# 1. 系统级I/O

输入输出是在主存和外部设备（如磁盘驱动器、中断和网络）之间复制数据的过程。输入操作是从IO设备复制数据到主存，输出操作是从主存复制数据到IO设备。

## 1.1. Unix IO

一个Linux文件就是一个m个字节的序列。所有的IO设备都被模型化为文件，所有的输入输出操作都被当作对相应文件的读和写来执行。将设备映射为文件的方式，允许Linux内核引出一个简单的、低级的应用接口，即Unix IO，使得输入和输出操作都能以一种统一的方式来执行：

1. **打开文件：**应用程序通过打开文件来告诉内核它想访问一个IO设备。内核返回一个非负整数，即描述符，用于在后续的操作中标识这个设备。内核会记录这个打开文件有关的信息，应用程序只需记住这个描述符。
2. **Linux Shell创建的每个进程开始时都有三个打开的文件**：标准输入（描述符0）、标准输出（描述符1）和标准错误（描述符2）。
3. **改变当前的位置：**对每个打开的文件，Linux内核保持着一个文件位置k，初始为0。k是从文件起始位置的字节偏移量。应用程序可以执行seek操作改变k。
4. **读写文件：**读操作是从文件当前位置k开始，复制n>0个字节到内存，然后更新k为k+n。如果一个文件大小m字节，当k>=m时，会触发一个**EOF(end-of-file)**的条件。写操作是从内存复制n>0个字节到文件，从文件当前位置k开始，然后更新k。
5. **关闭文件：**当应用完成了对文件的访问后，通知内核关闭该文件。内核释放文件打开时创建的数据结构，并将这个描述符恢复到可用的描述符池中。

## 1.2. 文件

每个Linux文件都有一个类型来标识它的角色：

1. **普通文件（binary file）**：包含任意数据，分为文本文件和二进制文件。文本文件只含有ASCII或Unicode字符；其他是二进制文件。
2. **目录（directory）**：包含一组链接的文件，其中每个链接都将一个文件名映射到一个文件，这个文件可能是另外一个目录。
3. **套接字（socket）**：用于与另一个进程通信。

# 2. I/O系统的基本功能

I/O系统管理的主要对象是I/O设备和相应的设备控制器。其最主要的任务是，完成用户提出的I/O请求，提高I/O速率，以及提高设备的利用率，并能为更高层的进程方便地使用这些设备提供手段。为了满足系统和用户的要求，I/O系统应该具有以下几方面的基本功能

1. **隐藏物理设备的细节**：I/O设备类型多，差异大。利用设备控制器（硬件）进行控制。**隐藏物理设备的使用细节，仅向上层提高少量的，抽象的读写命令**。
2. **与设备的无关性**：用户仅提供逻辑设备名来使用设备；例如打印时，它只需要提高读写命令和抽象的逻辑设备名即可，不必要指明是那一台打印机。
3. **提高处理机和I/O设备的利用率**：设备之间、设备与处理机之间均可并行操作。要求CPU快速响应I/O请求，减少对设备运行的干预时间。
4. **对I/O设备进行控制：**对I/O设备进行控制是驱动程序的功能。目前对I/O设备有四种控制方式：①采用轮询的可编程I/O方式；②采用中断的可编程I/O方式；③直接存储器访问方式；④I/O通道方式。具体控制方式与设备的传输速率和传输数据单位有关。
5. **确保对设备的正确共享：**以共享属性来分类，分为独占设备、共享设备。
   1. 独占设备，进程应互斥地访问这类设备，即系统一旦把这类设备分配给了某进程后，便由该进程独占，直至用完释放。典型的独占设备有打印机、磁带机等。系统在对独占设备进行分配时，还应考虑到分配的安全性。
   2. 共享设备，是指在一段时间内允许多个进程同时访问的设备。典型的共享设备是磁盘，当有多个进程需对磁盘执行读、写操作时，可以交叉进行，不会影响到读、写的正确性。
6. **错误处理：**大多数的设备都包括了较多的机械和电气部分，运行时容易出现错误和故障。从处理的角度，可将错误分为临时性错误和持久性错误。对于临时性错误，可通过重试操作来纠正，只有在发生了持久性错误时，才需要向上层报告。

