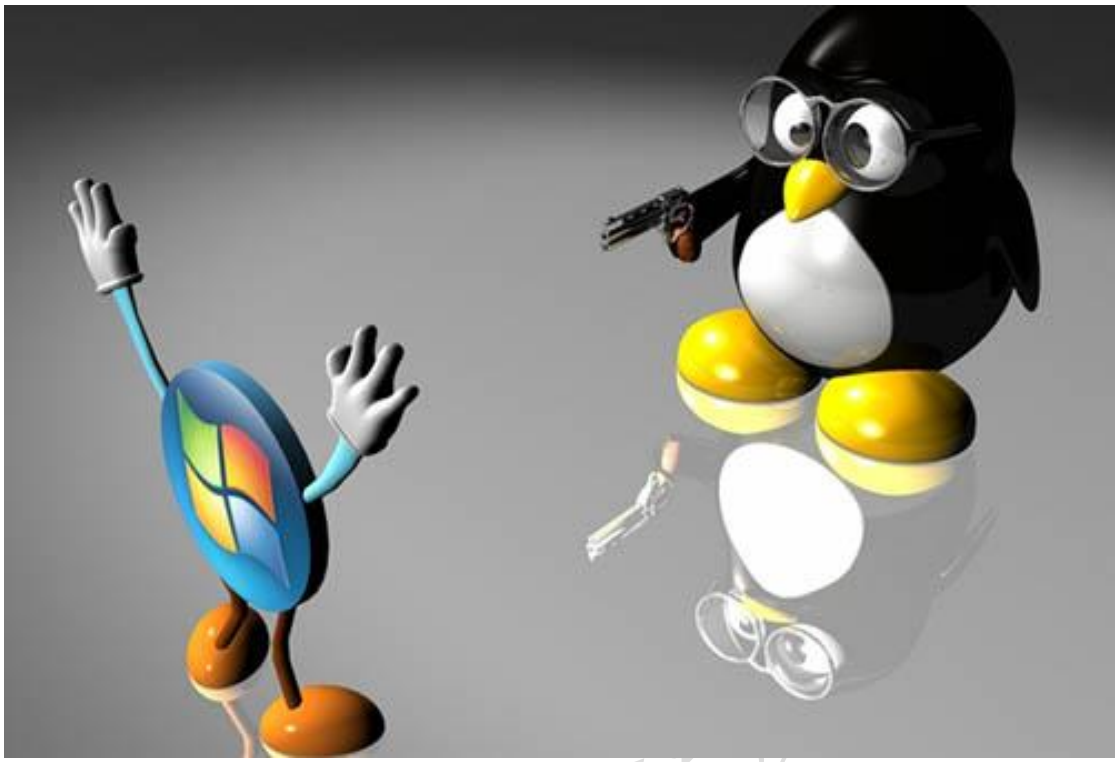


linux 文件管理（操作篇）

作者：七王爷

QQ:575119655 我

文件系统及 Shell 的基本概念

1．文件系统的含义

文件系统是用来管理和组织保存在磁盘驱动器上数据的系统软件

2．Linux 的文件系统

Linux 系统采用虚拟文件系统技术（VFS)、结构采用倒立树型、VFS 使 Linux 支持以下文件系统：

EXT2： 二次扩展

EXT3： 三次扩展

SWAP： 交换文件系统

FAT、FAT32

NTFS (默认不支持，需要特定的模块)

VFAT： 虚拟 FAT

SYSV： Unix 的文件系统

HPFS： OS/2 的文件系统

ISO9660： 光盘文件系统

NFS： 网络文件系统

3．Linux 中的文件分类

（1）普通文件

文本文件：采用 ASCII 编码方式，可编辑,可修改

二进制：不可查看，不可修改

（2）目录文件

存放的内容是目录中的文件名和子目录名

（3）设备文件（/dev)

用于用户访问物理设备所用，分为块设备和字符设备文件

（4）链接文件

软链接文件：目标文件和链接文件可以跨越索引点，相当于文件的快捷方式

删除原文件，则符号链接文件失去意义

删除符号链接文件，不影响原文件

硬链接文件：链接同一索引点中的文件，相当文件的副本

两个文件指向同一存储区，内容、长度相同

删除一个文件不影响，其它文件

（5）管道文件

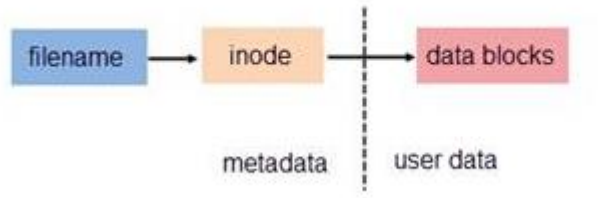
FIFO 缓存队列

qq:575119655 我爱七王爷

硬链接与软链接的联系与区别

文件都有文件名与数据，这在 Linux 上被分成两个部分：用户数据（user data）与元数据（metadata）。用户数据，即文件数据块（data block），数据块是记录文件真实内容的地方；而元数据则是文件的附加属性，如文件大小、创建时间、所有者等信息。在 Linux 中，元数据中的 inode 号（inode 是文件元数据的一部分但其并不包含文件名，inode 号即索引节点号）才是文件的唯一标识而非文件名。文件名仅是为了方便人们的记忆和使用，系统或程序通过 inode 号寻找正确的文件数据块

通过文件名打开文件



在 Linux 系统中查看 inode 号可使用命令 stat 或 ls -i（若是 AIX 系统，则使用命令 istat）

两种链接：硬链接（hard link）与软链接（又称符号链接，即 soft link 或 symbolic link）。链接为 Linux 系统解决了文件的共享使用，还带来了隐藏文件路径、增加权限安全及节省存储等好处。若一个 inode 号对应多个文件名，则称这些文件为硬链接。换言之，硬链接就是同一个文件使用了多个别名（link 就是 file 的一个别名，他们有共同的 inode）。硬链接可由命令 link 或 ln 创建

命令格式

```
link oldfile newfile
ln oldfile newfile
```

案例：

```
[root@node1]# ls -il
总计 0
1491138 -rw-r–r– 1 root root 48 07-14 14:17 file1
1491139 -rw-r–r– 2 root root 0 07-14 14:17 file2
```

```
[root@node1]# ln file2 file2hand
[root@node1]# ls -il
总计 0
1491138 -rw-r--r-- 1 root root 48 07-14 14:17 file1
1491139 -rw-r--r-- 2 root root 0 07-14 14:17 file2
1491139 -rw-r--r-- 2 root root 0 07-14 14:17 file2hand
```

由于硬链接是有着相同 inode 号仅文件名不同的文件，因此硬链接存在以下几点特性：

- 文件有相同的 inode 及 data block；
- 只能对已存在的文件进行创建；
- 不能交叉文件系统进行硬链接的创建；
- 不能对目录进行创建，只可对文件创建；
- 删除一个硬链接文件并不影响其他有相同 inode 号的文件。

尽管硬链接节省空间，也是 Linux 系统整合文件系统的传统方式，但是存在一下不足之处：

（1）不可以在不同文件系统的文件间建立链接

（2）只有超级用户才可以为目录创建硬链接。

软链接克服了硬链接的不足，没有任何文件系统的限制，任何用户可以创建指向目录的符号链接。因而现在更为广泛使用，它具有更大的灵活性，甚至可以跨越不同机器、不同网络对文件进行链接。

案例：

```
[root@ node1]# ln -s file1 file1soft
[root@ node1]# ls -il
总计 0
1491138 -rw-r--r-- 1 root root 48 07-14 14:17 file1
1491140 lrwxrwxrwx 1 root root 5 07-14 14:24 file1soft -> file1
1491139 -rw-r--r-- 2 root root 0 07-14 14:17 file2
1491139 -rw-r--r-- 2 root root 0 07-14 14:17 file2hand
```

