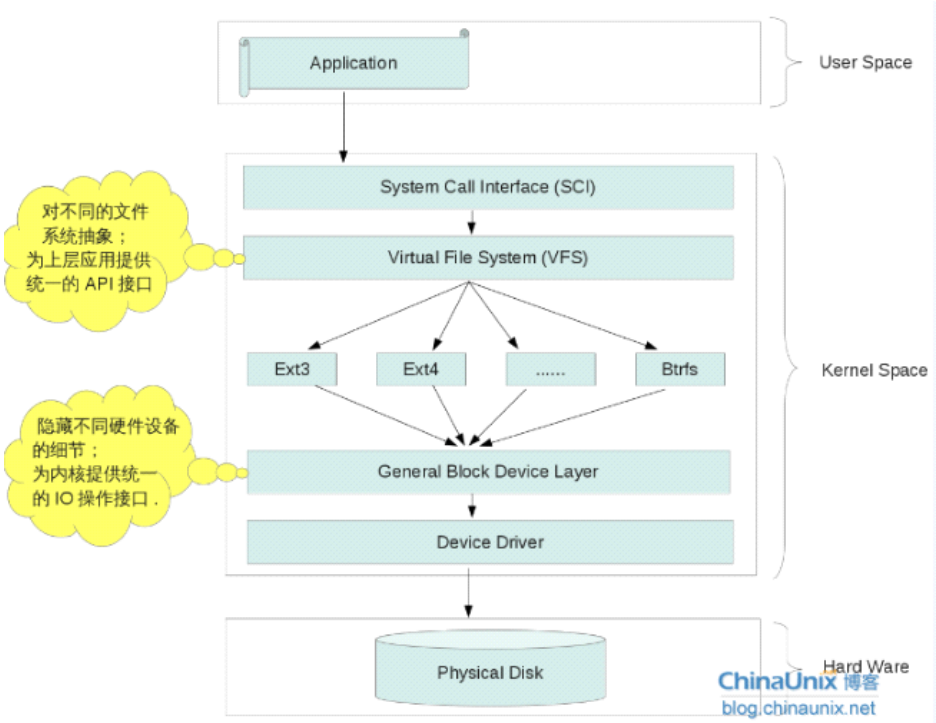
Linux磁盘与文件系统

在LINUX系统中有一个重要的概念：一切都是文件。 其实这是UNIX哲学的一个体现，而Linux是重写UNIX而来，所以这个概念也就传承了下来。在UNIX系统中，把一切资源都看作是文件，包括硬件设备。UNIX系统把每个硬件都看成是一个文件，通常称为设备文件，这样用户就可以用读写文件的方式实现对硬件的访问。



1. 硬盘驱动
   常见的硬盘类型有SATA。
2. General Block Device Layer
   这一层的作用：不同的硬盘驱动，会提供不同的IO接口，内核认为这种杂乱的接口，不利于管理，需要把这些接口抽象一下，形成一个统一的对外接口，这样，不管你是什么硬盘，什么驱动，对外而言，它们所提供的IO接口没什么区别，都一视同仁的被看作块设备来处理。
3. 文件系统
   目前大多Linux发行版本默认使用的文件系统一般是ext4。
4. 虚拟文件系统(VFS)
   Virtual File System这一层的作用：当我们通过mkfs.xxx系列命令创建了很多不同的文件系统，但这些文件系统都有各自的API接口，而用户想要的是，不管你是什么API，他们只关心mount/umount，或open/close等操作。所以，VFS就把这些不同的文件系统做一个抽象，提供统一的API访问接口，这样，用户空间就不用关心不同文件系统中不一样的API了。

