彎曲評論

科技 · 人物 · 潮流

《孙钟秀 . 操作系统教程》注释(稿)

(第五章--设备管理)

编注: 陈怀临

Everything is a file是现代操作系统的一个重要原理。通过文件系统的接口来统一抽象和管理。

第五章

块设备（Block Device）

设备管理

字符设备（Character Device）

现代计算机系统配置大量外部设备，外部设备分为两大类：一类是存储型设备（如磁带机、磁盘机等），以存储大量信息和快速检索为目标，在系统中存储持久性信息，它作为主存储器的扩充，所以又称辅助存储器；另一类是 I/O 型设备（如显示器、卡片机、打印机、通信设备等），它们把外界信息输入计算机，把计算结果从计算机中输出，完成计算机之间的通信或人机交互。

设备管理是操作系统中最为庞杂和琐碎的部分，通常使用 I/O 中断、缓冲区管理、通道、设备驱动调度等多种技术，这些措施较好地克服了由于设备和 CPU 速度的不匹配所引起的问题，使主机和设备能够并行工作，提高设备使用效率。另一方面，操作系统将所有设备（如磁带机、打印机、显示器和终端等）都定义为文件，将其统一在文件系统之下，赋予文件属性，对设备的操作就类似于对文件的操作，其优点是：尽可能统一文件和设备的 I/O 处理；尽可能把设备文件和普通文件纳入同一保护机制下。为了方便用户或高层进程使用，设备管理还对各种设备进行抽象，配置驱动程序，提供统一界面，屏蔽设备的物理细节和操作过程。为此，设备管理应具有以下功能：设备中断处理；缓冲区管理；设备分配和去配；设备驱动调度；虚拟设备及其实现。

5.1 I/O 硬件原理

5.1.1 I/O 系统

通常把 I/O 设备及其接口线路、控制部件、通道和管理软件称为 I/O 系统，把计算机的主存储器和设备介质之间的信息传送操作称为 I/O 操作。随着计算机技术的飞速进步和应用领域的不断扩大，计算机的 I/O 信息量急剧增加，I/O 设备的种类和数量越来越多，它们与主机的联络和信息交换方式各不相同，I/O 操作不仅影响计算机的通用性和可扩充性，而且成为计算机系统综合处理能力及性能价格比的重要因素。

可按照不同方式对设备进行分类：按 I/O 操作特性，分为输入型设备、输出型设备和存储型设备；按 I/O 信息交换单位，分为字符设备和块设备。输入型设备和输出型设备通常是字符设备，它与主存储器进行信息交换的单位是字节，即一次交换 1 个或多个字节；存储型设备通常是

通常I/O设备都是Memory Mapping的I／O。意味着，对I/O设备就像对内存操作一样的读写控制寄存器和缓冲区。

对通信系统，FPGA，ASIC芯片等，从CPU的角度看，都是设备。但这些设备本身就具备计算处理能力。都是在并行工作，例如，处理网络报文。

块设备。存储型设备又可分为顺序存取存储设备和直接存取存储设备。前者严格依赖信息的物理位置进行定位和读写，如磁带。后者的重要特性是存取任何一个物理块所需的时间几乎不依赖于此信息所处的位置，如磁盘。所谓块是存储介质上连续信息所组成的一个区域，块设备每次与主存储器交换一个或多个块信息。

不同设备的物理特性存在很大的差异，主要表现在：数据传输速率差别大，数据表示方式和传输单位不尽相同，错误的性质、报错方法及应对措施都不一样。这些差异使得无论从操作系统还是用户的角度都难以获得规范、一致的 I/O 解决方案。

5.1.2 I/O 控制方式

I/O 控制在计算机处理中占据重要的地位，为了有效地实现物理 I/O 操作，必须通过软硬件技术，对 CPU 和设备的职能进行合理分工，以平衡系统性能和硬件成本之间的矛盾。按照 I/O 控制器功能的强弱以及它和 CPU 之间联系方式的不同，可以把设备的控制方式分为 4 类，它们之间的差别在于：CPU 和设备并行工作的方式和程度不同。CPU 和设备并行工作具有重要的意义，能大幅度提高计算机系统的效率和资源利用率。

1. 轮询方式

轮询方式又称程序直接控制方式，使用查询指令测试设备控制器的忙闲状态位，确定主存储器和设备是否能交换数据。假如 CPU 上所运行的程序需要从设备读入一批数据，则 CPU 程序设置交换字节数和数据读入主存储器的起始地址，然后，向设备发出查询指令，设备控制器便把状态返回给 CPU。如果 I/O 操作忙或未就绪，则重复测试过程，继续进行查询；否则，开始数据传送，CPU 从 I/O 接口（控制器数据寄存器）读取一个字，再用存储指令保存到主存储器。如果传送尚未结束，再次向设备发出查询指令，直到全部数据传输完成。轮询方式使用 3 条指令：

(1) 查询指令：查询设备是否就绪；

(2) 读写指令：当设备就绪时，执行数据交换；

(3) 转移指令：当设备未就绪时，执行转移指令转向查询指令继续查询。

有时系统中同时有多个设备要求执行 I/O 操作，那么对每个设备都编写一段 I/O 数据处理程序，然后，轮流查询这些设备的状态位，当某一个设备准备就绪时，就调用这个设备的 I/O 处理程序进行数据传输，否则依次轮询下一个设备是否准备好。

一方面，CPU 轮询设备当前的状态，会终止原程序的执行，浪费宝贵的时间；另一方面，I/O 准备就绪后，需要 CPU 参与数据传输工作。可见 CPU 和设备只能串行工作，使主机不能充分发挥功效，设备也不能得到合理利用，整个系统效率很低。

2. 中断方式

中断方式要求 CPU 与设备控制器及设备之间存在中断请求线，设备控制器的状态寄存器有相应的中断允许位，CPU 与设备之间传输数据的过程如下。

(1) 进程发出启动 I/O 指令，这时 CPU 会加载控制信息到设备控制器的寄存器，然后，进程继续执行不涉及本次 I/O 数据的任务，或放弃 CPU 等待设备 I/O 操作完成；