软链接与硬链接，区别不仅仅是在概念上，在实现上也是不同的。区别：硬链接原文件＆链接文件公用一个 inode 号，说明他们是同一个文件，而软链接原文件＆链接文件拥有不同的 inode 号，表明他们是两个不同的文件；在文件属性上软链接明确写出了是链接文件，而硬链接没有写出来，因为在本质上硬链接文件和原文件是完全平等关系；链接数目是不一样的，软链接的链接数目不会增加；文件大小是不一样的，硬链接文件显示的大小是跟原文

件是一样的，这用强调，因为是等同的，而这里软链接显示的大小与原文件就不同了，file1 大小是 48B，而 file1soft 是 5B，这里面的 5 实际上就是“file1”的大小。

总之，建立软链接就是建立了一个新文件。当访问链接文件时，系统就会发现他是个链接文件，它读取链接文件找到真正要访问的文件。

在不同系统之间建立软链接、对目录建立链接，这里就不举例了，读者可以自己去尝试，我也是在不断实践中学习的。

当然软链接也有硬链接没有的缺点，因为链接文件包含有原文件的路径信息，所以当原文件从一个目录下移到其他目录中，再访问链接文件，系统就找不到了，而硬链接就没有这个缺陷，你想怎么移就怎么移；还有它要系统分配额外的空间用于建立新的索引节点和保存原文件的路径。

删除链接

有创建就有删除

rm -rf symbolic_name 注意不是 rm -rf symbolic_name/

案例

```
[root@ node1]# ls -il
总计 0
1491138 -rw-r–r– 1 root root 0 07-14 14:17 file1
1491140 lrwxrwxrwx 1 root root 5 07-14 14:24 file1soft -> file1
1491139 -rw-r–r– 2 root root 0 07-14 14:17 file2
1491139 -rw-r–r– 2 root root 0 07-14 14:17 file2hand
[root@ node1]# rm -rf file1soft
[root@ node1]# ls -il
总计 0
1491138 -rw-r–r– 1 root root 0 07-14 14:17 file1
1491139 -rw-r–r– 2 root root 0 07-14 14:17 file2
1491139 -rw-r–r– 2 root root 0 07-14 14:17 file2hand
```

linux 软连接和硬链接的区别:

（1）软连接可以 跨文件系统 ，硬连接不可以 。

实践的方法就是用共享文件把 windows 下的 aa.txt 文本文档连接到 linux 下/root 目录

下 bb,cc . ln -s aa.txt

/root/bb 连接成功 。ln aa.txt /root/bb 失败 。

（2）关于 I 节点的问题 。硬连接不管有多少个，都指向的是同一个 I 节点，会把 结点连接数增加 ，只要结点的连接数不是 0，文件就一直存在 ，不管你删除的是
源文件还是 连接的文件 。只要有一个存在 ，文件就 存在 （其实也不分什么 源文件连接文件的 ，因为他们指向都是同一个 I 节点）。 当你修改源文件或者连接文件
任何一个的时候 ，其他的 文件都会做同步的修改 。软链接不直接使用 i 节点号作为文件指针,而是使用文件路径名作为指针。所以 删除连接文件 对源文件无影响，但
是 删除 源文件，连接文件就会找不到要指向的文件 。软链接有自己的 inode,并在磁盘上有一小片空间存放路径名.
（3）软连接可以对一个不存在的文件名进行连接 。
（4）软连接可以对目录进行连接。
备注：I 节点 :它是 UNIX 内部用于描述文件特性的数据结构.我们通常称 I 节点为文件索引结点(信息结点).i 节点 含有关于文件的大部分的重要信 息,包括文件数据块在
磁盘上的地址.每一个 I 节点有它自己的标志号,我们称为文件顺序号.I 节点包含的信息 1.文件类型 2.文件属主关系 3.文件的访问权限 4.文件的时间截.

理解 inode

inode 是什么

理解 inode，要从文件储存说起。
文件储存在硬盘上，硬盘的最小存储单位叫做"扇区" (Sector)。每个扇区储存 512 字节 (相当于 0.5KB)。
操作系统读取硬盘的时候，不会一个个扇区地读取，这样效率太低，而是一次性连续读取多个扇区，即一次性读取一个"块" (block)。这种由多个扇区组成的"块"，是文件存取的最小单位。"块"的大小，最常见的是 4KB，即连续八个 sector 组成一个 block。
文件数据都储存在"块"中，那么很显然，我们还必须找到一个地方储存文件的元信息，比如文件的创建者、文件的创建日期、文件的大小等等。这种储存文件元信息的区域就叫做 inode，中文译名为"索引节点"。
每一个文件都有对应的 inode，里面包含了与该文件有关的一些信息。

inode 的内容

inode 包含文件的元信息，具体来说有以下内容：

* 文件的字节数
* 文件拥有者的 User ID
* 文件的 Group ID
* 文件的读、写、执行权限
* 文件的时间戳，共有三个：ctime 指 inode 上一次变动的时间，mtime 指文件内容上一次变动的时间，atime 指文件上一次打开的时间。
* 链接数，即有多少文件名指向这个 inode
* 文件数据 block 的位置

可以用 stat 命令，查看某个文件的 inode 信息：

```
stat example.txt
```

inode 的大小

inode 也会消耗硬盘空间，所以硬盘格式化的时候，操作系统自动将硬盘分成两个区域。一个是数据区，存放文件数据；另一个是 inode 区（inode table），存放 inode 所包含的信息。

每个 inode 节点的大小，一般是 128 字节或 256 字节。inode 节点的总数，在格式化时就给定，一般是每 1KB 或每 2KB 就设置一个 inode。假定在一块 1GB 的硬盘中，每个 inode 节点的大小为 128 字节，每 1KB 就设置一个 inode，那么 inode table 的大小就会达到 128MB，占整块硬盘的 12.8%。

查看每个硬盘分区的 inode 总数和已经使用的数量，可以使用 df 命令。

```
df -i
```

查看每个 inode 节点的大小，可以用如下命令：

```
sudo dumpe2fs -h /dev/hda | grep "Inode size"
```

由于每个文件都必须有一个 inode，因此有可能发生 inode 已经用光，但是硬盘还未存满的情况。这时，就无法在硬盘上创建新文件。

inode 号码

每个 inode 都有一个号码，操作系统用 inode 号码来识别不同的文件。

这里值得重复一遍，Unix/linux 系统内部不使用文件名，而使用 inode 号码来识别文件。对于系统来说，文件名只是 inode 号码便于识别的别称或者绰号。

表面上，用户通过文件名，打开文件。实际上，系统内部这个过程分成三步：首先，系统找到这个文件名对应的 inode 号码；其次，通过 inode 号码，获取 inode 信息；最后，根据

inode 信息，找到文件数据所在的 block，读出数据。

使用 ls -i 命令，可以看到文件名对应的 inode 号码：

```
ls -i example.txt
```

目录文件

Unix/Linux 系统中，目录（directory）也是一种文件。打开目录，实际上就是打开目录文件。

目录文件的结构非常简单，就是一系列目录项（dirent）的列表。每个目录项，由两部分组成：所包含文件的文件名，以及该文件名对应的 inode 号码。

ls 命令只列出目录文件中的所有文件名：

```
ls /etc
```

ls -i 命令列出整个目录文件，即文件名和 inode 号码：

```
ls -i /etc
```

如果要查看文件的详细信息，就必须根据 inode 号码，访问 inode 节点，读取信息。ls -l 命令列出文件的详细信息。

```
ls -l /etc
```