物理磁盘到文件系统

文件最终是保存在硬盘上的。硬盘最基本的组成部分是由坚硬金属材料制成的涂以磁性介质的盘片，不同容量硬盘的盘片数不等。

每个盘片有两面，都可记录信息。盘片被分成许多扇形的区域，每个区域叫一个扇区，在DOS中每扇区是128×2的2次方＝512字节。

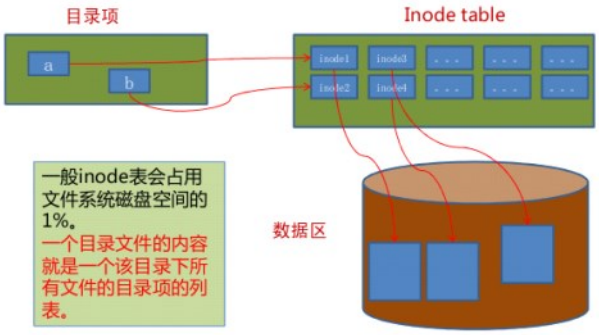
unix/linux管理硬盘，将磁盘块分为以下三个部分：

1) 超级块，文件系统中第一个块被称为超级块。这个块存放文件系统本身的结构信息。比如，超级块记录了每个区域的大小，超级块也存放未被使用的磁盘块的信息。
2) I-结点表。超级块的下一个部分就是i-节点表。每个i-节点就是一个对应一个文件/目录的结构，这个结构它包含了一个文件的长度、创建及修改时间、权限、所属关系、磁盘中的位置等信息。一个文件系统维护了一个索引节点的数组，每个文件或目录都与索引节点数组中的唯一一个元素对应。系统给每个索引节点分配了一个号码，也就是该节点在数组中的索引号，称为索引节点号
3) 数据区。文件系统的第3个部分是数据区。文件的内容保存在这个区域。磁盘上所有块的大小都一样。如果文件包含了超过一个块的内容，则文件内容会存放在多个磁盘块中。一个较大的文件很容易分布上千个独产的磁盘块中。

Linux正统的文件系统(如ext2、ext3)一个文件由目录项、inode和数据块组成。
目录项:包括文件名和inode节点号。
Inode：又称文件索引节点，是文件基本信息的存放地和数据块指针存放地。
数据块：文件的具体内容存放地。

Linux正统的文件系统(如ext2、3等)将硬盘分区时会划分出目录块、inode Table区块和data block数据区域。一个文件由一个目录项、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。

文件存储结构大概如下：



其中目录项的结构如下(每个文件的目录项存储在改文件所属目录的文件内容里)：

创建一个文件的过程

我们从前面可以知道文件的内容和属性是分开存放的，那么又是如何管理它们的呢?现在我们以创建一个文件为例来讲解。
在命令行输入命令：
$ who > userlist
当完成这个命令时。文件系统中增加了一个存放命令who输出内容的新文件userlist，那么这整个过程到底是怎么回事呢？
文件主要有属性、内容以及文件名三项。内核将文件内容存放在数据区，文件属性存放在i-节点，文件名存放在目录中。
创建成功一个文件主要有以下四个步骤：

1. 存储属性 也就是文件属性的存储，内核先找到一块空的i-节点。例如，内核找到i-节点号921130。内核把文件的信息记录其中。如文件的大小、文件所有者、和创建时间等。
2. 存储数据 即文件内容的存储，由于该文件需要3个数据块。因此内核从自由块的列表中找到3个自由块。如600、200、992，内核缓冲区的第一块数据复制到块600，第二和第三分别复制到922和600.
3. 记录分配情况，数据保存到了三个数据块中。所以必须要记录起来，以后再找到正确的数据。分配情况记录在文件的i-节点中的磁盘序号列表里。这3个编号分别放在最开始的3个位置。
4. 添加文件名到目录，新文件的名字是userlist 内核将文件的入口(47,userlist)添加到目录文件里。文件名和i-节点号之间的对应关系将文件名和文件和文件的内容属性连接起来，找到文件名就找到文件的i-节点号，通过i-节点号就能找到文件的属性和内容。

一、硬盘接口类型

硬盘的接口主要有IDE、SATA、SCSI、SAS和光纤通道等五种类型。

1. IDE和SATA接口硬盘多用于家用产品中，也有部分应用于服务器，SATA是一种新生的硬盘接口类型，已经取代了大部分IDE接口应用。
2. SCSI 、SAS主要应用于服务器上，普通家用设备一般不支持SCSI和SAS接口。SAS也是是一种新生的硬盘接口类型，可以和SATA以及部分SCSI设备无缝结合。
3. 光纤通道最初设计也不是为了硬盘设计开发的接口，是专门为网络系统设计的，但随着存储系统对速度的需求，才逐渐应用到硬盘系统中，并且其只应用在高端服务器上（价格昂贵）。

二、硬盘和分区

Linux中主要有两种分区类型，分别为MBR（Master Boot Record）和GPT（GUID Partition Table），是在磁盘上存储分区信息的两种不同方式。这些分区信息包含了分区从哪里开始的信息，这样操作系统才知道哪个扇区是属于哪个分区的，以及哪个分区是可以启动的。在磁盘上创建分区时，你必须在MBR和GPT之间做出选择。