有一种混合方式是：通过时钟中断，
然后去定期的查询是否有I/O，例如网卡是否有报文。

例如，Packet Buffer

(2) 设备控制器检查状态寄存器的内容，按照 I/O 指令的要求，执行相应的 I/O 操作(如发现读请求时，命令设备把数据传送到 I/O 缓冲寄存器)，一旦传输完成，设备控制器通过中断请求线发出 I/O 中断信号；

(3) CPU 收到并响应 I/O 中断后，转向设备的 I/O 中断处理程序执行；

(4) 中断处理程序(有的操作系统把控制权再交给设备驱动程序，并退出中断)执行数据读取操作，将 I/O 缓冲寄存器的内容写入主存储器，操作结束后退出中断处理程序，返回发生中断前的执行状态；

(5) 进程调度程序在适当时刻对于得到数据的进程恢复执行。

在 I/O 中断方式中，如果设备控制器的数据缓冲区较小，当缓冲器装满后便会发生中断，那么在数据传输的过程中，发生中断的次数会较多，这样会耗用大量 CPU 时间。若系统配置多种设备，这些设备都通过中断处理方式实现并行工作，会使中断次数急剧增加，造成 CPU 来不及响应或丢失数据的现象。但是，程序中断方式 I/O 由于不必忙式轮询 I/O 的准备情况，CPU 和设备可实现部分并行操作，与程序查询 I/O 的串行工作方式相比，使 CPU 资源得到更充分的利用。

3. DMA 方式

虽然中断方式消除程序轮询中的忙式测试，提高了 CPU 资源的利用率，但在响应中断请求之后，必须停止现行程序，转入中断处理程序并参与数据传输操作。如果设备能直接与主存储器交换数据而不占用 CPU，那么，CPU 资源的利用率还可以提高，这就出现了直接存储器存取(Direct Memory Access，DMA)方式。在 DMA 方式中，主存储器和设备之间有一条数据通路，成块地传送数据，无须 CPU 干预，实际数据传输操作由 DMA 直接完成。为了实现直接存储器存取操作，至少需要以下一些逻辑部件。

(1) 主存地址寄存器：存放主存中需要交换数据的地址，DMA 传送之前，由程序送入首地址；DMA 传送过程中，每次交换数据都把地址寄存器的内容加 1；

(2) 字计数器：记录传送数据的总字数，每次传送一个字就把字计数器减 1；

(3) 数据缓冲寄存器或数据缓冲区：暂存每次传送的数据；

(4) 设备地址寄存器：存放 I/O 信息的地址，如磁盘的柱面号、磁道号、块号；

(5) 中断机制和控制逻辑：用于向 CPU 提出 I/O 中断请求，及保存 CPU 发来的 I/O 命令，管理 DMA 的传送过程。

DMA 与主存储器之间采用字传送，DMA 与设备之间可能是字位或字节传送，所以，DMA 中要设置数据移位寄存器、字节计数器等硬件逻辑。可能会出现这样的问题：为什么控制器从设备读取数据后不立即将其送入主存储器，而需要内部缓冲区呢？其原因是，一旦磁盘开始读数据，从磁盘读出比特流的速率是恒定的，无论控制器是否做好接收这些比特的准备，若此时控制器要将数据直接复制到主存储器中，必须在每个字传送完毕后获得对系统总线的控制权。如果其他设备也在争用总线，则有可能暂时等待，当上一个字还未送入主存储器而下一个字已经到达时，控制器只能另寻暂存之地，如果总线非常忙，则控制器可能需要对大量的信息暂存。可见，采用内部缓冲区，在 DMA 操作启动前不需要使用总线，这样控制器的设计就比较简单，因为从

从总线的角度，CPU， 内存和I／O设备其实都是挂在Bus上的器件。CPU本身也是设备。DMA方式就是I／O
设备通过Bus直接把数据发送给内存DRAM。

http://en.wikipedia.org/wiki/Channel_I/O

DMA 到主存储器的传输对时间的要求并不十分严格。

DMA 不仅设有中断机制，还增加 DMA 传输控制机制，若出现 DMA 与 CPU 同时经总线访问主存储器的情况，CPU 总是把总线占有权让给 DMA，DMA 的这种占有称为"周期窃用"。窃取时间通常为一个存取周期，让设备和主存储器之间交换数据，而不再需要 CPU 干预，这样可以减轻 CPU 的负担。每次传输数据时，不必进入中断系统，进一步提高 CPU 资源的利用率。

由于 DMA 方式的实现线路简单，价格低廉，小型计算机、微型计算机中的快速设备均采用这种方案。但是，DMA 传输需要窃用时钟周期，会降低 CPU 的处理效率；DMA 的功能不够强，不能满足复杂的 I/O 操作要求，因而在大中型计算机中一般使用通道技术。

4. 通道方式

Channel

DMA 方式与程序中断方式相比，进一步减少了 CPU 对 I/O 操作的干预，但每发出一次 I/O 指令，只能读写一个数据块，用户希望一次能够读写多个离散的数据块，并把它们传送到不同的主存区域或相反，则需要由 CPU 发出多条启动 I/O 指令及进行多次 I/O 中断处理才能完成。通道方式是 DMA 方式的发展，能够再次减少 CPU 对 I/O 操作的干预。同时，为了充分发挥 CPU 和设备之间的并行工作能力，也为了让种类繁多且物理特性各异的设备能够以标准的接口连接到系统中，计算机系统引入自成体系的通道结构，通道方式的出现是现代计算机系统功能不断完善、性能不断提高的结果，是计算机技术的重要进步。

通道又称 I/O 处理器，能完成主存储器和设备之间的信息传送，与 CPU 并行地执行操作。采用通道技术主要解决 I/O 操作的独立性和硬部件工作的并行性，由通道来管理和控制 I/O 操作，大大减少设备和 CPU 之间的逻辑联系，把 CPU 从琐碎的 I/O 操作中解放出来，实现设备和 CPU 并行操作，通道之间并行操作，设备之间并行操作，达到提高整个系统效率的目的。

具有通道装置的计算机系统的主机、通道、控制器和设备之间采用四级连接，实施三级控制。一个 CPU 通常可以连接若干通道，一个通道可以连接若干控制器，一个控制器可以连接若干台设备。CPU 通过执行 I/O 指令对通道实施控制，通道通过执行通道命令对控制器实施控制，控制器发出动作序列对设备实施控制，设备执行相应的 I/O 操作。