理解了上面这些知识，就能理解目录的权限。目录文件的读权限（r）和写权限（w），都是针对目录文件本身（即不同用户能以什么权限访问操作对该目录文件，例如这里不同用户对 tmp 目录文件(d 可以查出 tmp 是目录文件，d 表示 directory，即目录)分别为 rwxr-xr-x，第一组的三个字符，即 rwx，表示文件拥有者用户的对该文件的读写权限，第二组的三个字符，即 r-x，表示文件拥有者用户所在的用户组里的其他用户对该文件的读写权限，第三组的三个字符，即 r-x，表示文件拥有者用户所在的用户组以外的用户对该文件的读写权限。一个某个用户下运行的进程访问操作该目录文件只能以该用户所具有的对该目录文件的权限进行操作)。由于目录文件内只有文件名和 inode 号码，所以如果只有读权限，只能获取文件名，无法获取其他信息，因为其他信息都储存在 inode 节点中，而读取 inode 节点内的信息需要目录文件的执行权限（x)。

从硬链接和软链接看 iNode

其中每个 dentry 都有一个唯一的 inode，而每个 inode 则可能有多个 dentry，这种情况是由 ln 硬链接产生的。

dentry 的中文名称是目录项，是 Linux 文件系统中某个索引节点(inode)的链接。这个索引节点可以是文件的，也可以是目录的。

硬链接：其实就是同一个文件具有多个别名，具有相同 inode，而 dentry 不同。

1. 文件具有相同的 inode 和 data block；
2. 只能对已存在的文件进行创建；
3. 不同交叉文件系统进行硬链接的创建
4. 不能对目录进行创建，只能对文件创建硬链接
5. 删除一个硬链接并不影响其他具有相同 inode 号的文件；

软链接：软链接具有自己的 inode，即具有自己的文件，只是这个文件中存放的内容是另一个文件的路径名。因此软链接具有自己的 inode 号以及用户数据块。

1. 软链接有自己的文件属性及权限等；
2. 软链接可以对不存在的文件或目录创建；
3. 软链接可以交叉文件系统；
4. 软链接可以对文件或目录创建；
5. 创建软链接时，链接计数 i_nlink 不会增加；
6. 删除软链接不会影响被指向的文件，但若指向的原文件被删除，则成死链接，但重新创建指向 的路径即可恢复为正常的软链接，只是源文件的内容可能变了。

Qq:575119655 我爱七王爷

文件分配方式是索引分配时的文件系统结构

一个文件系统里的文件分为目录文件和普通文件这两类。

如果文件分配方式是索引分配的话，则有索引节点这个概念的出现。

inode 也会消耗硬盘空间，所以硬盘格式化的时候，操作系统自动将硬盘分成两个区域。一个是数据区，存放文件数据；另一个是 inode 区（inode table），存放 inode 所包含的信息。

每个 inode 节点的大小，一般是 128 字节或 256 字节。inode 节点的总数，在格式化时就给定，一般是每 1KB 或每 2KB 就设置一个 inode。假定在一块 1GB 的硬盘中，每个 inode 节点的大小为 128 字节，每 1KB 就设置一个 inode，那么 inode table 的大小就会达到 128MB，占整块硬盘的 12.8%。

查看每个硬盘分区的 inode 总数和已经使用的数量，可以使用 df 命令：df -i

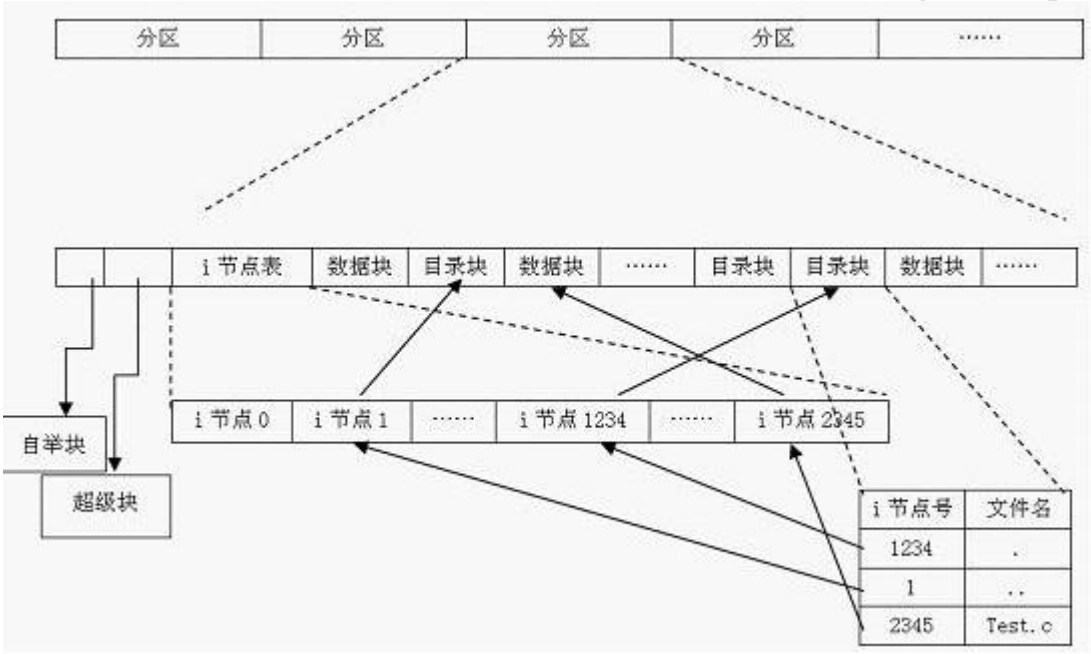
分区

分区结构

分区 (partition) 在被 Linux 的文件系统 (比如 ext2) 格式化的时候，会分成 inode table 和 block table 两部分，且大小都是固定的。该分区的所有 inode 都在 inode table 里，所有 block 都在 block table 里。

