Linux 的系统架构

摘要

Linux 系统虽然发行版本众多，但是其基本的系统结构和内核却基本一致，不同版本之间并不太大变化；Linux 系统一般由内核、shell、文件系统和应用程序四部分组成，同时，内核、shell 和文件系统一起形成了基本的操作系统结构，使得用户可以运行程序、管理文件并使用系统。本文主要通过对 Linux 系统的内核和文件系统两个部分进行剖析，深入的理解 Linux 系统结构。从而阐明 Linux 系统之所以被广泛应用，除了其开源之外的主要原因是系统的简洁和强大，能够更好地满足各类用户和各类作业的需要。

关键词：内核、shell、文件系统。

一、Linux 系统内核

内核是操作系统的核心，具有很多最基本的功能，它负责管理系统的进程、内存、设备驱动程序、文件和网络系统，决定着系统的性能和稳定性。Linux 内核由内存管理、进程管理、设备驱动程序、文件系统和网络管理等组成，下面分别对这几部分做以介绍。

1.内存管理

对任何一台计算机而言，其内存以及其它资源都是有限的。为了让有限的物理内存满足应用程序对内存的大需求量，Linux 采用了称为“虚拟内存”的内存管理方式。Linux 将内存划分为容易处理的“内存页”（4KB）。Linux 包括了管理可用内存的方式，以及物理和虚拟映射所使用的硬件机制。

内存管理要管理的不止 4KB 缓冲区。Linux 提供了对 4KB 缓冲区的抽象，例如 slab 分配器。这种内存管理模式使用 4KB 缓冲区为基数，然后从中分配结构，并跟踪内存页使用情况，比如哪些内存页是满的，哪些页面没有完全使用，

哪些页面为空。这样就允许该模式根据系统需要来动态调整内存使用。

为了支持多个用户使用内存，有时会出现可用内存被消耗光的情况，此时，页面可以移出内存并放入磁盘中。这个过程称为交换，因为页面会被从内存交换到硬盘上，内存管理的源代码可以在 ./linux/mm 中找到。

2 .进程管理

进程实际是某特定应用程序的一个运行实体。在 Linux 系统中，能够同时运行多个进程，Linux 通过在短的时间间隔内轮流运行这些进程而实现“多任务”。这一短的时间间隔称为“时间片”，让进程轮流运行的方法称为“进程调度”，完成调度的程序称为调度程序。

进程调度控制进程对 CPU 的访问。当需要选择下一个进程运行时，由调度程序选择最值得运行的进程。可运行进程实际上是仅等待 CPU 资源的进程，如果某个进程在等待其它资源，则该进程是不可运行进程。Linux 使用了比较简单的基于优先级的进程调度算法选择新的进程。

通过多任务机制，每个进程可认为只有自己独占计算机，从而简化程序的编写。每个进程有自己单独的地址空间，并且只能由这一进程访问，这样，操作系统避免了进程之间的互相干扰以及“坏”程序对系统可能造成的危害。为了完成某特定任务，有时需要综合两个程序的功能，例如一个程序输出文本，而另一个程序对文本进行排序。为此，操作系统还提供进程间的通讯机制来帮助完成这样的任务。Linux 中常见的进程间通讯机制有信号、管道、共享内存、信号量和套接字等。 内核通过 SCI 提供了一个应用程序编程接口（API）来创建一个新进程（fork、exec 或 Portable Operating System Interface [POSIX] 函数），