采用 I/O 通道设计后，I/O 操作的过程如下：CPU 在执行主程序时遇到 I/O 请求，启动在指定通道上选址的设备，一旦启动成功，通道开始控制设备进行操作，这时 CPU 就可以执行其他任务并与通道并行工作，直到 I/O 操作完成；当通道发出 I/O 操作结束中断时，处理器才响应并停止当前的工作，转而处理 I/O 操作结束事件。

5.1.3 设备控制器

I/O 设备通常由机械部件和电子部件所组成，一般是将其分开处理，以达到模块化和通用性的设计目标。电子部件称为设备控制器或适配器，在微型计算机中，它是可插入主板扩充槽的印刷电路板；机械部件是指设备本身。

控制器卡上一般都设有接线器，与设备相连的电缆线相接，许多控制器可以控制两个或更多的同类设备，它和设备之间的接口采用国际标准，这样各计算机厂商都可制造与标准接口相匹配

通道（Channel）方式之前主要用在大型机上。在中小计算机，通信设备系统不常见。channel有点类似协处理器。现代DMA Engine融合了早期Channel的工作方式，是独立的bus竞争，而非盗用。

控制寄存器的读写需要完整性。否则，设备可能出错。例如，某几个控制寄存器必须配合。所以对设备的操作有时需要串行化。

的控制器和设备，例如，IDE(Integrated Device Electronics，集成设备电子器件)设备、SCSI(Small Computer System Interface，小型计算机系统接口)设备。

操作系统是与控制器交互，而非与设备本身交互。多数微型计算机和小型计算机采用单总线模型，实现 CPU 和控制器之间的数据传输，而大中型计算机则采用多总线结构和多通道方式，以提高 CPU 与设备的并行操作程度。

控制器和设备之间的接口是一种低级接口，例如，磁盘被格式化为每个扇区 512 B，实际上从磁盘读出来的是字位串，以头标开始，随后是扇区的 4 096 比特(512×8)，最后是纠错码。其中，头标在磁盘格式化时写入，记录扇区地址，包括柱面号、磁头号、扇区号及同步信息。控制器的任务是把串行的位流装配成字节，存入控制器内部的缓冲区以形成字节块，在校验和确认无错后，这块数据将被复制到主存储器。

CRT 控制器也是串行位流设备，它从主存储器中读取欲显示字符的字节，产生用来调制 CRT 射线的信号，将结果显示在屏幕上。控制器还产生当水平方向扫描结束后的折返信号，及当整个屏幕扫描结束后的垂直方向的折返信号。CRT 正常工作之前，操作系统对它进行初始化，设置每屏行数、每行字符数。

如果没有控制器，这些复杂的操作就必须由操作系统设计人员通过编写程序来解决，而引入控制器后，只需通过传送简单的参数就可执行 I/O 操作，大大地简化系统的设计，有利于计算机系统对各类控制器和设备的兼容性。

综述设备控制器的功能和结构，设备控制器是 CPU 和设备之间的接口，它接收从 CPU 发来的命令，控制设备的操作，实现主存储器和设备之间的数据传输，从而使 CPU 从繁杂的设备操作中解放出来。设备控制器是可编址设备，当它连接多台设备时，应具有多个设备地址。设备控制器的主要功能是：

(1) 接收和识别 CPU 或通道所发来的命令，例如，磁盘控制器能接收读、写、查找等各种命令；

(2) 实现数据交换，包括设备和控制器之间的数据传输，且通过数据总线或通道，控制器和主存储器之间传输数据；

(3) 发现和记录设备及自身的状态信息，供 CPU 处理使用；

(4) 设备地址识别。

为了实现上面列举的各项功能，设备控制器必须有以下组成部分：控制寄存器及译码器，数据缓冲寄存器，状态寄存器，地址译码器，及用于对设备操作进行控制的 I/O 逻辑。

5.2 I/O 软件原理

控制器（Controller）通常占据一个高端内存地址区域。对控制器的操作，就类似于读写内存地址。PCI和相关的总线标准把一个I／O设备与内存地址空间映射统一起来。

[注释]：“Everything is a file”是现代操作系统的一个重要概念。通过文件系统的抽象和接口来屏蔽设备多线性。理解设备时，要站在总线角度看：CPU和其他设备，例如ASIC芯片，都是设备。都具备计算处理能力。需要和内存打交道，大家共享总线。

现在有的设备控制器可以有选项，攒多少个事件之后，才向CPU发一个中断申请。

盘 I/O 操作的效率。通用性意味着用统一标准的方法来管理所有设备。为了达到这些目标，把软件组织成层次结构，低层软件用来屏蔽硬件细节，高层软件向用户提供简洁、友善的界面。I/O 软件设计需要考虑的问题有：

(1) 设备无关性：程序员编写访问文件数据的程序时，与具体的物理设备无关；

(2) 出错处理：数据传输过程中产生的错误应该在尽可能靠近硬件的地方处理，低层软件能够解决的错误不让高层软件感知；

(3) 同步/异步传输：CPU 在启动 I/O 操作后既可继续执行其他工作，直至中断到达，称此为异步传输；又可采用阻塞方式，让启动 I/O 操作的进程挂起等待，直至数据传输完成，称此为同步传输，I/O 软件应支持这两种工作方式；

(4) 缓冲技术：建立数据缓冲区，让数据的到达率与离去率相匹配，以提高系统吞吐率。

为了合理、高效地解决以上问题，操作系统把 I/O 软件依次组织成 4 个层次：I/O 中断处理程序；I/O 设备驱动程序；独立于设备的 I/O 软件和用户空间的 I/O 软件。

user land device driver

5.2.2 I/O 中断处理程序

通常中断处理程序是设备驱动的组成部分之一，位于操作系统底层，是与硬件设备密切相关的软件，它与系统的其余部分尽可能少地发生联系。当进程请求 I/O 操作时，通常将被挂起，直到数据传输结束并产生 I/O 中断时，操作系统接管 CPU 后转向中断处理程序执行。