## 2.0. I/O分类

按传输速率分为低速设备、中速设备和高速设备。

按**信息交换的单位**分类，分为：

1. 块设备：**由于信息的存取总是以数据块为单位**，所以存储信息的设备称为块设备。它属于有结构设备，如磁盘等。磁盘设备的基本特征是传输速率较高，以及**可寻址**，即对它可随机地读/写任一块。
2. 字符设备：用于数据输入/输出的设备为字符设备，因为其传输的基本单位是字符。它属于无结构类型，如交互式终端机、打印机等。它们的基本特征是传输速率低、**不可寻址**，并且在输入/输出时常釆用中断驱动方式。

## 2.1. 四种IO控制方式

1. **程序控制IO：**CPU代表一个进程给IO模块发送一个IO命令，该进程进入忙等待，直到操作完成才可以继续执行。
2. **中断驱动方式：**CPU在向IO设备发出读命令后，可以转去做其它的事情，等到IO设备数据就绪，由IO设备主动发出中断请求打断CPU。这样是CPU和设备都可以尽量忙起来。
3. **DMA方式**：DMA方式基本思想是，在主存和IO设备之间直接开辟数据通路，彻底解放CPU。其特点是基本单位是数据块，所传送的数据是从设备直接送入内存的，或者相反。仅仅在一个或多个数据块传输开始或结束时才需要CPU的干预，这个数据块的传输是在DMA控制器的控制下完成的。
4. **通道控制方式**：IO通道是指专门负责输入输出的处理机。它可以进一步减少CPU的干预。IO通道与一般处理机的区别是通道指令类型单一，没有自己的内存，通道所执行的通道程序是放在主机的内存中的，也就是说通道与CPU内存共享。IO通道与DMA方式的区别是，DMA方式需要CPU来控制传输的数据块的大小、位置，而通道可自己控制。DMA方式对应一台设备的数据传递，而通道可以控制多台设备与内存交换。

### 2.1.1. 直接存储器访问DMA

DMA单元可以模拟处理器，能像CPU一样获得系统总线的控制权。DMA技术工作流程如下，当CPU想读或者写一块数据时，它通过向DMA模块发送一下信息来给DMA模块发送一条命令：

1. 请求读操作或写操作的信号，通过在CPU和DMA模块之间使用读写控制线发送；
2. 相关的IO设备地址，通过数据线发送；
3. 从存储器中读或往存储器写的起始地址，在数据线上传送，并由DMA模块保存在其地址寄存器中；
4. 读或写的字数，也通过数据线传送，并由DMA模块保存在其数据寄存器中。

然后CPU继续执行其他工作，此时它已经把这个IO操作委托给DMA模块。DMA模块直接从存储器或往存储器中传送整块数据，一次传送一个字，并且不需要再通过CPU。传送结束后，DMA模块给CPU发送一个中断信号。因此，只有在传送开始和结束时才会用到CPU。

## 2.2. IO子系统的层次结构

层次结构由：用户IO层软件、设备独立性软件、设备驱动程序、中断处理程序组成。

**用户IO层软件**的实现与用户交互的接口，用户可直接调用在用户层提供的、IO操作有关的函数库。

**设备独立性软件**，用于实现用户程序与设备驱动器接口的统一，使得应用程序独立于具体的物理设备设备。主要功能又：执行所有设备的共有操作、想用户层提供接口。

**设备驱动程序**，直接与硬件相关，负责具体实现对设备发出的操作命令。通常每一类设备配备一个设备驱动程序，它是IO进程与设备 控制器之间的通信程序。

**中断处理程序**，用于保存被中断进程的CPU环境，转入响应的中断处理程序进行处理，处理完并恢复中断进程的现场。

# 3. IO缓冲

假设用户要磁盘中读取多个数据块到用户进程地址空间中的一个区域，可以对磁盘单元执行一个IO命令，并等待数据传送完毕。这个等待可以是忙等待，也可以是进程被中断挂起。这种方法存在问题。首先，进程被挂起，等待相对更慢的IO完成。其次，这种挂起干扰了系统的交换策略，甚至可能导致单进程死锁。