在Linux中会把设备映射成为一个/dev目录下的系统文件，IDE接口类型的硬盘设备映射的文件名称前缀为“hd”，SCSI、SATA、SAS等接口的硬盘设备映射的文件名称前缀为“sd”（部分虚拟机或者云主机的名称可能是其他的，比如“vd”），后面拼接从“a”开始一直到“z”用来区分不同的硬盘设备，在硬盘名称后面拼接数字形式的分区号用来区分不同的分区。

1、MBR分区

MBR的意思是“主引导记录”，它是存在于硬盘驱动器开始部分的一个特殊的启动扇区。这个扇区包含了已安装的操作系统的启动加载器grub和驱动器的逻辑分区信息。

MBR支持最大2TB磁盘，所以它无法处理大于2TB容量的磁盘。

MBR格式的磁盘分区分为主分区和扩展分区。主分区总数不能大于4个，其中最多只能有一个扩展分区。且基本分区可以马上被挂载使用但不能再分区，扩展分区必须再进行二次分区后才能挂载。扩展分区下的二次分区被称之为逻辑分区，逻辑分区数量限制视磁盘类型而定。

MBR的主分区号为1-4，逻辑分区号为从5开始累加的数字。比如设备主板上装了4块硬盘，分别为2块IDE接口硬盘，1块SCSI接口硬盘和一块SATA接口硬盘。其中2块IDE接口硬盘的分区策略为2个主分区和2个逻辑分区，SCSI分区策略为3个主分区和3个逻辑分区，SATA分区策略为4个主分区。硬盘文件和分区名称如下：

	硬盘	主分区1	主分区2	主分区3	主分区4	逻辑分区1	逻辑分区2	逻辑分区3	.....	逻辑分区n
IDE 1	/dev/hda	/dev/hda 1(p)	/dev/hda 2(p)	/dev/hda 3(e)	/	/dev/hda 5(l)	/dev/hda 6(l)	/	.....	/
IDE 2	/dev/hdb	/dev/hdb 1(p)	/dev/hdb 2(p)	/dev/hdb 3(e)	/	/dev/hdb 5(l)	/dev/hdb 6(l)	/	.....	/
SCSI	/dev/sda	/dev/sda 1(p)	/dev/sda 2(p)	/dev/sda 3(p)	/dev/sda 4(e)	/dev/sda 5(l)	/dev/sda 6(l)	/dev/sda 7(l)	.....	/
SATA	/dev/sdb	/dev/sdb 1(p)	/dev/sdb 2(p)	/dev/sdb 3(p)	/dev/sdb 4(p)	/	/	/	.....	/

其中分区名称后面的（p）代表主分区，（e）代表扩展分区，（l）代表逻辑分区。需要注意的是，如果分区策略中存在逻辑分区，则说明一定会有扩展分区。主分区数最多只能有3个，扩展分区数最多只能是1个，如果没有扩展分区则可以创建4个基本分区。

想要创建逻辑分区，则必须先将唯一的扩展分区创建出来，并且如果删除了扩展分区，那么它下面的所有逻辑分区也会被自动删除。

如果是SCSI接口硬盘则最多只能有15（其中扩展分区不能直接使用所以不计算）个分区，其中主分区最多4个，逻辑分区最多12个。IDE接口硬盘最多只能有63（其中扩展分区不能直接使用所以不计算）个分区，其中主分区最多4个，逻辑分区最多60个。

2、GPT分区

GPT意为GUID分区表，驱动器上的每个分区都有一个全局唯一的标识符（globally unique identifier，GUID）。支持的最大磁盘可达18EB，它没有主分区和逻辑分区之分，每个硬盘最多可以有128个分区，具有更强的健壮性与更大的兼容性，并且将逐步取代MBR分区方式。GPT分区的命名和MBR类似，只不过没有主分区、扩展分区和逻辑分区之分，分区号直接从1开始累加一直到128。

三、逻辑卷

LVM（逻辑卷）的产生是因为传统的分区一旦分区好后就无法在线扩充空间，也存在一些工具能实现在线扩充空间但是还是会面临数据损坏的风险；传统的分区当分区空间不足时，一般的解决办法是再创建一个更大的分区将原分区卸载然后将数据拷贝到新分区，但是在企业的生产系统往往不允许停机或者允许停机的时间很短，LVM就能很好的解决在线扩充空间的问题，而且不会对数据造成影响，LVM还能通过快照在备份的过程中保证日志文件和表空间文件在同一时间点的一致性。