停止进程（kill、exit），并在它们之间进行通信和同步（signal 或者 POSIX 机制）。

3.设备驱动程序

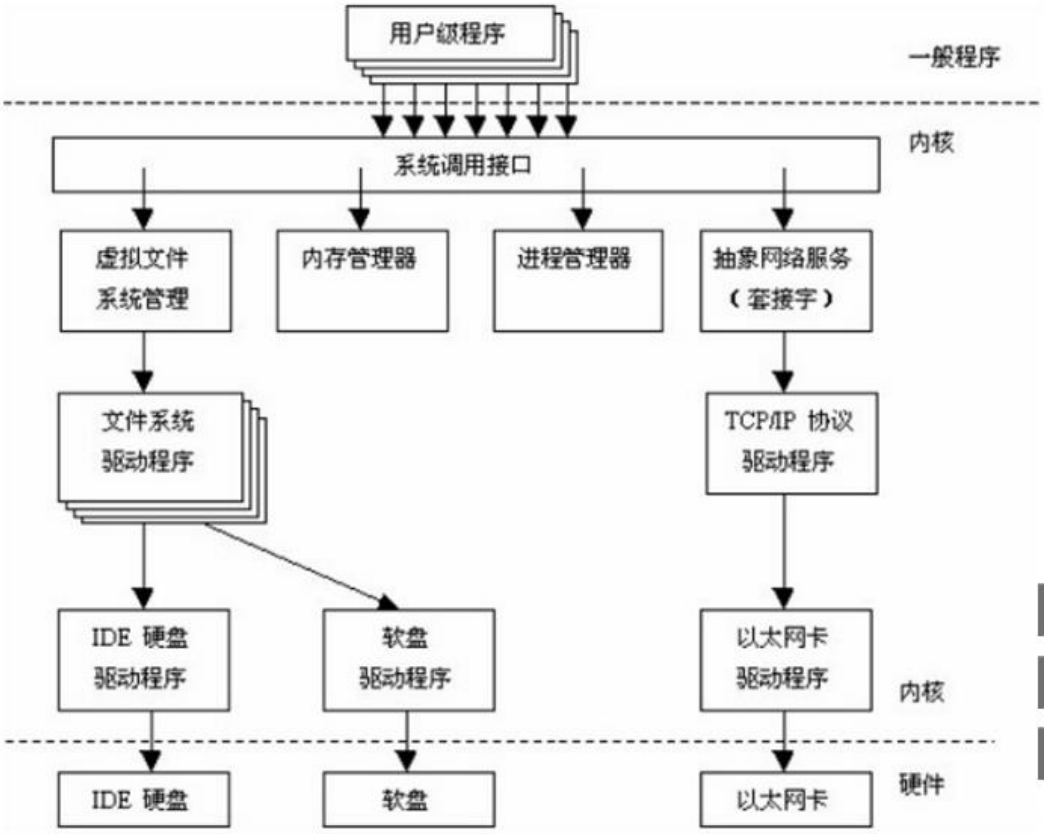
设备驱动程序是 Linux 内核的主要部分。和操作系统的其它部分类似，设备驱动程序运行在高特权级的处理器环境中，从而可以直接对硬件进行操作，但正因为如此，任何一个设备驱动程序的错误都可能导致操作系统的崩溃。设备驱动程序实际控制操作系统和硬件设备之间的交互。设备驱动程序提供一组操作系统可理解的抽象接口完成和操作系统之间的交互，而与硬件相关的具体操作细节由设备驱动程序完成。一般而言，设备驱动程序和设备的控制芯片有关，例如，如果计算机硬盘是 SCSI 硬盘，则需要使用 SCSI 驱动程序，而不是 IDE 驱动程序。

4.网络接口（NET）

提供了对各种网络标准的存取和各种网络硬件的支持。网络接口可分为网络协议和网络驱动程序。网络协议部分负责实现每一种可能的网络传输协议。众所周知，TCP/IP 协议是 Internet 的标准协议，同时也是事实上的工业标准。Linux 的网络实现支持 BSD 套接字，支持全部的 TCP/IP 协议。Linux 内核的网络部分由 BSD 套接字、网络协议层和网络设备驱动程序组成。

网络设备驱动程序负责与硬件设备通讯，每一种可能的硬件设备都有相应的设备驱动程序。

综上所述，一个形象的框图如下所示：



二、Linux 文件系统

Linux 操作系统中单独的文件系统并不是由驱动器号或驱动器名称（如 A:或 C:等）来标识的，而是将独立的文件系统组合成了一个层次化的树形结构，并且由一个单独的实体代表这一文件系统。Linux 将新的文件系统通过一个称为“挂装”或“挂上”的操作将其挂装到某个目录上，从而让不同的文件系统结合成为一个整体。Linux 操作系统的一个重要特点是它支持许多不同类型的文件系统。Linux 中最普遍使用的文件系统是 Ext2，它也是 Linux 土生土长的文件系统。但 Linux 也能够支持 FAT、VFAT、FAT32、MINIX 等不同类型的文件系统，从而可以方便地和其它操作系统交换数据。Linux 支持许多不同的文件系统，并且将它们组织成了一个统一的虚拟文件系统.

虚拟文件系统（VirtualFileSystem,VFS）:隐藏了各种硬件的具体细节，把文件系统操作和不同文件系统的具体实现细节分离了开来，为所有的设备提供

了统一的接口，VFS 提供了多达数十种不同的文件系统接口。虚拟文件系统可以分为逻辑文件系统和设备驱动程序。逻辑文件系统指 Linux 所支持的文件系统，如 ext2,fat 等，设备驱动程序指为每一种硬件控制器所编写的设备驱动程序模块。

虚拟文件系统（VFS）是 Linux 内核中非常有用的一个方面，因为它为文件系统提供了一个通用的接口抽象。VFS 在 SCI 和内核所支持的文件系统之间提供了一个交换层。即 VFS 在用户和文件系统之间提供了一个交换层。在 VFS 上面，是对诸如 open、close、read 和 write 之类的函数的一个通用 API 抽象。在 VFS 下面是文件系统抽象，它定义了上层函数的实现方式。它们是给定文件系统（超过 50 个）的插件。文件系统的源代码可以在 ./linux/fs 中找到。

文件系统层之下是缓冲区缓存，它为文件系统层提供了一个通用函数集（与具体文件系统无关）。这个缓存层通过将数据保留一段时间（或者随即预先读取数据以便在需要时就可用）优化了对物理设备的访问。缓冲区缓存之下是设备驱动程序，它实现了特定物理设备的接口。因此，用户和进程不需要知道文件所在的文件系统类型，而只需要象使用 Ext2 文件系统中的文件一样使用它们。