当设备向 CPU 提出中断请求时，CPU 响应请求并转入中断处理程序执行，通常需要做到：检查设备状态寄存器的内容，判断产生中断的原因，根据 I/O 操作的完成情况进行相应的处理；若数据传输有错，应向上层软件报告设备出错信息，实施重执行；若正常结束，应唤醒等待传输的进程，使其转换为就绪态；若有等待传输的 I/O 命令，应通知相关软件启动下一个 I/O 请求。

5.2.3 I/O 设备驱动程序

设备驱动程序包括与设备密切相关的所有代码，其任务是：把用户提交的逻辑 I/O 请求转化为物理 I/O 操作的启动和执行，如设备名转化为端口地址、逻辑记录转化为物理记录、逻辑操作转化为物理操作等。同时，监督设备是否正确执行，管理数据缓冲区，进行必要的纠错处理。

笼统地说，设备驱动程序的功能是从独立于设备的软件中接收并执行 I/O 请求。一条典型的 I/O 请求是读第 n 块，如果请求到来时驱动程序空闲，它立即执行这个请求；但如果它正在处理请求，会将新到来的请求放入 I/O 请求队列中，待当前请求完成后，再依次从 I/O 请求队列中取出 I/O 请求进行处理。

执行 I/O 请求的第一步是将其转换为更具体的形式。对于磁盘驱动程序而言，它包含：计算所请求块的物理地址、检查驱动器是否运转、检测磁头臂是否定位在正确的柱面位置等，简而言之，它必须确定需要哪些控制器命令及其执行次序，然后就向控制器的设备寄存器写入这些命令和相应的参数。某些控制器一次只能接收一条命令(如 DMA)，另一些控制器可接收一串命令并自动进行处理(如通道方式)。

现代OS都支持异步I/O。进程不需要阻塞等待。

设备驱动程序发出控制命令之后，系统有两种处理方式。在大多数情况下，执行设备驱动程序的进程需要等待命令完成，在命令开始执行后，进程阻塞自己，直到中断处理时才解除阻塞。另一种情况是执行驱动程序的进程无须阻塞，例如，有些终端的滚动屏幕只需向控制器的设备寄存器中写入几个字节，整个操作可在几微秒内完成。无论是哪种情况，在操作完成后都要检查数据传输是否有错，如果有错，返回错误状态提示信息给调用者；如果正常，驱动程序将数据传送给与设备无关的软件层；如果 I/O 请求队列中有请求在排队，则选中其中一个请求执行，否则等待下一个 I/O 请求的到来。

设备驱动程序主要包含三部分功能：

1. 设备初始化

在系统初次启动或设备传输数据时，预置设备和控制器以及通道的状态。

2. 执行设备驱动例程

负责启动设备，进行数据传输，对于具有通道的 I/O 系统，此例程还负责形成通道指令，启动通道工作。

3. 执行中断处理例程

负责处理设备和控制器及通道所发出的各种中断。

5.2.4 独立于设备的 I/O 软件

虽然设备驱动程序是设备专用的，但大部分 I/O 软件却与设备无关，独立于设备的 I/O 软件和设备驱动程序之间的界限取决于具体的系统，其基本功能是执行适用于所有设备的常用 I/O 功能，并向用户层软件提供一致性接口。

1. 设备命名和设备保护

对于操作系统而言，设备都被看做文件，它与磁盘文件一样，通过路径名来寻址，每个设备具有文件名、inode、文件所有者、权限位等属性。例如，在 UNIX/Linux 中，设备名/dev/tty00 唯一地确定为一个特别文件设置的 inode 节点，其中包含主设备号，通过它来分配正确的终端设备驱动程序；次设备号作为参数，确定驱动程序要读写的是哪台终端设备。设备不仅具有文件名，且支持与文件相关的所有系统调用，如 open、close、read、write、stat 及 lseek 等。

设备保护需要检查用户是否有权访问所申请的设备，在多数大中型计算机系统中，用户进程对 I/O 设备的直接访问是绝对禁止的，I/O 指令定义为特权指令，通过系统调用的方式间接地供用户使用。此外，在 UNIX/Linux 中还使用一种灵活的方法，对应于 I/O 设备的特别文件采用 rwx 保护机制，以判断文件的所有者和组成员是否具有向设备发送和读取数据的权限。Windows 中的设备作为命名对象出现在文件系统中，对文件的保护规则也适用于 I/O 设备。

设备文件依赖于 inode 来实现，文件目录并不能区分文件名是代表一个磁盘文件还是设备文件，但 inode 的内容是不同的，磁盘文件的 inode 包含指向数据块的指针，而设备文件的 inode 则包含指向内核设备驱动程序的指针，用来控制设备的 I/O 操作。

换言之，写设备驱动程序就是要实现该设备相应的open，close，read，write等函数。

有时还分为两个缓冲区域，例如，描述符缓冲和数据（Payload）缓冲。

这是为什么user land driver很流行的原因。

2. 提供与设备无关的块尺寸

当创建文件并向其输入数据时，此文件必须被分配新的磁盘块。为了完成这种分配工作，操作系统需要为每个磁盘都配置一张记录空闲块的表或位示图，分配空闲块的算法是独立于设备的，可在高于设备驱动程序的层次处理。

不同磁盘的扇区大小可能不同，独立于设备的 I/O 软件屏蔽这一事实并向高层软件提供统一的数据块尺寸，它可将若干扇区合并为一个逻辑块，称为簇。这样，高层软件就只能和大小相同的簇交互，而不必关心物理扇区的尺寸。类似地，字符设备每次对一个字符进行操作，有些字符设备操作更大单位的数据（如 80 个字符），这类差别也必须在本层进行屏蔽。

3. 缓冲技术

数据离开设备之后，通常不能直接送达目的地，如网络上传送的数据包需要经过分析、检查才能知道置于何处。某些设备有严格的时间约束，如数字音频设备，所以，必须通过缓冲区来消除填满速率和清空速率的影响。块设备和字符设备都需要缓冲技术，可通过在主存储器建立缓冲区的方法来解决。对于块设备，硬件每次读写均以块为单位，而应用程序可按照任意大小单元处理数据，如果应用程序读写的长度和位置不恰好是完整的扇区时，也必须通过缓冲区来实现，先将数据读或写入缓冲区，再从缓冲区将数据传送给用户，或当缓冲区满时再写至磁盘。对于字符设备，当用户把数据写入系统的速度快于系统输出数据的速度，或字符设备提供数据的速度快或慢于应用程序消耗数据的速度时，也需要建立缓冲区。