在LVM中PE(Physical Extend)是卷的最小单位，默认4M大小，就像我们的数据是以页的形式存储一样，卷就是以PE的形式存储。PV(Physical Volume)是物理卷，如果要使用逻辑卷，首先第一步操作就是将物理磁盘或者物理分区格式化成PV，格式化之后PV就可以为逻辑卷提供PE。VG(Volume Group)是卷组，VG就是将很多PE组合在一起生成一个卷组，当然这里的PE是可以跨磁盘的，如果当前服务器磁盘空间不足就可以增加一个新磁盘对当前系统不会产生任何影响。

LV(Logical Volume)是逻辑卷，逻辑卷最终是给用户使用的，前面几个都是为创建逻辑卷做的准备，创建逻辑卷的大小只要不超过VG剩余空间就可以。

四、文件系统

当硬盘分区被创建完成之后，还并不能直接挂载到目录上存储文件，需要选择合适的文件系统进行格式化。常见的分区类型有FAT32、FAT16、NTFS、HP-UX等，而专供Linux使用的主流的一些分区有ext2/3/4、physical volume (LVM) 、softwareRAID、swap、vfat、xfs等。

其中：

1、ext2/3/4：是适合Linux的文件系统类型，由于ext3文件系统多了日志记录功能，因此系统恢复起来更加快速，ext4是ext3的升级，效率更加高，因此建议使用默认类型ext4类型，而不要使用ext2/3；

2、physical volume (LVM)：这是一种弹性调整文件系统大小的机制，即可以让文件系统变大或变小，而不改变原文件数据的内容，功能不错，但性能不佳。

3、softwareRAID：利用Linux系统的特性，用软件仿真出磁盘阵列功能。

4、swap：就是内存交换空间。由于swap并不会使用到目录树的挂载，因此用swap就不需要指定挂载点。

5、vfat：同时被Linux与windows所支持的文件系统类型。如果主机硬盘同时存在windows和linux两种操作系统，为了进行数据交换，可以使用该文件系统。

6、xfs：一个文件系统类型，在centos7中将被作为默认的文件系统类型，替换ext4。

五、使用fdisk操作分区

本文主要以CentOS7发行版的Linux作为实验，我们使用fdisk工具来操作分区，Fdisk 是各种 Linux 发行版本中最常用的分区工具。

```
# 查看fdisk工具帮助
[root@test ~]# fdisk -h
用法：
 fdisk [选项] <磁盘>    更改分区表
 fdisk [选项] -l <磁盘> 列出分区表
 fdisk -s <分区>        给出分区大小(块数)

选项：
 -b <大小>              扇区大小(512、1024、2048或4096)
 -c[=<模式>]            兼容模式："dos"或"nondos"(默认)
 -h                     打印此帮助文本
 -u[=<单位>]            显示单位："cylinders"(柱面)或"sectors"(扇区，默认)
 -v                     打印程序版本
 -C <数字>              指定柱面数
 -H <数字>              指定磁头数
 -S <数字>              指定每个磁道的扇区数
```

从中我们可以看出，使用 fdisk -l 命令可查看分区表信息

```
[root@test ~]# fdisk -l
```

```
磁盘 /dev/sda：21.5 GB, 21474836480 字节，41943040 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
磁盘标签类型：dos
磁盘标识符：0x0004ae9e

   设备 Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048     2099199     1048576   83  Linux
/dev/sda2         2099200    41943039    19921920   8e  Linux LVM

磁盘 /dev/sdb：1073.7 GB, 1073741824000 字节，2097152000 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节


磁盘 /dev/sdc：8589.9 GB, 8589934592000 字节，16777216000 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节


磁盘 /dev/mapper/centos_test-root：18.2 GB, 18249416704 字节，35643392 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节


磁盘 /dev/mapper/centos_test-swap：2147 MB, 2147483648字节，4194304个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
```

从中我们可以看出，有5个设备，分别为/dev/sda、/dev/sdb、/dev/sdc、/dev/mapper/cl-root、/dev/mapper/cl-swap。其中/dev/sda硬盘已经有2个分区分区为：/dev/sda1和/dev/sda2。/dev/mapper/centos_test-root和/dev/mapper/centos_test-swap两个设备是/dev/sda2分区创建的逻辑卷。这里的/dev/sdb、/dev/sdc硬盘设备并没有被分区，我们则是需要来操作这个硬盘，至于如何操作逻辑卷后面会讲到。