1.文件类型

Linux 下面的文件类型主要有：

1）普通文件：C 语言元代码、SHELL 脚本、二进制的可执行文件等。分为纯文本和二进制。

2）目录文件：目录，存储文件的唯一地方。

3）链接文件：指向同一个文件或目录的的文件。

4）设备文件：与系统外设相关的，通常在/dev 下面。分为块设备和字符设备。

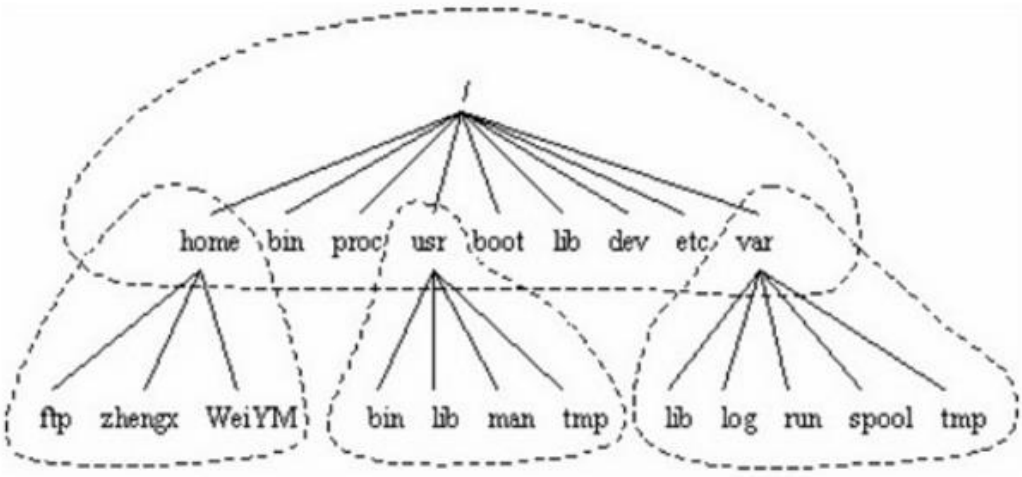
5）管道(FIFO)文件:提供进程建通信的一种方式

6）套接字(socket) 文件：该文件类型与网络通信有关

2. Linux 目录

文件结构是文件存放在磁盘等存贮设备上的组织方法。主要体现在对文件和目录的组织上。目录提供了管理文件的一个方便而有效的途径。Linux 使用标准的目录结构，在安装的时候，安装程序就已经为用户创建了文件系统和完整而固定的目录组成形式，并指定了每个目录的作用和其中的文件类型。

完整的目录树可划分为小的部分，这些小部分又可以单独存放在自己的磁盘或分区上。这样，相对稳定的部分和经常变化的部分可单独存放在不同的分区中，从而方便备份或系统管理。目录树的主要部分有 root、/usr、/var、/home 等。这样的布局可方便在 Linux 计算机之间共享文件系统的某些部分。如图：



Linux 采用的是树型结构。最上层是根目录，其他的所有目录都是从根目录出发而生成的。微软的 DOS 和 windows 也是采用树型结构，但是在 DOS 和 windows 中这样的树型结构的根是磁盘分区的盘符，有几个分区就有几个树型结构，他们之间的关系是并列的。但是在 linux 中，无论操作系统管理几个磁盘分区，这样

的目录树只有一个。从结构上讲，各个磁盘分区上的树型目录不一定是并列的。

3. Linux 磁盘分区

（1）**主分区，扩展分区和逻辑分区：**linux 分区不同于 windows,硬盘和硬盘分区在 Linux 都表示为设备.硬盘分区一共有三种：主分区，扩展分区和逻辑分区。硬盘的分区主要分为主分区(Primary Partion)和扩展分区(Extension Partion)两种，主分区和扩展分区的数目之和不能大于四个。主分区(Primary Partion)：可以马上被使用但不能再分区。扩展分区(Extension Partion)：必须再进行分区后才能使用，也就是说它必须还要进行二次分区。逻辑分区（(Logical Partion)）：由扩展分区建立起来的分区。逻辑分区没有数量上限制。扩展分区只不过是逻辑分区的“容器”，实际上只有主分区和逻辑分区进行数据存储。(2) **Linux 下硬盘分区的标识：**硬盘分区的标识一般使用/dev/hd[a-z]X 或者 /dev/sd[a-z]X 来标识，其中[a-z]代表硬盘号，X 代表硬盘内的分区号。整块硬盘分区的块号标识:Linux 下用 hda、hdb、sda、sdb 等来标识不同的硬盘;其中：IDE 接口硬盘：表示为/dev/hda1、/dev/hdb...；SCSI 接口的硬盘、SATA 接口的硬盘表示为/dev/sda、/dev/sdb ... ... ；硬盘内的分区：如果 X 的值是 1 到 4,表示硬盘的主分区（包含扩展分区）；逻辑分区从是从 5 开始的，比如 /dev/hda5 肯定是逻辑分区了；例如：用 hda1、hda2、 hda5、hda6 来标识不同的分区。其中，字母 a 代表第一块硬盘，b 代表第二块硬盘，依次类推。而数字 1 代表一块硬盘的第一个分区、2 代表第二个分区，依次类推。1 到 4 对应的是主分区(Primary Partition)或扩展分区(Extension Partition)。从 5 开始，对应的都是硬盘的逻辑分区(Logical Partition)。一块硬盘即使只有一个主分