通常在内核空间开辟缓冲区，数据在缓冲区中缓冲后，再在用户缓冲区和设备之间传输。缓冲涉及大量复制操作，故对 I/O 性能会产生很大影响。

4. 设备分配和状态跟踪

根据设备的物理特性，操作系统制订分配策略，可采用静态分配、动态分配和虚拟分配。一些独占型设备，如 CD-ROM，同一时刻只能被一个进程使用。当用户发出设备使用的请求时，要求操作系统检查相应设备的使用状态，并根据其忙闲状况来决定接受或拒绝这一请求。一种简单的处理方法是，直接用 OPEN 打开相应设备的特别文件来进行申请，若设备不可用，则 OPEN 失败；当进程用完独占型设备时，应该关闭并释放之。

5. 错误处理和报告

I/O 设备出错是经常发生的，错误应尽可能在接近硬件的地方加以处理，如果控制器发现错误，应该设法纠正和加以解决。如果未能处理错误，再交给设备驱动程序。多数错误是与设备紧密相关的，驱动程序知道应如何对其处理（如重试、忽略或报警）。一种典型的错误是磁盘块受损导致不能读写，驱动程序在尝试若干次读写操作不成功后才会放弃。如果发现错误处理不成功，或不能处理错误时，可向独立于设备的 I/O 软件报错，此后这个错误的处理就与设备驱动程序无关。低层软件所不能处理的情况交给高层软件处理。在许多情况下，错误恢复可由低层软件透明地得到解决，而高层软件甚至不知道错误的存在。如果发现 I/O 编程出错，如无效操作码、无效设备或无效缓冲区地址，则返回一个错误码并向调用者报错；如果是在读关键系统数据结构（如磁盘位示图）时出错，则只能向操作员报告，打印出错提示信息并终止运行。诸如此类与设备

例如，网卡驱动通常有一个Packet Buffer指针，指向DRAM中的一个区域。

I/O 地址传送给控制器，向其发出读、写或控制命令，控制设备进行数据传输。控制器接收通道发来的命令后，检查设备的状态，若设备不忙，则告知通道释放 CPU，并开始 I/O 操作，向设备发出动作序列，设备执行相应的动作；之后，通道独立执行通道程序中的各条通道命令字，直到通道程序执行结束。本次 I/O 操作结束之后，通道向 CPU 发出 I/O 操作结束中断，再次请求 CPU 的干预。

3. I/O 结束阶段

通道发现 I/O 系统中出现通道结束、控制器结束、设备结束或其他能产生中断的信号时，就向 CPU 申请 I/O 中断。同时，把产生中断的通道号和设备号以及通道状态字存入主存固定单元，CPU 响应 I/O 中断后，暂停现行程序的执行，转向 I/O 中断处理程序。

5.4 缓冲技术

为了解决 CPU 与设备之间速度不匹配的矛盾，及协调逻辑记录大小与物理记录大小不一致的问题，提高 CPU 和设备的并行性，减少 I/O 操作对 CPU 的中断次数，放宽对 CPU 中断响应时间的要求，操作系统普遍采用缓冲技术。缓冲用于平滑两种不同速度的硬部件或设备之间的信息传输，在主存储器中开辟一个存储区，称为缓冲区，专门用于临时存放 I/O 操作的数据。

实现缓冲技术的基本思想如下：当进程执行写操作输出数据时，先向系统申请一个输出缓冲区，然后，将数据送至缓冲区，若是顺序写请求，则不断地把数据填入缓冲区，直到装满为止，此后，进程可以继续计算，同时，系统将缓冲区的内容写到设备上。当进程执行读操作输入数据时，先向系统申请一个输入缓冲区，系统将设备上的一条物理记录读至缓冲区，然后根据要求，把当前所需要的逻辑记录从缓冲区中选出并传送给进程。

用于上述目的的专用主存区称为 I/O 缓冲区。在输出数据时，只有在系统来不及腾空缓冲区而进程又要写数据时，它才需要等待；在输入数据时，仅当缓冲区为空而进程又要从中读取数据时，它才被迫等待。其他时间可进一步提高 CPU 和设备的并行性，设备和设备之间的并行性，从而提高系统效率。在操作系统的管理下，常辟出缓冲区以服务于各种设备，常用的缓冲技术有单缓冲、双缓冲和多缓冲。

5.4.1 单缓冲

单缓冲是最简单的缓冲技术，每当应用进程发出 I/O 请求时，操作系统在主存储器的系统区中开设一个缓冲区。

对于块设备输入，单缓冲机制的工作过程是：先从磁盘把一块数据读至缓冲区，假设花费时间 T；接着，系统把缓冲区中的数据送到用户区，设所消耗的时间为 M，由于此时缓冲区已空，系统可预读紧接着的下一块，大多数应用将使用邻接块，然后应用进程对这批数据进行计算，共耗时 C。若不采用缓冲技术，数据直接从磁盘传送到用户区，每批数据处理时间约为 $T+C$，而采用单缓冲，每批数据处理时间为 $\max(C,T)+M$，通常 M 远远小于 C 或 T，故速度会快很多。

T
M
C

T和C有并行的部分了。所以效率提高了。

对于块设备输出，单缓冲机制的工作方式类似，先把数据从用户区复制到系统缓冲区，应用进程可继续请求输出，直到缓冲区填满，由系统写到磁盘上。对于字符设备输入，缓冲区用于暂存用户输入的一行数据，在输入时，应用进程挂起，等待一行数据输入完毕；在输出时，应用进程将第一行数据送入缓冲区后继续执行，如果在第一个输出操作未腾空缓冲区之前，又有第二行数据要输出，应用进程等待。如果希望实现 I/O 的并行工作，如把输入设备中的数据输入并加工，再从输出设备上输出，必须引入双缓冲技术。