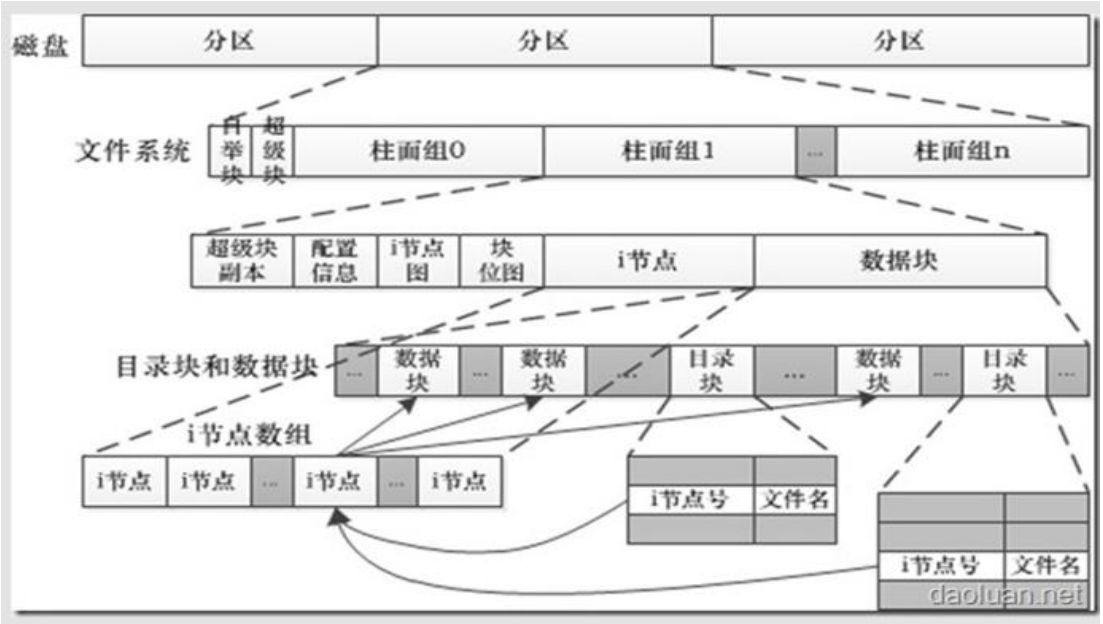
磁盘与文件系统

文件、目录、目录项、索引节点、超级块



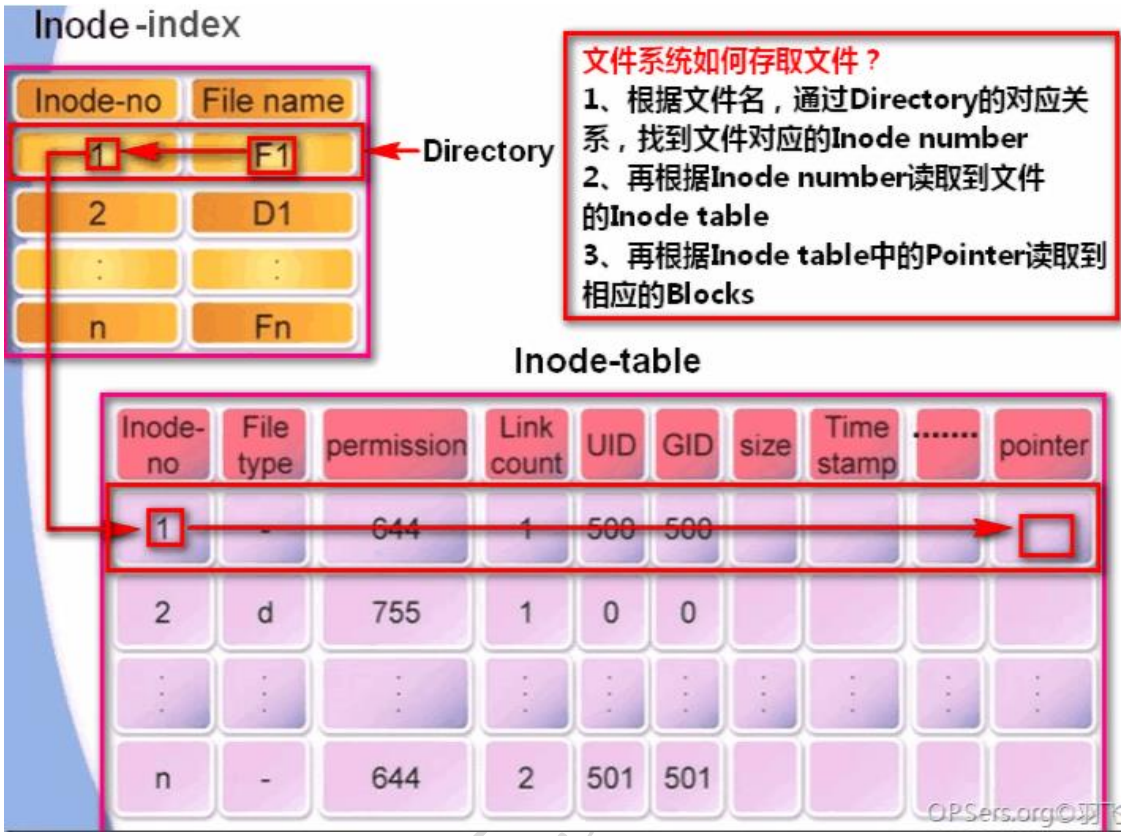
目录块里存放的是一个个的 FCB（文件控制块，一个一般 128 字节)【FCB 就是目录文件存放的业务数据】,而数据块里存放的是普通文件的业务数据。普通文件由目录块里的一个 FCB 加上多个数据块组成，而目录文件由目录块里的一个 FCB 加上多个其他多个目录块组成。一个索引节点只能被一个文件（无论是目录文件，还是普通文件）所用，不能同时被其他文件所用。一个目录块里只能存放位于目录树里处于同级的文件（无论是目录文件，还是普通文件)，所以一个根目录文件的 FCB 所在的目录块只能存放根目录文件的 FCB，与根目录文件同级的只有根目录文件自己。一个文件的 FCB 指向他的索引节点，他的索引节点指向该文件所拥有的块（如果该文件是目录文件，则该文件所拥有的块就是目录块；如果该文件是普通文件，则该文件所拥有的块就是数据块；)

换个说法



Superblock 是文件系统最基本的元数据，它定义了文件系统的类似、大小、状态，和其他元数据结构的信息（元数据的元数据）。Superblock 对于文件系统来说是非常关键的，因此对于每个文件系统它都冗余存储了多份。Superblock 对于文件系统来说是一个非常“高等级”的元数据结构。例如，如果 /var 分区的 Superblock 损坏了，那么 /var 分区将无法挂载。在这时候，一般会执行 fsck 来自动选择一份 Superblock 备份来替换损坏的 Superblock，并尝试修复文件系统。主 Superblock 存储在分区的 block 0 或者 block 1 中，而 Superblock 的备份则分散存储在文件系统的多组 block 中。当需要手工恢复时，我们可以使用 dumpe2fs /dev/sda1 | grep -i superblock 来查看 sda1 分区的 superblock 备份有哪一份是可用的。我们假设 dumpe2fs 输出了这样一行：Backup superblock at 163840, Group descriptors at 163841-163841 ，通过这条信息，我们就可以尝试使用这个 superblock 备份：/sbin/fsck.ext3 -b 163840 -B 1024 /dev/sda1。请注意，这里我们假设 block 的大小为 1024 字节

EXT3 文件系统结构图



文件系统如何存取文件的

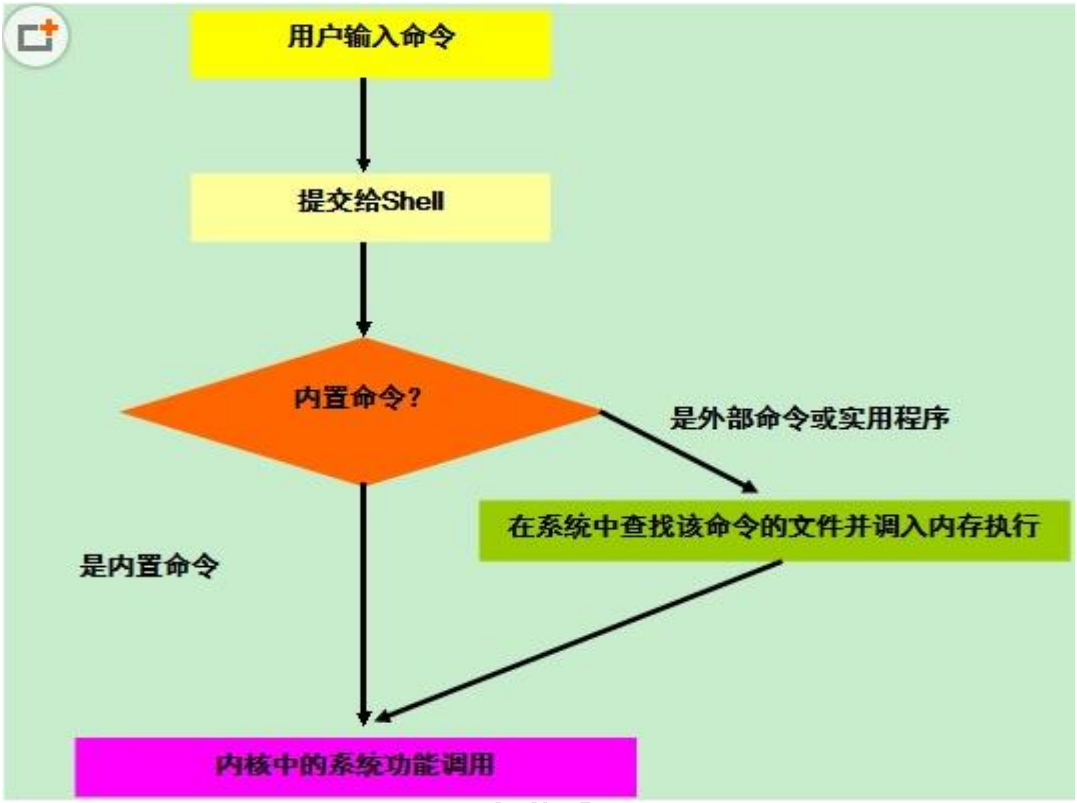
1、根据文件名，通过 Directory 里的对应关系，找到文件对应的 Inode number