区，逻辑分区也是从 5 开始编号的，这点应特别注意。总结：一个硬盘分区首先要大确认在哪个硬盘，然后再确认它所在硬盘内的哪个分区。

（3）Linux 下磁盘分区和目录的关系如下：

- 任何一个分区都必须挂载到某个目录上。
- 目录是逻辑上的区分。分区是物理上的区分。
- 磁盘 Linux 分区都必须挂载到目录树中的某个具体的目录上才能进行读写操作。
- 根目录是所有 Linux 的文件和目录所在的地方，需要挂载上一个磁盘分区。

4 .Linux 主要目录的功用。

/bin 二进制可执行命令

/dev 设备特殊文件

/etc 系统管理和配置文件

/etc/rc.d 启动的配置文件和脚本

/home 用户主目录的基点，比如用户 user 的主目录就是/home/user，可以用~user 表示

/lib 标准程序设计库，又叫动态链接共享库，作用类似 windows 里的.dll 文件

/sbin 系统管理命令，这里存放的是系统管理员使用的管理程序

/tmp 公用的临时文件存储点

/root 系统管理员的主目录（呵呵，特权阶级）

/mnt 系统提供这个目录是让用户临时挂载其他的文件系统。

/lost+found 这个目录平时是空的，系统非正常关机而留下“无家可归”的文件（windows 下叫什么.chk）就在这里

/proc 虚拟的目录，是系统内存的映射。可直接访问这个目录来获取系统信息。

/var 某些大文件的溢出区，比方说各种服务的日志文件

/usr 最庞大的目录，要用到的应用程序和文件几乎都在这个目录。其中包含：

/usr/X11R6 存放 X window 的目录

/usr/bin 众多的应用程序

/usr/sbin 超级用户的一些管理程序

/usr/doc linux 文档

/usr/include linux 下开发和编译应用程序所需要的头文件

/usr/lib 常用的动态链接库和软件包的配置文件

/usr/man 帮助文档

/usr/src 源代码，linux 内核的源代码就放在/usr/src/linux 里

/usr/local/bin 本地增加的命令

/usr/local/lib 本地增加的库

5. Linux 文件系统特性：

磁盘分区完毕后还需要进行格式化(format)，之后操作系统才能够使用这个分区。格式化的目的是能使操作系统可以使用文件系统格式（即我们上面提到文件系统类型）。每种操作系统能够使用的文件系统并不相同。如 windows 98 以前的微软操作系统主要利用的文件系统是 FAT (或 FAT16)，windows 2000 以后的版本有所谓的 NTFS 文件系统，至于 Linux 的正统文件系统则为 Ext2(Linux second extended file system, ext2fs)这一个。此外，在默认的

情况下，windows 操作系统是不会认识 Linux 的 Ext2 的。传统的磁盘与文件系统之应用中，一个分区就是只能够被格式化成为一个文件系统，所以我们可以说一个 filesystem 就是一个 partition。但是由于新技术的利用，例如我们常听到的 LVM 与软件磁盘阵列(software raid)， 这些技术可以将一个分区格式化为多个文件系统(例如 LVM)，也能够将多个分区合成一个文件系统(LVM, RAID)！所以说，目前我们在格式化时已经不再说成针对 partition 来格式化了， 通常我们可以称呼一个可被挂载的数据为一个文件系统而不是一个分区喔！

那么文件系统是如何运行的呢？这与操作系统的文件数据有关。较新的操作系统的文件数据除了文件实际内容外， 通常含有非常多的属性，例如 Linux 操作系统的文件权限(rwx)与文件属性(拥有者、群组、时间参数等)。文件系统通常会将这两部份的数据分别存放在不同的区块，权限与属性放置到 inode 中，至于实际数据则放置到 data block 区块中。 另外，还有一个超级区块(superblock) 会记录整个文件系统的整体信息，包括 inode 与 block 的总量、使用量、剩余量等。对于一个磁盘分区来说，在被指定为相应的文件系统后，整个分区被分为 1024，2048 和 4096 字节大小的块。根据块使用的不同，可分为：