5.4.2 双缓冲

两个缓冲并行的工作

为了加快 I/O 操作的执行速度，实现 I/O 的并行工作和提高设备利用率，需要引入双缓冲。在输入数据时，首先从设备读出数据填充缓冲区 1，系统从缓冲区 1 把数据传送到用户区，应用进程便可对数据进行加工和计算；与此同时，从设备读出数据填充缓冲区 2。当缓冲区 1 为空时，再次从设备读出数据到缓冲区 1，系统又可以把缓冲区 2 的数据传送到用户区，应用进程开始加工缓冲区 2 中的数据。两个缓冲区交替使用，使 CPU 和设备、设备和设备之间的并行性进一步提高，仅当两个缓冲区都为空且进程还要提取数据时，它才被迫等待。粗略地估计传输和处理一块的时间，如果 $C<T$，输入操作比计算操作速度慢，这时由于 M 远远小于 T，故将磁盘上的一块数据传送到一个缓冲区期间（所花费时间为 T），计算机已完成将另一个缓冲区中的数据传送到用户区并对这块数据进行计算的工作，所以，一块数据的传输和处理时间为 T，即 $\max(C,T)$，显然，这种情况可保证块设备连续工作；如果 $C>T$，计算操作比输入操作速度慢，每当上一块数据计算完毕后，需要把一个缓冲区中的数据传送到用户区，所花费时间为 M，再对这块数据进行计算，所花费时间为 C，所以，一块数据的传输和处理时间为 $C+M$，即 $\max(C,T)+M$，显然，这种情况使得进程不必等待 I/O 操作。双缓冲技术使系统效率提高，但复杂性也随之增加了。

采用双缓冲读卡及打印可这样进行：第一张卡片读入缓冲区 1，在打印缓冲区 1 中数据的同时，又把第二张卡片读入缓冲区 2。缓冲区 1 打印完毕时，缓冲区 2 也刚好输入完毕，让读卡机和打印机交换缓冲。这样，I/O 设备就能够处于并行工作的状态。

5.4.3 多缓冲

采用双缓冲技术虽然能提高设备的并行工作程度，但在设备和处理进程速度不匹配的情况下仍不十分理想。举例来说，若输入设备的速度快于进程消耗数据的速度，则输入设备很快就会把两个缓冲区填满；反之，若进程处理数据的速度快于数据输入速度，很快会把两个缓冲区抽空，造成进程经常处于等待态。为了改善这种情况，获得较高的并行度，常常采用多缓冲所组成的循环缓冲技术。

操作系统从主存区域中分配一组缓冲区，每个缓冲区都有一个链接指针指向下一个缓冲区，最后一个缓冲区指针指向第一个缓冲区，组成循环缓冲，缓冲区的大小等于物理记录的大小，多缓冲的缓冲区是系统的公共资源，可供进程共享，并由系统统一分配和管理。缓冲区用途可分为输入缓冲区、处理缓冲区和输出缓冲区。为了管理各类缓冲区，执行各种操作，必须设计专门的

例如，网卡驱动缓冲区可以有报文描述符缓冲区和报文数据缓冲区。

［注释］：设备通常是Memory Based I/O。控制Reg和缓冲区被映射到CPU地址空间，例如通过PCI。读写控制寄存器和缓冲区通常与DRAM内存一样。通常设备地址map到高端地址区域。通信系统里，还要事先设置一些cache属性等，例如，non-cacheable。总之，通过系统总线控制器来调控。

http://en.wikipedia.org/wiki/RAID

The most common types today are RAID 0 (striping), RAID 1 and variants (mirroring), RAID 5 (distributed parity) and RAID 6 (dual parity).

RAID技术被广泛的用在数据容错方面。大多数现代操作系统都已经支持。另外，RAID也可以是软件实现方案。许多文件系统可以支持Software – RAID。

5.5.5 独立磁盘冗余阵列

独立磁盘冗余阵列(Redundant Array of Independent Disks,RAID)的概念在1987年由美国加利福尼亚大学提出,已得到业界的认可,被广泛应用于大中型计算机和计算机网络系统中,其目的是:增加容错性,获得高性能。通过在不同的磁盘上维护冗余数据来增加容错性;通过把数据分布到多个磁盘,采用并行存取以加快数据传输速率来获得高性能。其策略是:用一组容量较小的、独立的、可并行工作的磁盘组成阵列来代替单一的大容量磁盘,再加入冗余技术,数据能以多种方式组织和分布存储,于是,独立的I/O请求能够被并行处理,分布式的单个I/O请求也能并行地从多个磁盘同时得到处理,从而增加容错性,改进I/O性能。

在基本的组织方式中,有多种类型的RAID,其主要差别在于冗余信息数量(增加磁盘的数量)和容错性级别(可纠正错位的数目),及冗余信息是集中在一个磁盘上还是分散在多个磁盘上。具体类型列举如下。

(1) RAID0:连续的数据条带(每个条带可规定为1个或多个扇区)以轮转方式写到全部的磁盘上,然后,采用并行交叉方式存取,缩短I/O请求的排队时间,适用于大数据量的I/O请求,

RAID 0方案是数据块分拆在各个磁盘上。

混合（Nested）RAID，例如RAID10的意思是RAID1+RAID 0两种方案混合，不是Ten的意思。RAID10:先做Mirror镜像，然后再做一次Data Striping分布

但并无冗余校验功能，可靠性较差。

镜像

(2) RAID1：采用镜像盘备份所有数据来提高容错性，读请求拥有最短寻道时间，写请求可并行完成。其缺点是容量下降一半，故成本很高。

(3) RAID2：采用数据字或字节方式交叉存放，并行存取以获得高性能，使用海明校验码，适合大量顺序数据的访问。由于使用多个冗余盘，其成本较高。基本上不用。

(4) RAID3：是 RAID2 的简化版本，其差别是它仅使用一只冗余盘，采用奇偶校验技术。

(5) RAID4：采用独立存取磁盘阵列，数据条带交叉存放，访问请求可并行地获得满足，适合要求较高 I/O 请求速度的应用场合，使用一只冗余盘存放奇偶校验码。

(6) RAID5：与 RAID4 的组织方式类似，但奇偶校验码循环分布在每个盘上，使得容错性更好。

多常用。

(7) RAID6：采用双重冗余技术，即使有两个数据磁盘发生错误，也可以重新生成数据。

5.5.6 提高磁盘 I/O 速度的方法

通常为磁盘设置数据缓冲区高速缓存，这样能够显著缩短等待磁盘 I/O 的时间。下面所介绍的一些方法也能有效地提高磁盘 I/O 速度。

1. 提前读　　可以理解为是利用了空间局部性原理。

对于采用顺序方式所访问的文件数据，在读当前盘块时已知下次要读出的盘块地址，因此，可在读当前盘块的同时，提前把下一盘块的数据也读入磁盘缓冲区。这样一来，当下次要读盘块中的那些数据时，由于已经提前把它们读入缓冲区，便可直接使用，从而缩短读数据的时间，相当于提高磁盘 I/O 速度。"提前读"功能已被许多操作系统如 UNIX、OS/2、Windows 所采用。