输入 fdisk /dev/sdb 命令，对/dev/sdb硬盘的分区表进行操作：

```
命令(输入 m 获取帮助)：m
    命令操作
       a   toggle a bootable flag 切换可引导标志
       b   edit bsd disklabel 编辑BSD磁盘标签
       c   toggle the dos compatibility flag 切换DOS兼容性标志
       d   delete a partition 删除分区
       g   create a new empty GPT partition table 创建一个新的空GPT分区表
       G   create an IRIX(SGI) partition table 创建一个ILIX（SGI）分区表
       l   list known partition types 列出已知分区类型
       m   print this menu 打印此菜单
       n   add a new partition 添加新分区
       o   create a new empty DOS partition table 创建一个新的空DOS分区表
       p   print the partition table 打印分区表
       q   quit without saving changes 不保存更改退出
       s   create a new empty Sun disklabel 创建一个新的空Sun标签
       t   change a partition's system id 更改分区的系统ID
       u   change display/entry units 更改显示/输入单元
       v   verify the partition table 验证分区表
       w   write table to disk and exit 将表写入磁盘并退出
       x   extra functionality (experts only) 额外功能（仅专家）
```

从上面的帮助信息中，可以得知一些选项的用途。这里主要注意"d"、"n"、"q"、"g"、"w"等选项。首先要明确分区格式，fdisk默认的分区格式是msdos（mbr），在此可输入"g"选项，将分区格式修改为GPT，不过在修改完保存退出之后，在输入 fdisk /dev/sdb 命令进入分区模式，会出现 WARNING: fdisk GPT support is currently new, and therefore in an experimental phase. Use at your own discretion. 信息，提示fdisk gpt分区是新的功能，目前还在实验阶段。所以如果要进行GPT分区，那么推荐使用 parted 命令，后面会介绍到。

第一步：**输入"n"选项来开始创建分区：**

```
命令(输入 m 获取帮助)：n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p):
```

可以看到交互界面打印的信息，提示需要选择一个分区类型，"p":为主分区；"e"：为扩展分区。在此我们选择"p"，创建一个基本分区：

```
Select (default p): p

分区号 (2-4，默认 2)：
```

交互界面提示需要选择一个分区号，范围为2-4。由于已经存在了一个基本分区，所以只可选择2、3、4（默认2，顺序累加）。在此我们输入2：
分区号 (2-4，默认 2)：2

起始 扇区 (2099200-314572799，默认为 2099200)：
可以看到交互界面提示序号选择起始扇区，默认为剩余未被分配的最小扇区，推荐选择默认（直接点击回车）；
将使用默认值 2099200
Last 扇区，+扇区 or +size{K,M,G} (2099200-314572799，默认为 314572799)：
交互界面提示，要输入需要分配的截止扇区，默认为未被分配的最大扇区，此处推荐默认（直接点击回车）
将使用默认值 314572799
分区 2 已设置为 Linux 类型，大小设为 149 GiB
这表示分区表已经设置成功，输入选项q表示要放弃本次分区表的修改并退出，w选项表示保存本次分区表的修改并退出，此处选择w表示将分区信息写入到磁盘，此次分区完成；

再对/dev/sdb进行编辑
回到最初操作分区表的地方，选择"d"选项，删除分区的功能：
命令(输入 m 获取帮助)：d
分区号 (1,2，默认 2)：
交互界面提示输入要删除的分区的分区号，此处选择2：
分区号 (1,2，默认 2)：2
分区 2 已删除
交互界面提示本次分区表操作成功，输入选项"w"，表示将分区信息写入到磁盘，此次删除分区完成。

再对/dev/sdb进行编辑
回到最初选择分区类型的地方，选择"e"，创建扩展分区：
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): e
分区号 (2-4，默认 2)：
交互界面提示要输入扩展分区的分区号，可选范围为2-4，此处选择2：

Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): e
分区号 (2-4，默认 2)：2
起始 扇区 (2099200-314572799，默认为 2099200)：

交互界面提示输入要分配给扩展分区的起始扇区，此处选择默认：
将使用默认值 2099200
Last 扇区，+扇区 or +size{K,M,G} (2099200-314572799，默认为 314572799)：

交互界面提示输入要分配给扩展分区的截止扇区，此处选择默认：
将使用默认值 314572799
分区 2 已设置为 Extended 类型，大小设为 149 GiB

交互界面提示本次对分区表的操作已完成，输入"w"选项，保存本次对分区表的操作；当再次创建分区的时候，交互界面就会将扩展分区的选项"e"替换成为逻辑分区的选项"l"：
Partition type:
   p   primary (1 primary, 1 extended, 2 free)
   l   logical (numbered from 5)
Select (default p):
之后再要创建逻辑分区和之前创建分区的步骤一样。

