：
  - 因指令执行不正确而中断，转向异常处理程序执行
- 软件中断（软件，进程间调度）：
  - 决定不同进程、线程间的调度执行顺序
- 信号（软件，进程间通信）：
  - 用于内核或进程对某个进程的重点，向进程同时某个特定事件发生或迫使进程执行信号处理程序

2.2.2（6）中断与信号

- 中断（硬件）vs. 信号（软件）相同：
  - 概念一致，CPU接收中断，进程接收信号
  - 都是异步执行
    - 中断：每条指令结束时安排中断查询
    - 信号：异常处理末尾或者时钟中断结束
  - 均采用“向量表”机制实现中断处理
  - 均采用“屏蔽”措施

2.2.2（6）中断与信号

- 中断（硬件）vs. 信号（软件）不同：
  - 前者由硬件和软件结合实现
  - 后者则完全由软件实现

  - 中断向量表和处理程序位于系统空间
  - 信号向量表位于系统空间，信号处理程序则由应用程序提供，并在用户空间执行

2.2.3中断和异常的响应及服务

- 区分中断和异常：
  - 中断：指令执行结束后检查中断寄存器是否有中断事件发生
    - 若无或者中断被屏蔽，继续执行程序的后续指令
    - 若有则暂停当前程序的执行，转向内核的中断处理程序
  - 异常：它是在执行指令时由指令本身发生的，指令的实现或执行逻辑一旦发生异常情况，立即转向内核的异常处理程序

2.2.3中断和异常的响应及服务

- 所有计算机系统都采用硬件和软件（硬件中断装置和软件中断处理程序）结合的方法实现中断处理
  1. 发现中断源：未屏蔽中断时，由硬件发现中断/异常事件
     - 若存在多个中断源，则根据中断优先级先后响应
  2. 保护现场：保存PSW至核心栈
  3. 转向中断/异常处理事件：基于中断向量，查询中断向量表，获取保存在系统空间的中断/异常处理程序
  4. 恢复现场：恢复PSW，返回中断点继续执行
     - 访管指令：执行下一条指令（当用户进程取得CPU时间时）
     - 致命故障：直接结束进程
     - 页面故障：返回发生异常的指令，继续进行

2.2.3中断和异常的响应及服务

- IBM PC 通常在计算机内存的低地址处开辟中断向量表
  - 表中每一项称为一个中断向量，
  - 存放了一个中断处理程序的入口地址及相关信息
- 当发现中断源并响应中断时，中断装置将把先行PSW内容压进堆栈，接着再把指令指针IP和代码段基地址内容也压入堆栈，这样就保存了原运行程序的状态

2.2.4 中断事件处理原则

中断

- 硬件故障中断
- I/O中断
- 时钟中断

异常

- 程序性中断
- 访问中断

2.2.4 中断事件处理原则

- 硬件故障中断事件
  - 电源故障、内存故障、设备故障，等等
  - 这种事件是由硬件故障产生的，排除故障须进行人工干预
- 中断处理能做的工作是：
  - 保护现场
  - 防止故障蔓延
  - 报告给操作员并提供故障信息以便维修和校正
  - 对程序中所造成的破坏进行估价和恢复

2.2.4 中断事件处理原则

- I/O中断事件：
  - I/O操作正常结束：设置等待I/O的进程为就绪状态，继续执行
  - I/O操作发生故障（ERROR）：索取设备状态字，分析故障原因并告警
  - I/O操作发生异常（WARN）：依据不同情况采取措施
  - 新增、断开设备：修改设备状态位

2.2.4 中断事件处理原则

- 时钟中断
  - 时钟是操作系统进行调度工作的重要工具：
    - 维护系统的绝对日期和时间
    - 分时进程按时间片轮转
    - 实时进程定时发送或接收控制信号
    - 系统定时唤醒或阻塞进程
    - 对用户进程记帐
    - 测量系统性能等
    - 利用定时器能够确保操作系统在必要时获得控制权
    - 陷入死循环的进程最终会因时间片耗尽而被迫让出处理器

千年虫 Y2K
Year 2000 Problem

- 在Intel x86 PC中，Linux利用CMOS中的时间作为系统启动时的基准时间
- Linux系统时间的测量基准是jiffies(瞬时)(一个全局变量)。系统启动时，CMOS中记录的时间转化从1970年1月1日0时0分0秒（UNIX纪元）算起的jiffies值（累积秒数）

```
$ date +%s
```

2.2.4 中断事件处理原则

- 绝对时钟（硬中断）
  - 系统设置绝对时钟寄存器，定时地把该寄存器的内容加1。如果开始时这个寄存器的内容为0，那么，只要操作员告诉系统开机时的年、月、日、时、分、秒，以后就可推算出当前的年、月、日、时、分、秒
  - 计算当前时间时，只要按时钟中断的次数和绝对时钟寄存器的内容推算就可得到
- 间隔时钟（软中断）
  - 间隔时钟是定时将一个间隔时钟寄存器的内容减1，当间隔时钟寄存器的内容为0时，产生一个间隔时钟中断，起到闹钟的作用，意味着预定的时间到了。操作系统经常利用间隔时钟作控制调度