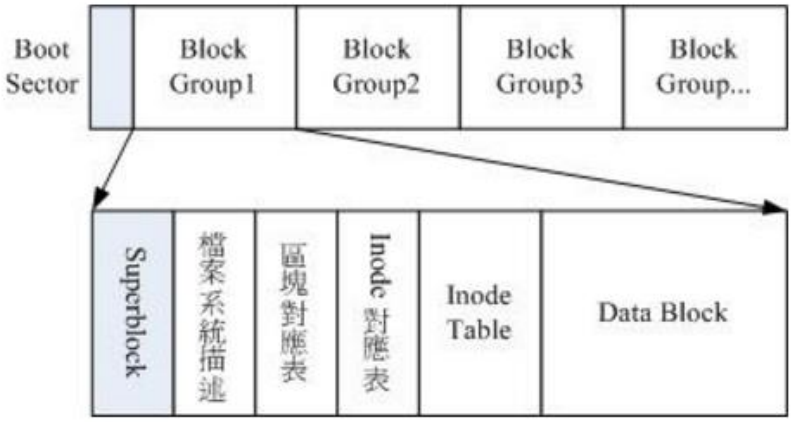
（1）超级块(Superblock):这是整个文件系统的第一块空间。包括整个文件系统的基本信息，如块大小，inode/block 的总量、使用量、剩余量，指向空间 inode 和数据块的指针等相关信息。

（2）inode 块(文件索引节点): 文件系统索引,记录文件的属性。它是文件系统的最基本单元，是文件系统连接任何子目录、任何文件的桥梁。每个子目录和文件只有唯一的一个 inode 块。它包含了文件系统中文件的基本属性(文件的长度、创建及修改时间、权限、所属关系)、存放数据的位置等相关信息。

（3）数据块(Block):实际记录文件的内容，若文件太大时，会占用多个 block。为了提高目录访问效率,Linux 还提供了表达路径与 inode 对应关系的 dentry 结构。它描述了路径信息并连接到节点 inode，它包括各种目录信息，还指向了 inode 和超级块。

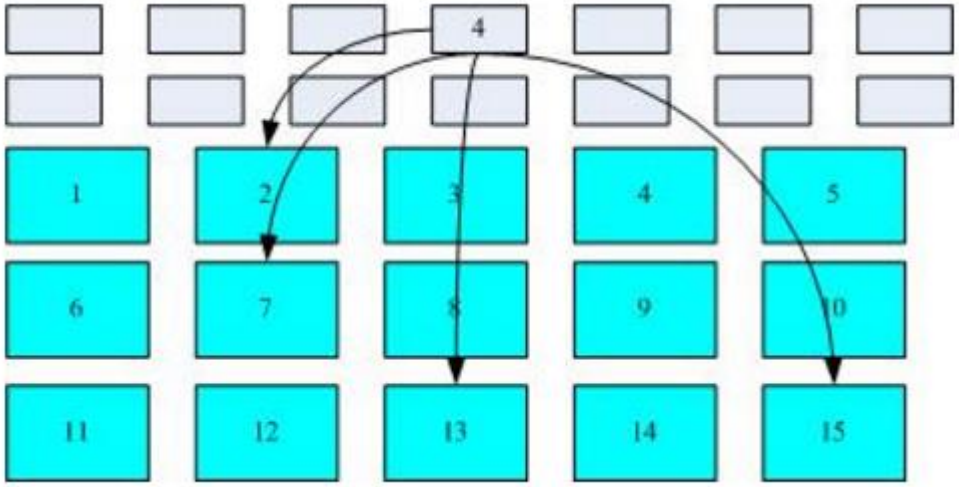
就像一本书有封面、目录和正文一样。在文件系统中，超级块就相当于封面，从封面可以得知这本书的基本信息； inode 块相当于目录，从目录可以得知各章节内容的位置；而数据块则相当于书的正文，记录着具体内容。

Linux 正统的文件系统(如 ext2、3 等)将硬盘分区时会划分出超级块、inode Table 区块和 data block 数据区域。一个文件由一个超级块、inode 和数据区域块组成。Inode 包含文件的属性(如读写属性、owner 等，以及指向数据块的指针)，数据区域块则是文件内容。当查看某个文件时，会先从 inode table 中查出文件属性及数据存放点，再从数据块中读取数据。如图：



将 inode 与 block 区块用图解来说明一下，如下图所示，文件系统先格式化出 inode 与 block 的区块，假设某一个文件的属性与权限数据是放置到 inode 4 号(下图较小方格内)，而这个 inode 记录了文件数据的实际放置点为 2, 7, 13, 15 这四个 block 号码，此时我们的操作系统就能够据此来排列磁盘

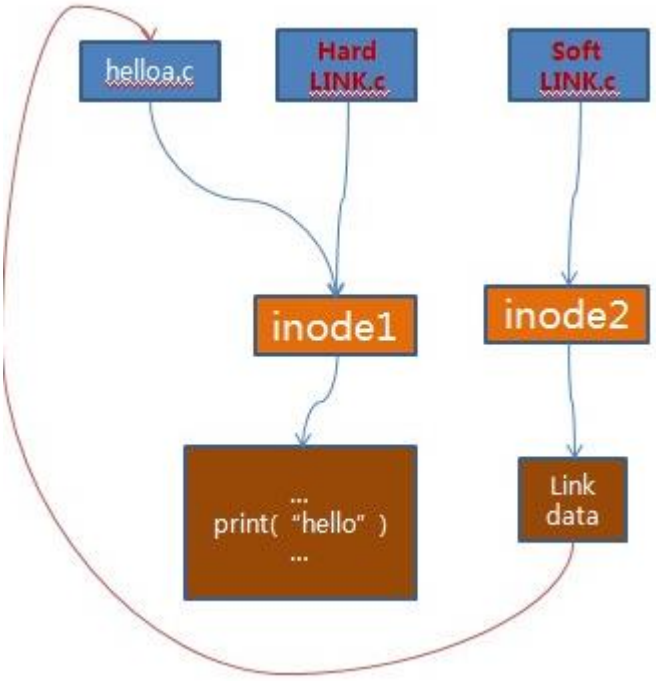
的阅读顺序，可以一口气将四个 block 内容读出来！ 那么数据的读取就如同下图中的箭头所指定的模样了。