Demo1：实验使用的分区
```
/dev/sdb1 p ext4 # 指定扇区
/dev/sdb2 p xfs  # + 容量大小
/dev/sdb3 e
/dev/sda5 l ext4
```

六、分区格式化
```
[root@test ~]# mkfs.ext4 /dev/sdb1  # 把该设备格式化成ext4文件系统
[root@test ~]# mkfs.ext3 /dev/sdb1  # 把该设备格式化成ext3文件系统
[root@test ~]# mkfs.xfs  /dev/sdb1  # 把该设备格式化成xfs文件系统
[root@test ~]# mkfs.vfat /dev/sdb1  # 把该设备格式化成vfat文件系统
```

多文件系统格式对比：
ext2文件系统的确高效稳定。但是，随着Linux系统在关键业务中的应用，Linux文件系统的弱点也渐渐显露出来了:其中系统缺省使用的ext2文件系统是非日志文件系统。

Ext3文件系统是直接从Ext2文件系统发展而来，目前ext3文件系统已经非常稳定可靠。它完全兼容ext2文件系统。用户可以平滑地过渡到一个日志功能健全的文件系统中来。这实际上了也是ext3日志文件系统初始设计的初衷。
Ext3日志文件系统的特点
1、高可用性
系统使用了ext3文件系统后，即使在非正常关机后，系统也不需要检查文件系统。宕机发生后，恢复ext3文件系统的时间只要数十秒钟。

2、数据的完整性:
ext3文件系统能够极大地提高文件系统的完整性，避免了意外宕机对文件系统的破坏。

Ext4 是 Ext3 的改进版，修改了 Ext3 中部分重要的数据结构，而不仅仅像 Ext3 对 Ext2 那样，只是增加了一个日志功能而已。Ext4 可以提供更佳的性能和可靠性，还有更为丰富的功能：

1. 与 Ext3 兼容。 执行若干条命令，就能从 Ext3 在线迁移到 Ext4，而无须重新格式化磁盘或重新安装系统。原有 Ext3 数据结构照样保留，Ext4 作用于新数据，当然，整个文件系统因此也就获得了 Ext4 所支持的更大容量。
2. 更大的文件系统和更大的文件。 较之 Ext3 目前所支持的最大 16TB 文件系统和最大 2TB 文件，Ext4 分别支持 1EB（1,048,576TB， 1EB=1024PB，1PB=1024TB）的文件系统，以及 16TB 的文件。
3. 无限数量的子目录。 Ext3 目前只支持 32,000 个子目录，而 Ext4 支持无限数量的子目录。
4. Extents。 Ext3 采用间接块映射，当操作大文件时，效率极其低下。比如一个 100MB 大小的文件，在 Ext3 中要建立 25,600 个数据块（每个数据块大小为 4KB）的映射表。而 Ext4 引入了现代文件系统中流行的 extents 概念，每个 extent 为一组连续的数据块，上述文件则表示为“该文件数据保存在接下来的 25,600 个数据块中”，提高了不少效率。
5. 多块分配。 当写入数据到 Ext3 文件系统中时，Ext3 的数据块分配器每次只能分配一个 4KB 的块，写一个 100MB 文件就要调用 25,600 次数据块分配器，而 Ext4 的多块分配器“multiblock allocator”（mballoc） 支持一次调用分配多个数据块。
6. 延迟分配。 Ext3 的数据块分配策略是尽快分配，而 Ext4 和其它现代文件操作系统的策略是尽可能地延迟分配，直到文件在 cache 中写完才开始分配数据块并写入磁盘，这样就能优化整个文件的数据块分配，与前两种特性搭配起来可以显著提升性能。
7. 快速 fsck。 以前执行 fsck 第一步就会很慢，因为它要检查所有的 inode，现在 Ext4 给每个组的 inode 表中都添加了一份未使用 inode 的列表，今后 fsck Ext4 文件系统就可以跳过它们而只去检查那些在用的 inode 了。
8. 日志校验。 日志是最常用的部分，也极易导致磁盘硬件故障，而从损坏的日志中恢复数据会导致更多的数据损坏。Ext4 的日志校验功能可以很方便地判断日志数据是否损坏，而且它将 Ext3 的两阶段日志机制合并成一个阶段，在增加安全性的同时提高了性能。
9. “无日志”（No Journaling）模式。 日志总归有一些开销，Ext4 允许关闭日志，以便某些有特殊需求的用户可以借此提升性能。
10. 在线碎片整理。 尽管延迟分配、多块分配和 extents 能有效减少文件系统碎片，但碎片还是不可避免会产生。Ext4 支持在线碎片整理，并将提供 e4defrag 工具进行个别文件或整个文件系统的碎片整理。
11. inode 相关特性。 Ext4 支持更大的 inode，较之 Ext3 默认的 inode 大小 128 字节，Ext4 为了在 inode 中容纳更多的扩展属性（如纳秒时间戳或 inode 版本），默认 inode 大小为 256 字节。Ext4 还支持快速扩展属性（fast extended attributes）和 inode 保留（inodes reservation）。
12. 持久预分配（Persistent preallocation）。 P2P 软件为了保证下载文件有足够的空间存放，常常会预先创建一个与所下载文件大小相同的空文件，以免未来的数小时或数天之内磁盘空间不足导致下载失败。 Ext4 在文件系统层面实现了持久预分配并提供相应的 API（libc 中的 posix_fallocate()），比应用软件自己实现更有效率。
13. 默认启用 barrier。 磁盘上配有内部缓存，以便重新调整批量数据的写操作顺序，优化写入性能，因此文件系统必须在日志数据写入磁盘之后才能写 commit 记录，若 commit 记录写入在先，而日志有可能损坏，那么就会影响数据完整性。Ext4 默认启用 barrier，只有当 barrier 之前的数据全部写入磁盘，才能写 barrier 之后的数据。（可通过 "mount -o barrier=0" 命令禁用该特性。）