2. 延迟写　　懒惰算法。Lazy Algorithm

在执行写操作时，磁盘缓冲区中的数据本应立即写回磁盘，但考虑到此缓冲区中的数据不久之后会再次被进程访问，因此，并不立即将缓冲区中的数据写盘，而是把它挂在空闲缓冲区队列的末尾。随着空闲缓冲区的使用，存有输出数据的缓冲区不停地向队列头移动，直至移动到空闲缓冲区队列之首，当再有进程申请缓冲区且分配到此缓冲区时，才把其中的数据写到磁盘上，于是这个缓冲区可作为空闲缓冲区分配。只要存有输出数据的缓冲区还在队列中，任何对此数据的访问均可直接从中找回，不必再去访问磁盘，这样做可以减少磁盘 I/O 的次数。同样，在 UNIX、OS/2 和 Windows 系统中也都采用这一技术。

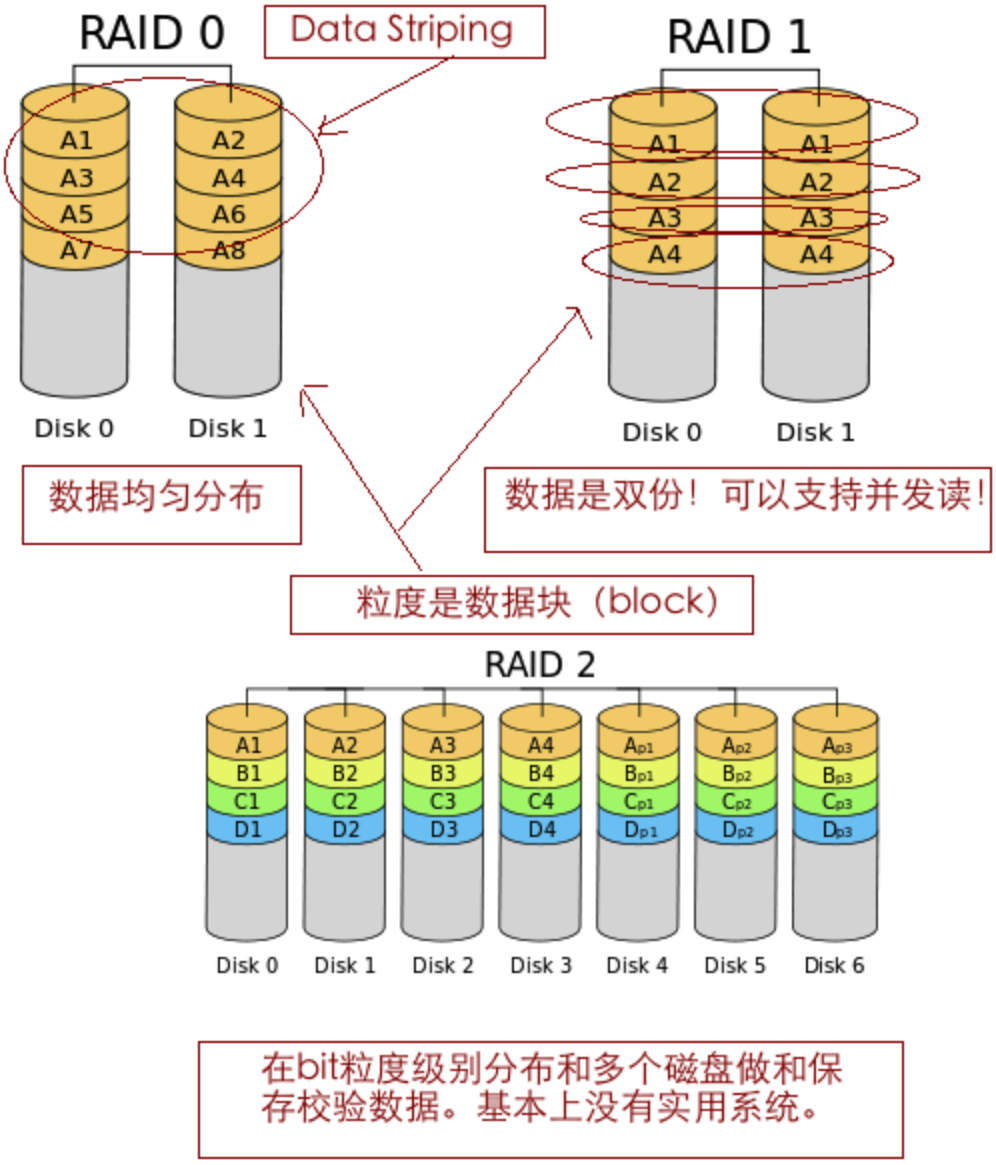
UNIX/Linux 提供两种读盘方式和三种写盘方式。"正常读"是指把磁盘块信息读入主存缓冲区；"提前读"是指在读磁盘当前块时，把下一磁盘块也读入主存缓冲区；"正常写"是指把主存缓冲区中的信息写至磁盘块，且写进程应等待写操作完成；"异步写"是指写进程无须等待写盘结束就可通回工作；"延迟写"是指仅在缓冲区首部设置"延迟写"标志，然后，释放此缓冲区并将其链入空闲缓冲区链表的尾部，当其他进程申请到此缓冲区时，才真正把缓冲区信息写回磁盘块。