Ext2 是索引式文件系统，基本上不太需要常常进行碎片整理。但是如果文件系统使用太久，常常删除/编辑/新增文件时，那么还是可能会造成文件数据太过于离散的问题，此时或许会需要进行重整一下的。

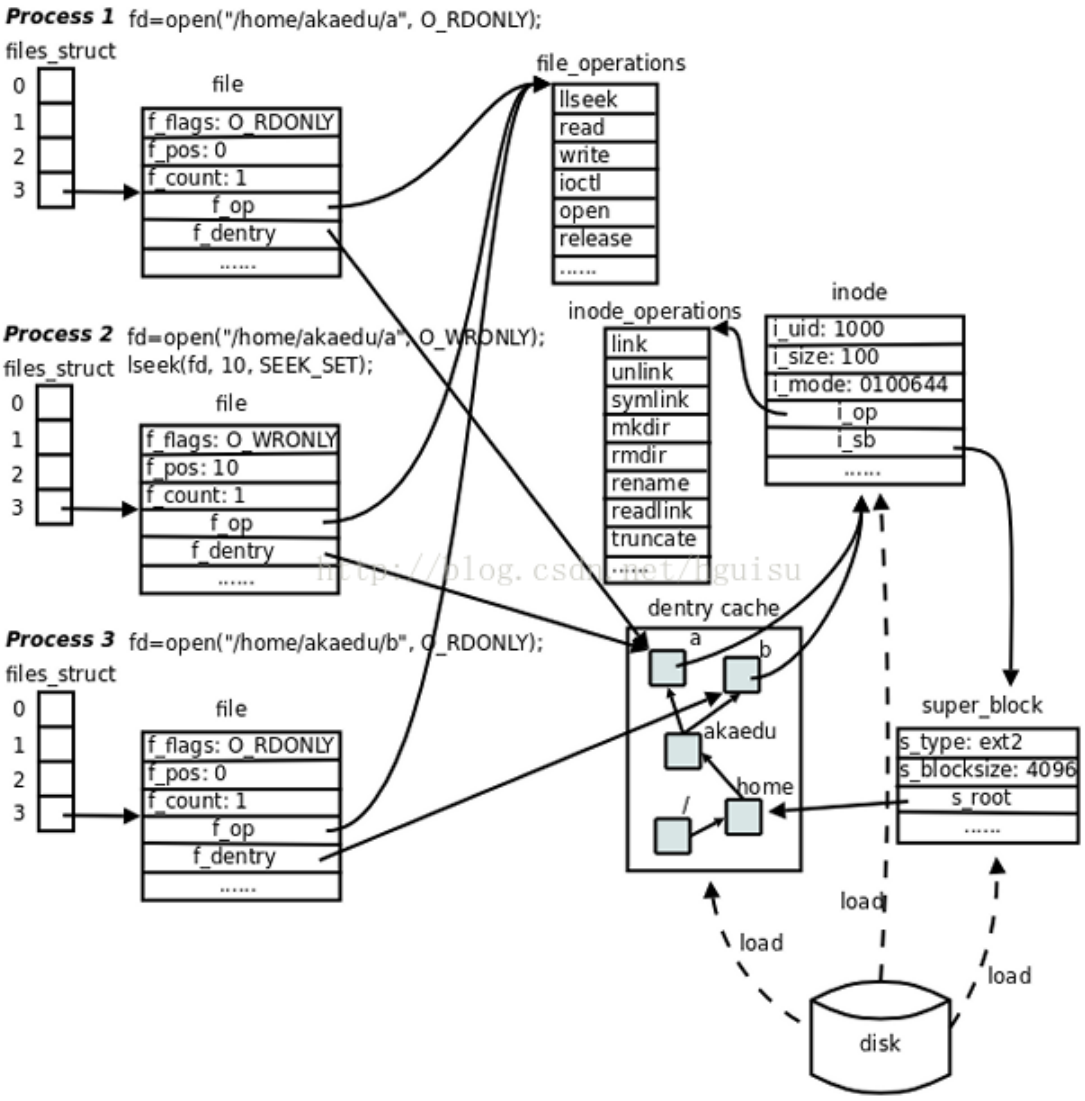
可以用 ln 命令对一个已经存在的文件再建立一个新的连接，而不复制文件的内容。连接有软连接和硬连接之分，软连接又叫符号连接。它们各自的特点是：硬连接：原文件名和连接文件名都指向相同的物理地址。目录不能有硬连接；硬连接不能跨越文件系统（不能跨越不同的分区）文件在磁盘中只有一个拷贝，节省硬盘空间；由于删除文件要在同一个索引节点属于唯一的连接时才能成功，因此可以防止不必要的误删除。符号连接：用 ln -s 命令建立文件的符号连接符号连接是 linux 特殊文件的一种，作为一个文件，它的数据是它所连接的文件的路径名。类似 windows 下的快捷方式。可以删除原有的文件而保存连接文件，没有防止误删除功能。两者区别如下：（1）硬链接原文件和新文件的 inode 编号一致。而软链接不一样。（2）对原文件删除，会导致软链接不可用，而硬链接不