2、再根据 Inode number 读取到文件的 Inode table

3、再根据 Inode table 中的 Pointer 读取到相应的 Blocks

QQ:575119653

shell 命令



关于文件的基本操作

必须记住：当创建一个文件的时候，系统保存了有关该文件的全部信息，包括：

- " 文件的位置。
- " 文件类型。
- " 文件长度。
- " 哪位用户拥有该文件，哪些用户可以访问该文件。
- " i 节点。
- " 文件的修改时间。
- " 文件的权限位。

QQ:575119653

linux 文件属性

```
-rw-rw-rw- 1 root root 42304  Sep 4 18:22  test.txt
```

- 第一列：文件权限；
- 第二列：文件连接数；
- 第三列：文件所有者；
- 第四列：文件所属用户组；
- 第五列：文件大小；
- 第六列；文件最后修改时间
- 第七列：文件名；

第一个字符意义：

·[d]:目录。

·[-]:文件。

·[l]:连接文件（linkfile）。

·[b]:设备文件里的可供储存的接口设备。

·[c]:设备文件里的串行端口设备。

接下来的字符三个为一组，且均为“rwx”的组合。[r]代表可读权限[w]可写权限[x]可执行权限。这三个权限的位置不会更改如果没有权限则为[-]。

第一组：文件所有者权限；

第二组：通过友用户组权限；

第三组：其他非本用户组权限；

- **如何改变文件属性和权限**

chgrp：改变所属用户组；

例：`chgrp users install.log`

chown：改变文件所有者；

例：chown bin install.log

chmod：改变文件权限；

- **数字改变文件权限：**

Linux 文件的基本权限有九个分别是 owner、group、others 三种身份各有自己的 read、write、execute 权限，我们可以用数字代表各个权限：

qq.576119655 我爱七王爷

r:4

w:2

x:1

每种身份各自的权限分数累加；

例：-rwxrwxrwx 分数为 777.

如果要启用 test.txt 文件的所有权限：chmod 777 test.txt

符号类型改变权限：

命令	身份	操作	权限	
chmod	u	+(加入)	r	文件或目录
	g	-(除去)	w	
	o	=(设置)	x	
	a			

如果要将 test.txt 设置为”-rwxr-xr-x”:chmod u=rwx,go=rx test.txt

如果要给所有人加入权限 x 则为：chmod a+x test.txt

QQ:575119655 我爱

实战

1．pwd

功能：显示当前目录

实例：

?# pwd

2．cd

功能：改变当前目录

实例：

?# cd /usr/sbin （进入/usr/sbin）

cd ../u1 （进入父目录下面的 u1 目录）

cd .. (返回父目录)

cd (进入用户主目录)

cd ~ （进入用户主目录）

3．mkdir

功能：建立目录

格式：mkdir [参数] <目录名>

参数：-m 属性值：指定目录的属性（r、w、x 或 4、2、1)

-p ：循环建立目录

实例：

mkdir d1 (在当前目录下建立 d1 子目录)

mkdir /d1/d2 /d1/d3 /d1/d4（同时在/d1 目录中建立 d2,d3,d4 子目录)

mkdir –p /d1/d2 （在/d1 目录中建立 d2 子目录，先建 d1）

4．rmdir

功能：删除空目录

格式：rmdir [参数] <目录名>

参数：-p：循环删除

实例：

rmdir a1 (删除当前目录下的 a1 子目录)

rmdir /etc/a2 (删除/etc 下 a2 子目录)

rmdir -p /a1/a2

5．ls

功能：显示目录中的文件及子目录的名称

qq·575119655 我爱七王爷

格式：ls [参数] [文件名]

参数：-a： 显示所有文件（包括隐含文件，以“.”开头的文件为隐含文件）

-l： 以长格式显示文件名及目录名（显示文件的详细信息）

-F： 显示文件名,同时显示类型

-R： 显示目录中所有文件及子目录中的文件结构

-t： 按照修改时间排序显示

-s： 显示文件的大小，以 block 为单位

-S： 按照文件大小排序显示

-m： 分列显示文件名

实例：

ls

ls /bin/

ls grub.conf 查找当目录下是否存在 grub.conf

ls -l

ls --help 显示 ls 的详细的帮助信息

ls -a

6．cp

功能：复制文件

格式：cp [参数] <源路径> <目标路径>

参数：-f：文件在目标路径中存在时，则直接覆盖

-i：文件在目标路径中存在时，提示是否覆盖

-r：复制指定中所有内容和结构

-b：生成覆盖文件的备份

-a：保持文件原有属性

实例：

cp /etc/* /a1/ 把/etc 下的所有文件复制到 a1 目录中

cp /bin/ls /dir1 /bin/ls 文件到/目录取名为 dir1

cp -f /sbin/* /exe/ 直接将/sbin 下的所有文件到/exe 目录中

cp -r /soft /abc/

7．rm

功能：删除文件

格式：rm [参数] <文件名>

参数：-f： 不提示确认删除

-i： 提示确认删除

-r:　　递归删除

实例:

rm　/m1/f1　　删除/m1 目录下的 f1

rm　–f　/m1/*　删除 m1 目录下的所有文件

#rm　-rf　/m1　　强制删除一个目录

8．mv

功能：移动文件、重命名文件

格式：mv　[参数]　<源路径>　<目标路径>

参数：-i:　　如果存在文件重名则提示是否覆盖

-b:　　建立覆盖文件的备份

-f:　　如果存在文件重名则直接覆盖

实例:

mv　/m1/f1　　/m2/　移动/m1 目录下 f1 文件到/m2 目录下

mv　f1　f2　　将当前目录下的 f1 文件改名 f2

mv　-f　/d1/*　/d2/　移动/d1 中的所有文件到/d2 目录中

9．file

功能：检测文件类型

格式：file　[参数]　<文件名>

参数：　-z：检测压缩包文件

-L：跟随符号链接

实例:

file　/etc/lilo.conf

file　-z　1.gz

10．显示文本文件内容

（1）　cat

格式：cat　[参数]　<文件名>

参数：-n　显示行号

-b　显示行号，但空白行不显示行号

实例:

#cat　/etc/inittab

#cat　-n　/d1/f1　/d1/f2 >txt1

（2）more

格式：more　[参数]　<文件名>

实例:

qq.3751119655 我爱七王爷

ls|more　　将 ls 的显示结果输入到 more 中

more　/etc/rc.d/rc　　显示/etc/rc.d/rc 文件的内容

说明：可以用回车键向下换行，空格键向下换页　按Q键退出 more

（3）less

实例：

ls|less　　将 ls 的显示结果输入到 less 中

less　/var/named/localhost.zore

说明：Page up /Page Down 键上下滚动、Q键退出

（4）head、tail

实例：

head　-20　　/a1　　显示 a1 中前２０行

tail　-20　　/a1　　显示中 a1 后２０行

（5）cut

功能：显示文件中指定数目的字符、字节或字段

格式：#cut　参数　数字　文件

参数：-c　以字符为单位

-b　以字节为单位

-f　以字段为单位

-d　指定分隔符

实例：

#cut　-c 3　/etc/passwd

#cut　-f　1-6　-d　:　/etc/passwd

#cut　-b　-8　/etc/passwd

11．搜索文件

（1）　find

格式：find　查找路径　查找条件

条件：　-name　“文件名”：查找指定名称文件

-lname　“文件名”：查找指定文件所有的接文件

-user　用户名：查找指定用户拥有的文件

-group　组名：查找指定组拥有的文件

-links　n：查找拥有 n 个链接的文件

-mtime n：查找在 N 天前被修改过的文件

-atime　n：查找在 N 天前被访问过的文件

-type　d/f/b/l/p：查找指定类型的文件

Qq:575119655 我爱七王爷

-s n[bcwkMG]：查找指定大小的文件

-empty：查找为空的文件

-perm mode：查找指定属性的文件

-exec command {}\; 查找指定的文件并执行指定的命令

-newer 文件名 查找比指定文件新的文件

实例：

find / -name “named*”

find / -user root

#find ./ -type d -exec chmod 755 {} \;