缓冲：在输入请求发出前就开始执行输入传送，并且在输出请求发出一段时间后才开始执行输出传送。

## 3.1. 单缓冲

当用户进程发出IO请求时，操作系统给该操作分配一个位于内存中系统部分的缓冲区。输入传送的数据放入到系统缓冲区中，当传送完成时，进程把该块移到用户空间，并立即请求另一块。（预读）用户可以在下一块数据读取的同时，处理已读入的数据块。

## 3.2. 双缓冲

作为对单缓冲方案的改进，可以给操作分配两个缓冲区。在一个进程往一个缓冲区传送数据的同时，操作系统正在清空另一个缓冲区。

## 3.3. 循环缓冲

使用两个以上的缓冲区。

# 4. 磁盘调度

## 4.1. 磁盘性能参数

**寻道时间：**磁头定位到磁道所需要的时间；

**旋转延迟：**将磁盘的待访问地址区域旋转到读/写磁头可访问的位置所需要的时间；

**存取时间：**寻道时间+旋转延迟。

**传输时间：**一旦磁头定位完成，磁头就通过下面旋转的扇区开始执行读操作或者写操作，此即数据传送部分。该部分的时间即为传输时间。

除了存取时间和传输时间外，一次磁盘IO操作通常还会有一些排队延迟。当进程发出一个IO请求时，它必须在一个队列中等待该设备可用。

影响：从磁盘读扇区的顺序，对IO的性能有很大影响！

## 4.2. 磁盘调度策略

不同的扇区读取顺序造成的IO性能差异可以追溯到寻道时间。为提高IO性能，需要减少花费在寻道上的时间。

### 4.2.1. 先进先出FIFO

按顺序处理队列中的项目。该策略具有公平的优点，每个请求都会按接到的顺序得到处理。性能上接近于随机调度。

### 4.2.2. 优先级

基于优先级的调度的控制并不会优化磁盘的利用率，不过可以满足系统的其他目标。比如较短作业和交互作业的优先级更高，而较长作业的优先级较低，使得大量的短作业可以迅速完成IO操作。但可能导致长时间作业饥饿。

### 4.2.3. 最短服务时间优先

选择使磁头臂从当前位置开始移动最少的磁盘IO请求。最短服务时间优先策略总是选择导致最小寻道时间的请求。

### 4.2.4. SCAN

电梯算法：要求磁头臂仅沿一个方向移动，并在途中满足所有未完成的请求，直到它到达这个方向上的最后一个磁道，或者在这个方向上没有别的请求为止，接着反转服务方向，沿相反反向扫描。

# 5. 磁盘高速缓存

### 5.1. 概述

磁盘高速缓存是内存中为磁盘扇区设置的一个缓冲区，它包含有磁盘中某些扇区的副本。当出现一个请求某一特定扇区的IO请求时，首先检测该扇区是否在磁盘高速缓存中。如果在，则该请求通过这个高速缓存来满足；否则，把被请求的扇区读到磁盘高速缓存中。由于访问的局部性现象的存在，当一块数据被取入高速缓存以满足一个IO请求时，很有可能将来还会访问到这一块数据。

### 5.2. 置换策略

最常用的是最近最少使用算法（LRU）：置换在高速缓冲中未被访问的时间最长的块。

另一个方法使最不经常使用页面置换算法(LFU)：置换集合中被访问次数最少的块。

# 详细参考：

<https://blog.csdn.net/qq_44853882/article/details/108284979?utm_medium=distribute.pc_relevant.none-task-blog-baidujs_baidulandingword-2&spm=1001.2101.3001.4242>