3. 虚拟盘

虚拟盘是指用主存空间去仿真磁盘，又叫 RAM 盘。虚拟盘的设备驱动程序可接收所有标

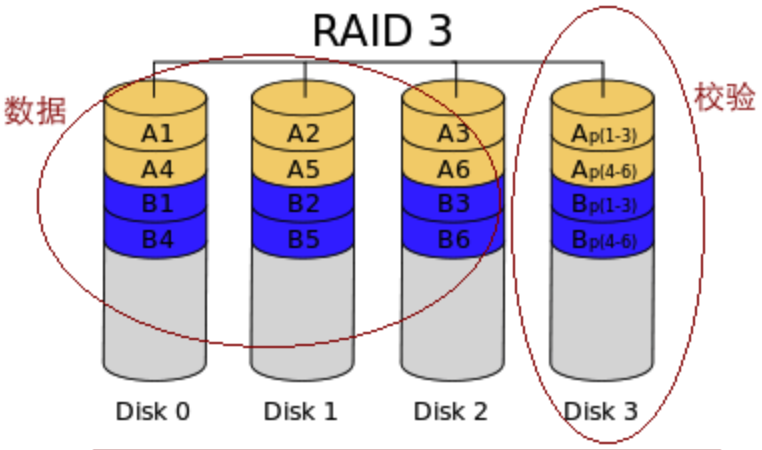
类似CPU Cache的Write Back。不要立刻写回。

RAID图解（1）

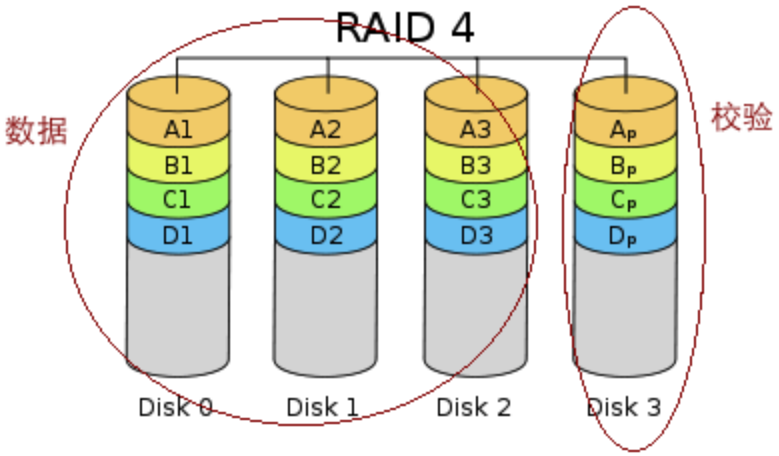


在bit粒度级别分布和多个磁盘做和保存校验数据。基本上没有实用系统。

RAID图解（2）

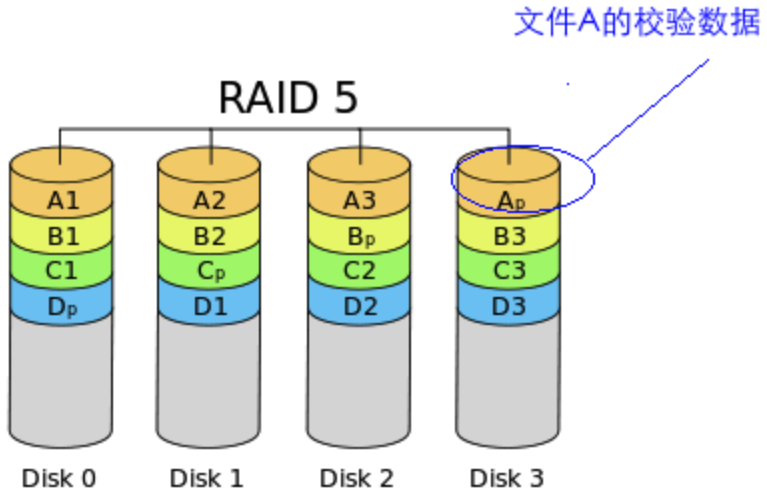


粒度是字节（Byte）。用一个专门的磁盘存放校验纠错数据。

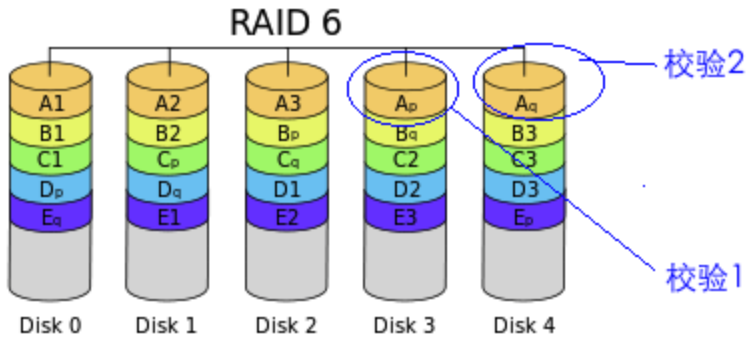


粒度是Block块。可以支持并发。例如，对A1和B2的READ。

RAID图解（3）



粒度是Block；不用专门的校验磁盘。校验数据分布在数据磁盘阵列里。



RAID 6可以坏两个磁盘，仍然正常工作。

粒度是Block；不用专门的校验磁盘。与RAID 5比较，RAID 6用两个校验磁盘。

Level	Description 粒度和相关属性	number of drives 至少的磁盘数目	Fault tolerance 可以容错的磁盘数目
RAID 0	Block-level striping without parity or mirroring	2	0 (none)
RAID 1	Mirroring without parity or striping	2	n−1 drives
RAID 2	Bit-level striping with dedicated Hamming-code parity	3	One drive
RAID 3	Byte-level striping with dedicated parity	3	One drive
RAID 4	Block-level striping with dedicated parity	3	One drive
RAID 5	Block-level striping with distributed parity	3	One drive
RAID 6	Block-level striping with double distributed parity	4	Two drives

[注释：] RAID磁盘阵列的是通过多个小的廉价的磁盘的组合，可以比一个单一的昂贵的大磁盘提供更好的性价比。例如并发的操作和容错。RAID 0是数据分布；RAID 1是备份镜像；之后的都是有校验纠缠磁盘的。RAID 5是基于数据块和分布式存放纠错，可以支持2个磁盘坏还正常工作。