#find . –empty

#find . -perm 644

#find . -newer oldfile ! -newer newfile

（2） whereis

功能：查找二进制可执行、手册及源文件

实例：

whereis umount

（3） locate

功能：在资料库中查找指定的文件

格式：#locate [参数] 文件名

参数：

-u /-U 建立资料库，-u 会由根目录开始，-U 则可以指定开始位置

-e <目录 1 ...> 排除指定目录在寻找范围之外

-l <0/1> 设置安全级别,1 启用安全模式, 0 关闭安全模式

-f <文件类型> 在资料库中排除特定类型文件排除在外

-n 至多显示 n 个输出

-r 使用正规运算式做寻找的条件

-o 指定资料库存的名称。

-d 指定资料库的路径

实例：

#locate -u

#locate chdrv

#locate -n 100 a.out

（4） which

功能：查找指定命令文件

QQ:575119655 我爱七王爷

格式：#which 文件名

实例：

#which mount

12．grep

功能：在文本文件中查找匹配的字符串

格式：#grep [参数] “字符串” 文件

参数： -? 同时显示匹配行上下的？行

-b 在匹配行前面打印该行所在的块号

-c 只打印匹配的行数，不显示匹配的内容

-f File 从文件中提取模板，空文件中包含 0 个模板，所以什么都不匹配

-h 当搜索多个文件时，不显示匹配文件名前缀

-I 忽略大小写差别

-q 取消显示，只返回退出状态。0 则表示找到了匹配的行

-l 打印匹配模板的文件清单

-L 打印不匹配模板的文件清单

-n 在匹配的行前面打印行号

-s 不显示关于不存在或者无法读取文件的错误信息

-v 反检索，只显示不匹配的行

-w 如果被\<和\>引用，就把表达式做为一个单词搜索

正则表达式元字符：

^ 锚定行的开始 如：'^grep'匹配所有以 grep 开头的行

$ 锚定行的结束 如：'grep$'匹配所有以 grep 结尾的行

. 匹配一个非换行符的字符 如：'gr.p'匹配 gr 后接一个任意字符，然后是 p

* 匹配零个或多个先前字符 如：'*grep'匹配所有一个或多个空格后紧跟 grep 的行。 .*一起用代表任意字符

[] 匹配一个指定范围内的字符，如'[Gg]rep'匹配 Grep 和 grep

[^] 匹配一个不在指定范围内的字符，如：'[^A-FH-Z]rep'匹配不包含 A-R 和 T-Z 的一个字母开头，紧跟 rep 的行

\(..\) 标记匹配字符，如'\(love\)'，love 被标记为 1

\< 锚定单词的开始，如:'\<grep'匹配包含以 grep 开头的单词的行

\> 锚定单词的结束，如：'grep\>'匹配包含以 grep 结尾的单词的行

x\{m\} 重复字符 x，m 次，如：'0\{5\}'匹配包含 5 个 o 的行

x\{m,\} 重复字符 x,至少 m 次，如：'o\{5,\}'匹配至少有 5 个 o 的行

x\{m,n\} 重复字符 x，至少 m 次，不多于 n 次，如：'o\{5,10\}'匹配 5--10 个 o 的行

\w　　匹配文字和数字字符，也就是[A-Za-z0-9]，如：'G\w*p'匹配以 G 后跟零个或多个文字或数字字符，然后是 p

\W　　\w 的反置形式，匹配一个或多个非单词字符，如点号句号等

\b　　单词锁定符，如: '\bgrepb\'只匹配 grep

+　　匹配一个或多个先前的字符。如：'[a-z]+able'，匹配一个或多个小写字母后跟 able 的串，如 loveable,enable,disable 等。

?　　匹配零个或多个先前的字符。如：'gr?p'匹配 gr 后跟一个或没有字符，然后是 p 的行。

a|b|c　　匹配 a 或 b 或 c。如：grep|sed 匹配 grep 或 sed

()　　分组符号，如：love(able|rs)ov+匹配 loveable 或 lovers，匹配一个或多个 ov。

x{m},x{m,},x{m,n}　　作用同 x\{m\},x\{m,\},x\{m,n\}

实例：

ls -l | grep '^a'

通过管道过滤 ls -l 输出的内容，只显示以 a 开头的行。

grep 'test' d*

显示所有以 d 开头的文件中包含 test 的行。

grep 'test' aa bb cc

显示在 aa，bb，cc 文件中匹配 test 的行。

grep '[a-z]\{5\}' aa

显示所有包含每个字符串有 5 个连续小写字符的字符串的行。

grep 'w\(es\)t.*\1' aa

如果 west 被匹配，则 es 就被存储到内存中，并标记为 1，然后搜索任意个字符（.*），这些字符后面紧跟着另外一个 es（\1），找到就显示该行。如果用 egrep 或 grep -E，就不用"\"号进行转义，直接写成'w(es)t.*\1'就可以了。

13．文件压缩操作

（1）gzip（*.gz）

功能：对单个文件压缩

格式：gzip [参数] 压缩（解压缩）文件名

参数：　-d：用于解压缩

-t：检验压缩文件是否损坏

-l：显示压缩文件压缩比例

-r ：递归式地查找指定目录并压缩其中的所有文件或者是解压缩

-num：　用指定的数字 num 调整压缩的速度，-1 或 --fast 表示最快

压缩方法（低压缩比）

实例：

?# gzip /a1

gzip -d /a1.gz

gzip -dv /*.gz

gzip -r /www

（2）zip

功能：压缩文件

格式：#zip [参数] 文件或目录名

-d 从压缩文件内删除指定的文件。

-F 尝试修复已损坏的压缩文件。

-t<mmddyy> 把压缩文件的日期设成指定的日期。

-u 更换较新的文件到压缩文件内。

-v 显示指令执行过程或显示版本信息。

-z 替压缩文件加上注释。

-<压缩效率> 压缩效率是一个介于 1-9 的数值。

-P 口令 利用指定口令加密压缩包

-m 添加文件到 ZIP 压缩包中

实例：

?#zip abc.zip f1 f2 f3

#zip -r –P 123 soft.zip /soft/

#zip -d soft.zip soft/wb

#zip –u soft.zip abc.log

#zip -m soft.zip install.log

（3）unzip

功能： 解压缩 zip 文件

格式： #unzip [.zip 文件]

-l 显示压缩文件内所包含的文件。

-P <密码>

-d <目录>指定文件解压缩后所要存储的目录。

实例：

?#unzip f1.zip -d /f1

#unzip –l f1.zip

（4）tar(*.tar)

QQ:575119655 我爱七王爷

功能：为文件或目录创建备份

格式：tar [参数] 文件/目录名

参数： -t：列出压缩包中的文件（*.tar）

-x：解压缩（*.tar）

-z：使用 gzip 的压缩文件

-c：创建压缩包

-f：指定文件名

-j：使用 bzip 的压缩文件

-v：显示操作信息

-C：指定解压路径（默认路径为当前路径）

-r： 向压缩包添加文件

-u：更新压缩包中的文件

-k：还原文件过程中，遇到相同文件不覆盖

-m：还原文件过程中，修改文件的时间为当前时间

实例：

```
?# tar -cvf  all.tar *.jpg
# tar -uvf all.tar logo.gif
# tar -rf all.tar *.gif
# tar  -cvfz   etc.tar.gz   /etc
# tar  -xvf  all.tar
# tar  -xzvf   etc.tar.gz  -C /soft
```

除以上的压缩工具外，还有 compress(uncompress)、bzip2(unbzip2)