2.2.4 中断事件处理原则

- Linux 设置的三类不同的间隔定时器
  - real：按实际经过时间计时，不论进程状态
    - SIGALARM：默认为终止当前进程
  - virtual：在用户态下为当前进程计时
    - SIGVTALRM：默认为终止当前进程
  - profile：在用户态和内核态下为当前进程计时
    - SIGPROF：默认为终止当前进程

应用程序可以捕获这些信号，并自定义信息处理程序

2.2.4 中断事件处理原则

- 程序性中断事件
  - 语法错误，可由编译程序发现并报错
  - 逻辑错误，可由测试程序发现并报错
  - 程序运行过程中所产生的异常
    - 定点溢出
    - 阶码下溢
    - 除数为零

借助信号处理机制，由操作系统捕获这类中断事件，再交给应用程序自行处理

2.2.4 中断事件处理原则

- 访管中断
  - 表示当前运行程序对操作系统功能的调用
  - 可看作是机器指令的一种扩充
  - 包括操作码和访管参数两部分
    - 操作码表示此指令为访管指令
    - 访管参数表示具体的访管要求

2.2.4 中断事件处理原则

- 访管中断
  - 表示当前运行程序对操作系统功能的调用
  - 可看作是机器指令的一种扩充
  - 包括操作码和访管参数两部分
    - 操作码表
    - 访管参数

1. 程序执行访管指令，并通过适当方式指名系统调用号
2. 通过中断机制进入访管中断处理程序，现场保护到核心栈
3. 通过系统调用入口表找到相应功能服务程序的入口地址
4. 执行相应的服务程序，正常情况下载结束后返回系统调用的下一条指令继续执行

2.2.5 中断优先级和多重中断

- 中断是随机发生的，在计算机的每一瞬间，都可能有多个中断事件发生，需要设置中断优先级：按中断请求的轻重缓急的程度预定的顺序
  - 紧迫程度相当的归为同一级别
  - 紧迫程度差别大的归为不同级别
- 优先顺序原则：根据某个中断源或中断级若得不到及时响应，造成计算机出错的严重性程度来定

2.2.5 中断优先级和多重中断

- 中断优先级的实现由软硬件结合：
  - 硬件：根据排定的优先次序做一个硬件链式排定器，当有高一级的中断事件产生时，封住比它优先级低的所有中断源
  - 软件：编写一个查询程序，依据优先级次序自高到低进行查询
- IBM系列机中，中断优先级响应顺序从高到低是：
  - 机器校验中断
  - 自愿性中断
  - 程序性中断
  - 外部中断
  - I/O中断

2.2.5 中断优先级和多重中断

- 中断的屏蔽
  - 中断屏蔽是指产生并提出中断请求后，CPU允许响应或禁止响应的状态位
  - 计算机均配置可编程中断控制器，通过指令设置可编程中断控制器的屏蔽码
  - 中断的屏蔽是根据程序状态字中的中断屏蔽位来实现的。通常允许中断时屏蔽位为1，禁止中断时，屏蔽位为0

2.2.5 中断优先级和多重中断

- 中断寄存器：用于记录中断事件
  - 中断字：中断寄存器的内容，每一位对应一个中断事件。
  - 每当一条机器指令执行结束的时刻，中断控制部件扫描中断字，查看是否有中断事件发生，若有则处理器便响应这个中断请求
- 由于同一时刻可能有多个中断事件发生，中断装置将根据中断屏蔽要求和中断优先级选取一个，然后把中断寄存器的内容送入程序状态字寄存器的中断码字段，且把中断寄存器相应位清“0”

2.2.5 中断优先级和多重中断

- 多重中断：中断同时出现、中断虽不同时出现却被硬件同时发现、其他中断正在处理期间，CPU又响应了新的中断事件，于是暂停正在运行的中断处理程序，转去执行新的中断处理程序，这就是多重中断，又称中断嵌套
- 原则：优先级高的允许打短优先级低的，但优先级低的不允许打短优先级高的

2.2.5 中断优先级和多重中断

- 旧程序状态字：被中断的程序的程序状态字
- 新程序状态字：中断处理程序的程序状态字
- 实现新旧程序状态字的交换
  - 通常，系统为每一种中断都开辟了主存的固定单元存放新的和旧的程序状态字

- 从而能够实现嵌套的中断调用
- 当前中断，会屏蔽掉所有比它优先级低的中断
- 因此，不会出现中断处理程序被自身中断的情况

2.2.5 中断优先级和多重中断

- 串行处理：同一优先级的不同中断，通常由同一个中断处理程序按自左至右的顺序逐个处理并清除之
- 即时处理：如果在处理中断程序时出现程序性中断，表示当前程序有异常，需要对其立即响应并处理