设备管理有关的数据结构

- 描述设备、控制器等部件的表格：系统中常常为每一个部件、每一台设备分别设置一张表格，常称为设备表或部件控制块。这类表格具体描述设备的类型、标识符、状态，以及当前使用者的进程标识符等
- 建立同类资源的队列：系统为了方便对I/O设备的分配管理，通常在设备表的基础上通过指针将相同物理属性的设备连成队列（称设备队列）
- 面向进程I/O请求的动态数据结构：每当进程发出I/O请求时，系统建立一张表格（称I/O请求包），将此次I/O请求的参数填入表中，同时也将该I/O有关的系统缓冲区地址等信息填入表中。I/O请求包随着I/O的完成而被删除
- 建立I/O队列：如请求包队列

独占设备的分配

- 在申请设备时，如果设备空闲，就将其独占，不再允许其他进程申请使用，一直等到该设备被释放，才允许被其他进程申请使用

  考虑效率问题，并避免由于不合理的分配策略造成死锁

静态分配：

在进程运行前，完成设备分配；运行结束时，收回设备

缺点：设备利用率低

动态分配：

在进程运行过程中，当用户提出设备要求时，进行分配，一旦停止使用立即收回

优点：效率好；缺点：分配策略不好时，产生死锁

分时式共享设备的分配

- 所谓分时式共享就是以一次I/O为单位分时使用设备，不同进程的I/O操作请求以排队方式分时地占用设备进行I/O
- 由于同时有多个进程同时访问，且访问频繁，就会影响整个设备使用效率，影响系统效率。因此要考虑多个访问请求到达时服务的顺序，使平均服务时间越短越好

SPOOLing方式使用外设

设备驱动程序(1/3)

- 与设备密切相关的代码放在设备驱动程序中，每个设备驱动程序处理一种设备类型
- 一般，设备驱动程序的任务是接收来自与设备无关的上层软件的抽象请求，并执行这个请求
- 每一个控制器都设有一个或多个设备寄存器，用来放向设备发送的命令和参数。设备驱动程序负责释这些命令，并监督它们正确执行
- 在设备驱动程序的进程释放一条或多条命令后，系有两种处理方式，多数情况下，执行设备驱动程序的进程必须等待命令完成，这样，在命令开始执行后，它阻塞自己，直到中断处理时将它解除阻塞为止；而在其它情况下，命令执行不必延迟就很快完成

设备驱动程序(2/3)

- 设备驱动程序与外界的接口
  - 与操作系统的接口

为实现设备无关性，设备作为特殊文件处理。用户的I/O请求、对命令的合法性检查以及参数处理在文件系统中完成。在需要各种设备执行具体操作时，通过相应数据结构转入不同的设备驱动程序

  - 与系统引导的接口（初始化，包括分配数据结构，建立设备的请求队列）
  - 与设备的接口

因此在设备驱动程序当中会有涉及到了这样一些函数，

设备驱动程序接口函数(3/3)

- 驱动程序初始化函数（如向操作系统登记该驱动程序的接口函数，该初始化函数在系统启动时或驱动程序安装入内核时执行）
- 驱动程序卸载函数
- 申请设备函数
- 释放设备函数
- I/O操作函数