14．ln

功能：建立链接文件

格式：ln [参数] <源文件> <链接文件>

参数：-s：建立软链接文件

-i：提示是否覆盖目标文件

-f：直接覆盖已存在的目标文件

实例：

```
?# ln  /etc/abc  /abc
# ln  -s  /a1    /etc/a1
```

15. touch

功能：创建空文件或修改文件的时间

格式：#touch [参数] 文件名

QQ:575119655 我爱七王爷

参数：-r 修改文件的时间为指定文件的日期时间

-d <yymmdd> 指定文件日期时间为 yymmdd

-t <yymmddhhmm> 指定文件日期时间为 yymmddhhmm

-c 不创建指定的文件

实例：

?#touch /f1 /f2

#touch -r /f1 /f2

#touch -d "6:03pm 05/06/2000" file

16. sort

功能：对文本文件进行排序

格式：sort [参数] [文件]

参数：-o 文件名 将排序结果保存到指定文件中

-u 去除重复行

实例：

?#sort /etc/passwd

#sort –o /etc/oldpass /etc/passwd

17. paste

功能：合并文本文件

实例：

?#paste f1 f2>f3

18. cmp

功能：比较两个文件是否有差异

格式：#cmp [参数] 文件 1 文件 2

参数：-l 显示不同的字节位置

-s 不显示不同之处，只显示返回状态

实例：

?#cmp f1 f2

#cmp -l f1 f2

18. diff

功能：显示文件的不同之处

格式：#diff [参数] 文件 1 文件 2

参数：-c 显示全文，并标出不同之处

-u 合并显示，并标出不同之处

实例：

qq:575119655 我爱七王爷

?#diff f1 f2

#diff -c f1 f2

19. wc

功能：统计文件中的行数、单词数、字符数

格式：#wc [参数] 文件名

参数：-c 统计字符数

-w 统计单词数

-l 统计行数

实例：

?#wc /etc/passwd

#wc -c /etc/passwd

注：

1．命令自动补齐

按 TAB 键进行自动扩充

2．命令历史记录

Linux 系统采用.history 文件存放命令历史记录，以下是与历史记录相关的操作：

！ n：调用正序第 n 个命令

!! ：调用上一条命令

history：查看命令历史记录

history -c：清除历史记录

history -w 文件名：保存历史记录到指定的文件中

3．命令重定向

“>”：将命令输出结果保存到指定文件中，如果文件不存在先建立，存在就覆盖

“>>”：将命令输出结果保存到指定文件中，如果文件不在存先建立，存在就覆盖

“<”：把文件内容输入指定命令

“<<”：将一对分隔之间的正文输入给指定命令

实例：

?#ls>abc.txt

#ls /etc >>abc.txt

#wc</etc/passwd

#wc<<!

4．命令管道（前一个命令的输出作为后一个命令的输入）

实例：ls|more

5．命令替换（取指定命令的结果）

qq:675119655 我爱七王爷

实例：# cat `ls abc`

6．多个命令执行顺序

“；”：顺序执行多个命令

“||”：前后命令的执行存在“逻辑或”关系，只有||前面的命令执行失败后，它后面的命令才被执行

“&&”：前后命令的执行存在“逻辑与”关系，只有&&前面的命令执行成功后，它后面的命令才被执

实例：

?# ls /etc;cd /etc

ls /d1/f1||touch f1

7．命令别名

实例：#alias [别名=’命令’]

三、vi 编辑器的使用

1．vi 定义

Vi 是 Unix/Linux 系统中的一种文本编辑软件

2．vi 三种模式

命令模式：删除字符、排版

插入模式：插入字符、删除、修改字符

最后行模式：通过命令操作 vi 软件

3．vi 的使用

（1）启动 VI

vi [参数] [文件名]

（2）VI 常用命令

~移动光标

左移一个字符：按 h

右移一个字符：按 l

下移一行： 按 j

上移一行： 按 k

移至行首： 按^

移至行尾： 按$

移至文件顶部：按 H

移至文件尾部：按 L

移至文件中部：按 M

前翻一屏： 按 ctrl+f

QQ:575119655 我爱七王爷

后翻一屏：　按 ctrl+b

前翻半屏：　按 ctrl+d

后翻半屏：　按 ctrl+u

移动光标到指定行：输入:数字

~插入文本

在光标右边插入文本：　按 a

在一行的结尾处添加文本：　按 A

光标左边插入文本：　按 i

在行首插入文本：　按 I

在光标所在行的下一行插入新行：按 o

在光标所在行的上一行插入新行：按 O

~撤消和重复操作

撤消上一个操作：按 u

撤消光标所在行的更改：按 U

重复操作：输入”．”

~删除文本

删除当前字符：　按 x

删除一词：　按 dw

删除一行：　按 dd（剪切）

删除行的部分内容：按 D（删除光标右的内容）或按 d0（删除光标左的内容）

删除到文件的结尾：按 dG

dL ：删除直到屏幕上最后一行的内容

dH ：删除直到屏幕上第一行的内容

dG ：删除直到工作缓存区结尾的内容(文尾)

d1G ：删除直到工作缓存区开始的内容 （文头

:n,md：从第 n 行开始删除 m 行

~复制和粘贴

复制一行内容：按 yy

粘贴：　按 p

查找和替换命令

/string：向前查找字符串

?string：向后查找字符串

n：继续上一次查找

shift+n：以与上一次相反的方向查找

qq:57219955 我爱七王爷

:%s/字符串 1 /字符串 2/g：在全文中替换字符串 1 为字符串 2

n,ms/字符串 1/字符串 2/g：替换 n 到 m 行中的字符串 1 为字符串 2

~保存和退出命令

:w：写缓冲区

:w 文件名：把缓冲区写入指定文件

:wq：保存退出

:q!：不保存退出

:wq!：保存退出，对拥有者忽略只读权限

E !：取消自上次保存以来所做的修改

ZZ：保存退出

~修改文本命令

cl ： 更改当前字符

cw ： 修改到某个单词的结尾位置

c3w ： 修改到第三个单词的结尾位置

cb ： 修改到某个单词的开始位置

c0 ： 修改到某行的结尾位置

c)：修改到某个语句的结尾位置

c4)：修改到第四个语句的结尾位置

c} ： 修改到某个段落的结尾位置

c2{ ： 修改到当前段落起始位置之前的第 2 个段落位置

ctc ： 修改当前行直到下一个字符 c 所出现位置之间的内容

C ： 修改到某一行的结尾

cc ： 修改当前行

~会话定制命令

:set： 显示 vi 变量

:set all：显示所有可能的 vi 变量和它们当前的值

:set nu：显示行号

:set nonu：隐藏行号

:set showmode：显示当前操作模式

:set noshowmode：隐藏当前操作模式

:set ai：自动对齐

:set dir=目录：设置缓冲区的位置

4．VI 的高级应用

（1）编辑多个文件

qq 575119655 我爱七王爷

#vi [-o] 文件 1 文件 2 .....