受影响。（3）对原文件的修改，软、硬链接文件内容也都修改，因为是指向同一个文件的。如图所示：



6. 文件系统在内核中的表示

Linux 内核的 VFS 子系统可以图示如下：



文件与 IO：每个进程在 PCB（Process Control Block）中都保存着一份文件描述符表，文件描述符就是这个表的索引，每个表项都有一个指向已打开文件的指针，已打开的文件在内核中用 file 结构体表示，文件描述符表中的指针指向 file 结构体。在 file 结构体中维护 File Status Flag（file 结构体的成员 f_flags）和当前读写位置（file 结构体的成员 f_pos）。在上图中，进程 1 和进程 2 都打开同一文件，但是对应不同的 file 结构体，因此可以有不同的 File Status Flag 和读写位置。file 结构体中比较重要的成员还有 f_count，表示引用计数（Reference Count），后面我们会讲到，dup、fork 等系统调用会导致

多个文件描述符指向同一个 file 结构体，例如有 fd1 和 fd2 都引用同一个 file 结构体，那么它的引用计数就是 2，当 close(fd1)时并不会释放 file 结构体，而只是把引用计数减到 1，如果再 close(fd2)，引用计数就会减到 0 同时释放 file 结构体，这才真的关闭了文件。每个 file 结构体都指向一个 file_operations 结构体，这个结构体的成员都是函数指针，指向实现各种文件操作的内核函数。比如在用户程序中 read 一个文件描述符，read 通过系统调用进入内核，然后找到这个文件描述符所指向的 file 结构体，找到 file 结构体所指向的 file_operations 结构体，调用它的 read 成员所指向的内核函数以完成用户请求。在用户程序中调用 lseek、read、write、ioctl、open 等函数，最终都由内核调用 file_operations 的各成员所指向的内核函数完成用户请求。

file_operations 结构体中的 release 成员用于完成用户程序的 close 请求，之所以叫 release 而不叫 close 是因为它不一定真的关闭文件，而是减少引用计数，只有引用计数减到 0 才关闭文件。对于同一个文件系统上打开的常规文件来说，read、write 等文件操作的步骤和方法应该是一样的，调用的函数应该是相同的，所以图中的三个打开文件的 file 结构体指向同一个 file_operations 结构体。如果打开一个字符设备文件，那么它的 read、write 操作肯定和常规文件不一样，不是读写磁盘的数据块而是读写硬件设备，所以 file 结构体应该指向不同的 file_operations 结构体，其中的各种文件操作函数由该设备的驱动程序实现。每个 file 结构体都有一个指向 dentry 结构体的指针，“dentry”是 directory entry（目录项）的缩写。我们传给 open、stat 等函数的参数的是一个路径，例如/home/akaedu/a，需要根据路径找到文件的 inode。为了减少读盘次数，内核缓存了目录的树状结构，称为 dentry cache，其中每个节点是一个 dentry 结构

体，只要沿着路径各部分的 dentry 搜索即可，从根目录/找到 home 目录，然后找到 akaedu 目录，然后找到文件 a。dentry cache 只保存最近访问过的目录项，如果要找的目录项在 cache 中没有，就要从磁盘读到内存中。每个 dentry 结构体都有一个指针指向 inode 结构体。inode 结构体保存着从磁盘 inode 读上来的信息。在上图的例子中，有两个 dentry，分别表示/home/akaedu/a 和/home/akaedu/b，它们都指向同一个 inode，说明这两个文件互为硬链接。inode 结构体中保存着从磁盘分区的 inode 读上来信息，例如所有者、文件大小、文件类型和权限位等。每个 inode 结构体都有一个指向 inode_operations 结构体的指针，后者也是一组函数指针指向一些完成文件目录操作的内核函数。和 file_operations 不同，inode_operations 所指向的不是针对某一个文件进行操作的函数，而是影响文件和目录布局的函数，例如添加删除文件和目录、跟踪符号链接等等，属于同一文件系统的各 inode 结构体可以指向同一个 inode_operations 结构体。inode 结构体有一个指向 super_block 结构体的指针。super_block 结构体保存着从磁盘分区的超级块读上来的信息，例如文件系统类型、块大小等。super_block 结构体的 s_root 成员是一个指向 dentry 的指针，表示这个文件系统的根目录被 mount 到哪里，在上图的例子中这个分区被 mount 到/home 目录下。file、dentry、inode、super_block 这几个结构体组成了 VFS 的核心概念。对于 ext2 文件系统来说，在磁盘存储布局上也有 inode 和超级块的概念，所以很容易和 VFS 中的概念建立对应关系。