对独占设备，包含启动I/O的指令；对共享设备，将I/O请求形成一个请求包，排到设备请求队列，如果请求队列空，则直接启动设备

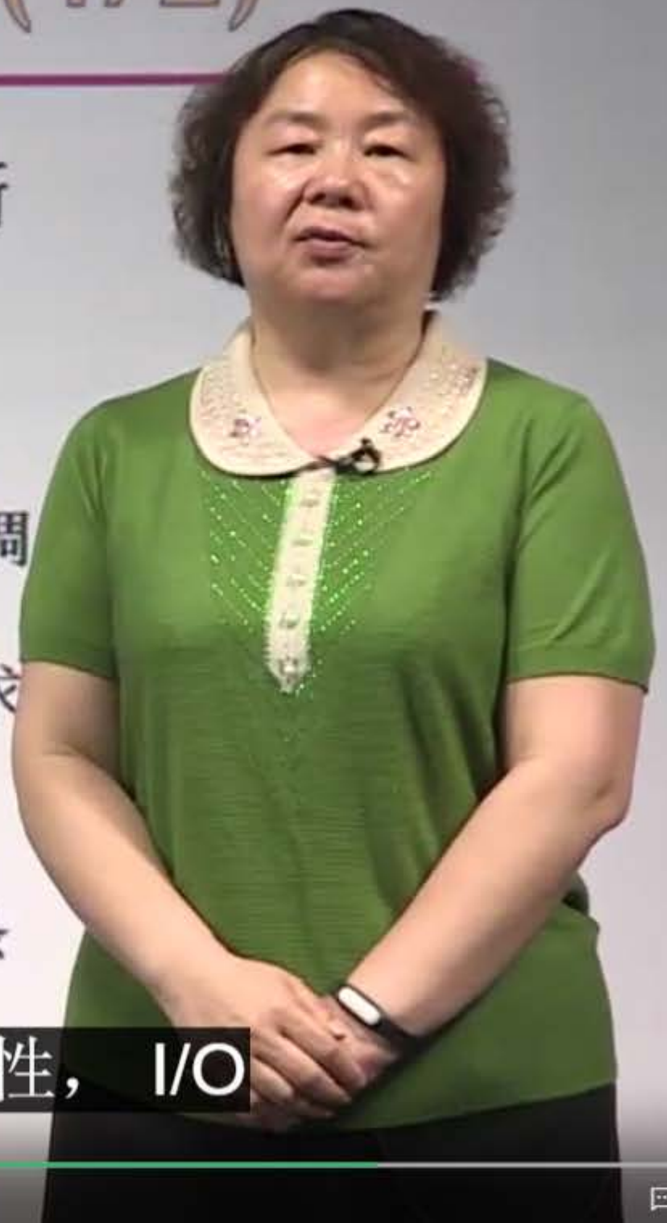
- 中断处理函数

对I/O完成做善后处理，一般是唤醒等待刚完成I/O请求的阻塞进程，使其能进一步做后续工作；如果存在I/O请求队列，则启动下一个I/O请求

一种典型的实现方案：I/O进程(1/2)

- I/O进程：专门处理系统中的I/O请求和I/O中断工作

- I/O请求的进入
  - 用户程序：调用send将I/O请求发送给I/O进程；调用block将自己阻塞，直到I/O任务完成后被唤醒
  - 系统：利用wakeup唤醒I/O进程，完成用户所要求的I/O处理
- I/O中断的进入
  - 当I/O中断发生时，内核中的中断处理程序发一条消息给I/O进程，由I/O进程负责判断并处理中断



下面我们讨论与下 I/O 进程的一些特性， I/O

一种典型的实现方案：I/O进程(2/2)

- I/O进程
  - 是系统进程，一般赋予最高优先级。一旦被唤醒，它可以很快抢占处理机投入运行
  - I/O进程开始运行后，首先关闭中断，然后用receive去接收消息