-o：同一窗口打开多个文件

:next：编辑下一个文件

:prev：编辑上一个文件

:args：显示所有打开的文件名称

Ctrl+ww ：切换窗口

:e 文件名：读入另一文件

（2）高级复制

:start,end cp dest：将 start 到 end 行的内容复制到 dest 行以下

:start,end m dest：将 start 到 end 行的内容移动到 dest 行以下

nY：将从光标所在行开始的 n 行内容暂存

:[m],[n]w<文件名>：把 m 到 n 内容另存到指定文件中

（3）其它高级应用

:X 输入口令：以口令加密保存

:ctrl+s：锁定 VI（ctrl+q 解锁）

:r !<命令>：执行指定命令，并将命令结果插入到当前文件中

:J：合并上下两行

四、软件包的管理

1．Linux 软件包主要类型

（1）RPM 包

由于 RedHat 公司开发的一种软件封包方式，可以用于多种 Linux 系统

RPM 包名称格式为：软件名-版本号.运行平台.rpm（foo-1.2.0-3.i386.rpm）

Linux 系统会使用专用的数据库记录 RPM 包的安装情况

（2）TAR 包

TAR 包封装的通常是软件源代码，并且利用了 gzip 或其它方式进行二次压缩

2．RPM 包的管理

（1）安装 RPM 包

#rpm -ivh [详细选项] RPM 包名称

详细选项：

--nodeps 忽略依赖关系

--rplacepkgs 强制覆盖已存在的 RPM 包

实例：

?#rpm –ivh grub-0.93-7.i386.rpm

（2）升级 RPM 包

qq 575119655 我爱七王爷

?#rpm -Uvh [--nodeps/--replacepkgs] 软件包名称

实例：

?#rpm –Uvh bind-9.2.20-8.i385.rpm

（3）查询 RPM 包

#rpm -q[a/f 文件名/l/] [软件名称]

实例：

#rpm –q bind（查询软件名为 bind 的软件包）

#rpm –qa（查询系统中所有已安装的 RPM 包）

#rpm –qf /boot/grub/grub.conf （查询 grub.conf 文件所属的 RPM 包）

#rpm -ql bind （查询软件名为 bind 软件包并显示软件包的内容）

（4）卸载 RPM 包

#rpm -e 软件名称

实例：#rpm -e bind

（5）检验 RPM 包

#rpm –V 软件名称

实例：#rpm -V bind

3．TAR 软件包的管理

（1）安装 TAR 包

解压缩

配置安装参数

编译和安装

实例：

#tar –xzvf bind-9.2.0.tar.gz

#cd bind-9.2.0

#./configure --prefix=/usr/named

#make

#make install

（2）卸载 TAR 包

直接删除安装

实例：

#rm -rf /usr/named

*过滤文本 grep

grep [选项] 需要查找的字符串 文件名

qq:575119655 我爱七王爷

grep RMP install.log

*删除某个目录及其所有文件及子目录 rm

rm [选项] 文件或目录

rm -i 交互式删除

*改变指定文件的访问时间和修改时间 touch

touch [选项] 设定的时间 文件

改变文件的访问时间为系统当前时间 touch -a text.txt

文件的访问时间，修改时间和改变时间

- 访问时间（Access）：读取一次文件的内容，访问时间便会更新。比如对文件使用 less 命令或者 more 命令。（ls、stat 这样的命令不会修改文件访问时间）。
- 修改时间（Modify）：对文件内容修改一次便会更新该时间。例如使用 vim 等工具更改了文件内容并保存后，文件修改时间发生变化。通过 ls –l 列出的时间便是这个时间。要想看到文件访问时间可使用 ls –ul 命令。
- 改变时间（Change）：更改文件的属性便会更新该时间，比如使用 chmod 命令更改文件属性，或者执行其他命令时隐式的附带更改了文件的属性如文件大小等。

简单记忆

访问时间——>进去看了我的文件内容，我就要记着时间

修改时间——>改了我的文件内容，我就要记着时间。可以随便看。

改变时间——>改了我文件的属性，我就要记着时间。

案例：

创建一个文件 touch new.txt

查看文件状态 stat new.txt

QQ:675119655 视野七王爷

linux umask 使用详解

umask 使用方法

A 什么是 umask?

当我们登录系统之后创建一个文件总是有一个默认权限的，那么这个权限是怎么来的呢?这就是 umask 干的事情。umask 设置了用户创建文件的默认 权限，它与 chmod 的效果刚好相反，umask 设置的是权限“补码”，而 chmod 设置的是文件权限码。一般在/etc/profile、$ [HOME]/.bash_profile 或$[HOME]/.profile 中设置 umask 值。

你的系统管理员必须要为你设置一个合理的 umask 值，以确保你创建的文件具有所希望的缺省权限，防止其他非同组用户对你的文件具有写权限。在已经登录之后，可以按照个人的偏好使用 umask 命 令来改变文件创建的缺省权限。相应的改变直到退出该 shell 或使用另外的 umask 命令之前一直有效。一般来说，umask 命令是在/etc /profile 文件中设置的，每个用户在登录时都会引用这个文件，所以如果希望改变所有用户的 umask，可以在该文件中加入相应的条目。如果希望永久 性地设置自己的 umask 值，那么就把它放在自己$HOME 目录下的.profile 或.bash_profile 文件中。

B 如何计算 umask 值

umask 命令允许你设定文件创建时的缺省模式，对应每一类用户(文件属主、同组用户、其他用户)存在一个相应的 umask 值中的数字。对于文件来说，这一数字的最 大值分别是 6。系统不允许你在创建一个文本文件时就赋予它执行权限，必须在创建后用 chmod 命令增加这一权限。目录则允许设置执行权限，这样针对目录来 说，umask 中各个数字最大可以到 7。

该命令的一般形式为：

umask nnn

其中 nnn 为 umask 置 000-777。

计算出你的 umask 值：

可以有几种计算 umask 值的方法，通过设置 umask 值，可以为新创建的文件和目录设置缺省权限。下表列出了与权限位相对应的 umask 值。

在计算 umask 值时，可以针对各类用户分别在这张表中按照所需要的文件/目录创建缺省权限查找对应的 umask 值。

例如，umask 值 002 所对应的文件和目录创建缺省权限分别为 6 6 4 和 7 7 5。

有一种计算 umask 值的方法。我们只要记住 umask 是从权限中“拿走”相应的位即可。

umask 值与权限

umask 文件 目录

0 6 7

1 6 6

2 4 5

3 4 4

4 2 3

5 2 2

6 0 1

7 0 0

例如，对于 umask 值 0 0 2，相应的文件和目录缺省创建权限是什么呢?

第一步，我们首先写下具有全部权限的模式，即 777 (所有用户都具有读、写和执行权限)。

第二步，在下面一行按照 umask 值写下相应的位，在本例中是 0 0 2。

第三步，在接下来的一行中记下上面两行中没有匹配的位。这就是目录的缺省创建权限。

第四步，对于文件来说，在创建时不能具有执行权限，只要拿掉相应的执行权限比特即可。

这就是上面的例子， 其中 u m a s k 值为 0 0 2:

1) 文件的最大权限 rwx rwx rwx (777)

2) umask 值为 0 0 2 --- --- -w-

3) 目录权限 rwx rwx r-x (775) 这就是目录创建缺省权限

4) 文件权限 rw- rw- r-- (664) 这就是文件创建缺省权限

下面是另外一个例子，假设这次 u m a s k 值为 0 2 2:

1) 文件的最大权限 rwx rwx rwx (777)

2) u m a s k 值为 0 2 2 --- -w- -w-

3) 目录权限 rwx r-x r-x (755) 这就是目录创建缺省权限

4) 文件权限 rw- r-- r-- (644) 这就是文件创建缺省权限

C 常用的 umask 值

下表列出了一些 umask 值及它们所对应的目录和文件权限。

常用的 umask 值及对应的文件和目录权限

umask 值 目录 文件

0 22 7 5 5 6 4 4

0 27 7 5 0 6 4 0

0 02 7 7 5 6 6 4

0 06 7 7 1 6 6 0

0 07 7 7 0 6 6 0

D umask 命令

如果想知道当前的 umask 值，可以使用 umask 命令：

$umask

如果想要改变 umask 值，只要使用 umask 命令设置一个新的值即可：

$ umask 002

确认一下系统是否已经接受了新的 u m a s k 值：

$umask

002

$touch testfile

$ls -l testfile

rw- rw- r--

在使用 umask 命令之前一定要弄清楚到底希望具有什么样的文件/目录创建缺省权限。否则可能会得到一些非常奇怪的结果；例如，如果将 umask 值设置为 6 0 0，那么所创建的文件/目录的缺省权限就是 0 6 6!

QQ:575119655 我爱七王爷

QQ:575119655 我爱七王爷