7. 挂载文件系统

linux 系统中每个分区都是一个文件系统，都有自己的目录层次结构。linux 会将这些分属不同分区的、单独的文件系统按一定的方式形成一个系统的总的目录层次结构。这里所说的“按一定方式”就是指的挂载。 将一个文件系统的顶层目录挂到另一个文件系统的子目录上，使它们成为一个整体，称为挂载。把该子目录称为挂载点。例如要读取硬盘中的一个格式化好的分区、光盘或软件等设备时，必须先把这些设备对应到某个目录上，而这个目录就称为“挂载点（mount point）”，这样才可以读取这些设备。 挂载后将物理分区细节屏蔽掉，用户只有统一的逻辑概念，所有的东西都是文件。注意：（1）挂载点必须是一个目录；（2）一个分区挂载在一个已存在的目录上，这个目录可以不为空，但挂载后这个目录下以前的内容将不可用。

挂载时使用 mount 命令，其格式：mount [-参数] [设备名称] [挂载点]

其中常用的参数有：

-t 指定设备的文件系统类型（什么提到的文件类型）

-o 指定挂载文件系统时的选项。有些也可用在/etc/fstab 中。常用的有

codepage=XXX 代码页

iocharset=XXX 字符集

ro 以只读方式挂载

rw 以读写方式挂载

nouser 使一般用户无法挂载

user 可以让一般用户挂载设备

（1）挂载 windows 的文件系统：

①首先使用 sudo fdisk -l 查看挂载的设备，例如最下面有：/dev/hda5；

②mkdir 创建一个目录，这里的目录是作为挂在目录，就是你要把 E 盘挂到这个目录下：mk /mnt/winc；

③windows 和 linux 使用的不是一个文件系统，一般情况下 linux 不挂载 windows 文件系统，所以要你手动 mount。mount -t vfat /dev/hda5 /mnt/winc （ -t vfat 指出这里的文件系统 fat32）现在就可以进入 /mnt/winc 等目录读写这些文件了。

（2）挂载光盘：

 # mk /mnt/cdrom

mount -t iso9660 /dev/cdrom /mnt/cdrom (关盘的名字一般都是 cdrom，这条命令一般都通用)

（3）虚拟机共享文件夹：

例如在 VirtualBox 下，主机是 Windows，Ubuntu 是 Guest。共分三步：

①首先要安装虚拟电脑工具包：在 VirtualBox 的菜单里选择"设备"->"安装虚拟电脑工具包"，你会发现在 Ubuntu 桌面上多出一个光盘图标，这张光盘默认被自动加载到了文件夹/media/cdom0，而且/cdrom 自动指向这个文件夹。默认设置下文件管理器会自动打开这张光盘，可以看到里面有个"VBoxLinuxAdditions.run"文件。打开一个命令行终端,依次输入"cd /cdrom"和"sudo sh ./VBoxLinuxAdditions.run"，不含双引号，开始安装工具包。安装完毕，会用英文提示要重启 Ubuntu，建议立刻重启。重启后，比较明显的变化是鼠标是共享模式，并且剪贴板也和 Windows 共享了。如果有这些变化，说明虚拟电脑工具包已经装成功。

②下一步设置共享文件夹。在共享文件夹设置窗口中，单击右侧的"添加

一个共享文件夹"，路径选择你想要共享的 Windows 文件夹，共享名任取一个自己喜欢的,比如"myshare",选项 read-only 是指是否只允许 ubuntu 读这个文件夹，请根据需要选择这个选项。

③在 ubuntu 下挂载这个共享文件夹:sudo mount -t vboxsf myshare /media/share，其中"myshare"是之前取的共享文件夹的名字，"/media/share"是要挂载到的目标文件.

三、总结

综上所述，Linux 内核和 Linux 文件系统的，作为 Linux 的系统结构中最最基本的部分，是 Linux 系统实现其他功能的基础；shell 是系统的用户界面，提供了用户与内核进行交互操作的一种接口。它接收用户输入的命令并把它送入内核去执行，是一个命令解释器。另外，shell 编程语言具有普通编程语言的很多特点，用这种编程语言编写的 shell 程序与其他应用程序具有同样的效果，正因如此，所以在本文中，并未对 shell 做过多介绍。

四、参考文献

[1]鸟哥.《鸟哥的 Linux 私房菜 基础篇》.[M].北京：人民邮电出版社，2010.

[2]Linux 系统目录结构-作用.

http://wenku.baidu.com/view/103dd4212f60ddccda38a06a.html.

[3]Linux 文件系统剖析.

http://www.ibm.com/developerworks/cn/linux/l-linux-filesystem/index.html.

[4]Linux 文件系统结构介绍.

https://linux.cn/article-6132-1.html.

[5]Linu 系 统 结 构 详 解.http://blog.csdn.net/hguisu/article/details/6122513.