两种情形：

- 没有消息，则开中断，将自己阻塞
- 有消息，则判断消息类型（I/O请求或I/O中断）

a. I/O请求

准备通道程序，发出启动I/O指令，继续判断有无消息

b. I/O中断，进一步判断正常或异常结束

正常：唤醒要求进行I/O操作的进程

异常：转入相应的错误处理程序

0:00 下面我们来介绍一下 I/O 设备的管理， I/O 设备管理呢涉及到了一些相应的数据结构 这里头呢包括了描述、 设备和控制器的表格 以及建立相同类资源的一个队列。 还有呢就是面向进程 I/O 请求的动态的数据结构以及建立一个I/O队列，I/O请求包的队列。 那我们探讨一下独占设备的分配过程。 在申请设备的时候，如果设备空闲，那么就将其独占， 不允许其他进程再使用，直到这个设备被释放， 那么通常为了考虑效率的问题， 并且呢避免由于不合理的分配策略来造成死锁。 那么这个时候呢，我们要考虑到是静态分配策略还是动态分配策略。 所谓静态分配策略就是在进程运行之前完成设备分配， 然后运行过程中就不会有设备的请求，运行结束之后收回设备， 它的缺点是设备利用率低，但是动态分配策略呢也是在 进程运行过程中，当用户提出来设备请求的时候，那么进行分配 一旦停止了使用就马上收回。 当然呢，它的优点是效率比较高，它的缺点就会 导致当分配策略不好的时候，会产生死锁问题。 这是独占设别分配的过程中应该考虑的问题， 对于共享设备的分配呢，往往采用的是一种叫分时式的共享， 所谓分时式的共享，就是以一次I/O为单位， 分时使用不同的设备，因此不同的进程 它的I/O操作呢它会以排队的方式来分时地 占用设备进行相应的I/O，那么由于同时 允许多个进程来进行访问，而且访问非常频繁， 因此就要考虑到了整个设备的使用效率问题， 因此我们在前面讲磁盘调度的时候 实际上就是针对这样一个问题，考虑到多个访问请求 到达时的这样一个顺序，又要考虑到平均的服务时间就要进行相应的优化工作， 所以这是共享设备使用的时候要考虑 设备的使用的性能的优化， 下面我们来介绍一下驱动程序，设备驱动程序呢实际上是与设备 密切相关的代码都放在了设备驱动程序里头， 通常情况下，每个设备驱动程序呢是管一类设备， 设备驱动程序呢它处的这个层次呢是比较底层，它从上层也就是 来自设备无关的上层的软件接受到了抽象的请求并且完成这个请求。 由于每一个控制器都设有一个或多个设备寄存器， 用来存放CPU向设备发送的命令 和参数，因此呢设备驱动程序实际上就是负责 发放这些命令并且监督这些指令执行的。 那么在设备驱动程序向设备释放了一条或多条指令之后呢， 通常有两种方式，一种方式呢就是设备驱动 程序呢它对应的这个进程要等待这个命令的完成。 因此在等待过程中，它呢阻塞自己 直到完成了设备请求之后，设备发来了中断。 那么其他的方式呢就是可以 不延迟，不去等待继续来执行别的事情。 所以这个是设备驱动程序在发出了命令之后 有两种处理方式，一种是进程等待，一种是进程不等待。 那么设备驱动程序它和其他部分 的接口呢包括了三个部分，第一个部分是对操作系统的接

口， 那么在操作系统当中呢，它是从设备无关层接受请求， 因为为了实现设备无关性，那么设备是作为特殊的 文件进行处理，所以用户的 I/O 请求对命令的合法性检查以及参数 处理都是在文件系统中完成了，在需要各种设备 进行具体操作的时候呢，要通过相应的数据结构把相应的 数据传送给设备驱动程序，这就是与操作系统的接口。 那么第二个是与系统引导的接口，也就是在 操作系统引导的时候对这些设备驱动程序要有一个初始化的过程，包括了分配必要的数据结构，建立设备的请求队列等等。 另外就是与设备的接口，就是对设备的各种寄存器，推送相应的命令和数据。 因此在设备驱动程序当中会有涉及到了这样一些函数， 包括了驱动程序的初始化函数， 它的卸载函数，申请设备的函数，释放设备还有 I/O 操作的函数，以及中断处理的这个函数 下面我们介绍一种典型的 I/O 实现方案， I/O 进程，I/O 进程呢是操作系统 为了处理各种各样的 I/O 请求 和 I/O 中断来设置的一个特殊的一个系统进程。 这个系统进程呢可以接收 I/O 请求，也可以 处理相应的中断信号，当用户进程 要想提出 I/O 请求的时候呢它可以通过 send 将 I/O 请求发送给这个 I/O 进程， 并且调用 block 将自己阻塞起来。 直到 I/O 任务完成被唤醒。 而操作系统呢可以利用 wakeup 来唤醒 I/O 进程，完成用户提出的 I/O 请求。 如果有 I/O 中断信号来了之后呢 那么内核当中的中断处理程序会把这个 信号组织成一条消息发送给 I/O 进程。 由 I/O 进程来负责判断并处理这个中断信号。 下面我们讨论与下 I/O 进程的一些特性， I/O 进程呢实际上是一个系统进程， 通常呢赋予它优先级非常高，一旦它被唤醒， 就可以很快的抢占CPU，投入运行， 那么 I/O 进程开始运行后首先呢要关闭中断 然后去用 receive 来去接收有没有消息。 如果没有消息那么就把中断打开，然后将自己阻塞。 直到被唤醒。 如果有消息，那就要判断消息类型， 因为这个消息可能会来自用户进程，也可能会来自中断。 根据它是什么类型做相应的处理， 如果这次消息是一个 I/O 请求，那么就去准备相应的程序， 发出 I/O 的指令，然后继续判断有没有其他的消息， 如果这次是个 I/O 中断的消息，那么就要判断一下是正常的结束还是异常， 如果是正常结束那就去唤醒，等待这个 I/O 结果的进程。 如果是异常的话就进入到相应的错误处理程序。 所以这就是一个典型的 I/O 进程的一个工作过程。