七、挂载分区

①：分区格式化完成之后则可以将分区挂载到某一个目录下面，正式开始使用改分区，我们在系统中创建一个用户挂载分区的目录：

```
[root@localhost ~]# mkdir /data
```

将分区挂载到目录上：

```
[root@localhost ~]# mount /dev/sda2 /data/
```

如果想要卸载挂载点：

```
[root@localnat201 ~]# umount /dev/sda2
```

②：设置开机自动挂载分区到挂载点，编辑 vim /etc/fstab ：

```
#
# /etc/fstab
# Created by anaconda on Sun Jun 25 07:16:25 2017
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=eb697457-a097-4263-8bbf-a75aa632d27c /                       ext4    defaults        1 1
/dev/sda2                                 /data                   xfs     defaults        0 0
/dev/sdb1 /root/x1/ ext2 defaults 0 0
/dev/sdb2 /root/x2/ ext3 defaults 0 0
/dev/sdb4 /root/x4/ ext4 ro,suid,dev,exec,auto,nouser,async 0 0
[root@test ~]# lsblk -f /dev/sdb # 查看分区设备的UUID号
```

字段

1. 设备文件，一般为设备的路径+设备名称，也可以写唯一识别码（UUID）
   a. [root@test ~]# lsblk -f /dev/sdb # 查看设备的UUID码
2. 挂载目录，指定要挂载到的目录，需要在挂载前创建好
3. 格式类型，指定文件系统的类型，比如ext3、ext4、xfs、swap等
4. 权限选项，一般设置为defaults，默认权限：rw,suid,dev,exec,auto,nouser,async
5. 是否备份，若1则开机后使用dunmp进行磁盘备份，为0则不备份
6. 若为1则开机后自动进行磁盘自检，为0则不自检

权限选项的详细参数：

auto - 在启动时或键入了 mount -a 命令时自动挂载。
noauto - 只在手动执行命令的情况下被挂载。
exec - 允许执行此分区的二进制文件。
noexec - 不允许执行此文件系统上的二进制文件。
ro - 以只读模式挂载文件系统。
rw - 以读写模式挂载文件系统。
user - 允许任意用户挂载此文件系统，若无显示定义，隐含启用 noexec, nosuid, nodev 参数。
users - 允许所有 users 组中的用户挂载文件系统.
nouser - 只能被 root 挂载。

owner - 允许设备所有者挂载.
sync - I/O 同步进行。
async - I/O 异步进行。
dev - 解析文件系统上的块特殊设备。
nodev - 不解析文件系统上的块特殊设备。
suid - 允许 suid 操作和设定 sgid 位。这一参数通常用于一些特殊任务，使一般用户运行程序时临时提升权限。
nosuid - 禁止 suid 操作和设定 sgid 位。
noatime - 不更新文件系统上 inode 访问记录，可以提升性能(参见 atime 参数)。
nodiratime - 不更新文件系统上的目录 inode 访问记录，可以提升性能(参见 atime 参数)。
relatime - 实时更新 inode access 记录。只有在记录中的访问时间早于当前访问才会被更新。（与 noatime 相似，但不会打断如 mutt 或其它程序探测文件在上次访问后是否被修改的进程。），可以提升性能(参见 atime 参数)。
flush - vfat 的选项，更频繁的刷新数据，复制对话框或进度条在全部数据都写入后才消失。
defaults - 使用文件系统的默认挂载参数，例如 ext4 的默认参数为:rw, suid, dev, exec, auto, nouser, async.

```
[root@localhost ~]# mount  -a # 手动执行 /etc/fstab 中定义的所有自动挂载

# 重新挂载设备
[root@test x4]# mount -o remount,rw /dev/sdb4
```

八、swap分区

创建swap交换分区添加和自动挂载

```
# 创建分区并激活
[root@test ~]# mkswap /dev/sdb1  # 将分区转换为swap格式
[root@test ~]# swapon /dev/sdb1  # 激活swap分区

# 确定交换分区
[root@test ~]# swapon -s
[root@test ~]# free -h

# 禁用交换分区
[root@test ~]# swapoff /dev/sdb1 # 禁用交换分区
```

vim /etc/fstab 写入自动挂载

```
[root@test ~]# vim /etc/fstab
[root@test ~]# mount -a  # 该命令对swap分区，不生效，若想成功执行，必须reboot

# 用dd命令创建一个文件然后步骤同上（虚拟内存文件）
[root@test ~]# dd if=/dev/zero of=test bs=1M count=1024
```

bs=bytes：同时设置读入/输出的块大小为bytes个字节。
count=blocks：仅拷贝blocks个块，块大小等于bs指定的字节数。

九、parted分区

如果你的硬盘大于2TB则必须要使用parted来创建GPT格式的分区。

```
[root@test ~]# parted /dev/sdc
[root@test ~]# parted -l #查看分区表信息
错误: /dev/sdc: unrecognised disk label
Model: VMware, VMware Virtual S (scsi)
Disk /dev/sdc: 8590GB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:
```

从中可看出与上面 fdisk -l 命令返回的差不多的信息。总共有5个设备：/dev/sda、/dev/sdb、/dev/sdc为物理设备，/dev/mapper/cl-swap 和/dev/mapper/cl-root为逻辑卷创建的设备。可以看到/dev/sdb还没有分区，并且还看到上面有一个错误信息 错误: /dev/sdb: unrecognised disk label 。这是由于该磁盘设备没有设置上标签（label）所以会有错误，只需要设置了标签就可以了。

```
[root@test ~]# parted /dev/sdc
(parted) mklabel gpt  # 使用 mklabel gpt 或者 mktable gpt 命令格式化分区类型和设置标签，此处可选择modos（mbr）和gpt类型，如果修改的分区标签类型，则分区所有数据将会丢失；
(parted) print        # 接下来可输入 print 选项，打印分区信息
```

由此可以看出分区已经是GPT分区格式；加下来需要创建分区，创建分区需要使用 mkpart 命令，在此我们可以输入 help mkpart 命令查看帮助信息：

```
(parted) print
Model: VMware, VMware Virtual S (scsi)
Disk /dev/sdc: 8590GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start  End  Size  File system  Name  标志
```

创建分区，创建分区需要使用 mkpart 命令，在此我们可以输入 help mkpart 命令查看帮助信息：

```
(parted) help mkpart
```

(parted) mkpart xfs 0 100% ,用 mkpart xfs 0 100% 命令创建分区，xfs是文件系统类型（这里只是做说明或者说是分区的名称，分区完成之后是需要使用 mkfs.xfs 命令进行真正的格式化的，否则不能挂载）， 0是磁盘的起始位置，100%是磁盘的结束位置：
创建的过程中，我们会看到有警告信息 The resulting partition is not properly aligned for best performance. ，说分区没有正确对齐，会影响最

佳新能。这里说的是磁盘的位置没有给一个合适的值。其实在使用fdisk分区的时候，会有默认的起始和结束扇区，所以如果不是很确定这个值，那么可以先试用fdisk命令进入分区模式，看一下默认的起始扇区和结束扇区是多少。

```
(parted) q
信息: You may need to update /etc/fstab.
(parted) mkpart xfs 2048s  10% # 使用s说明是扇区为单位

[root@test ~]# partprobe  #更新分区信息
[root@test ~]# partprobe /dev/sdc
```