**深入解析文件系统原理**

# Inode and Block概述

文件储存在硬盘上，硬盘的最小储存单位叫“扇区”（sector），每个扇区拥有512字节。

操作系统从硬盘上读取数据时，不会按照扇区的方式读取，这样读取效率太低，而是一次性连续读取多个扇区，即一次性读取一个“块”（Block，由多个扇区的空间组成）。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。Block存储文件数据。

文件数据存储在“块”中，那么还必须找到一个地方存储文件的元信息，比如文件的所属用户、文件所属组、文件的类型、文件的权限、文件的创建时间、文件的修改时间、文件的访问时间、文件使用的Block信息、文件的硬链接数、文件的大小等等属性信息。这种储存文件元信息的区域叫做Inode。中文译名为“索引节点”，也叫i节点。

Inode 和 Block 是文件系统组成的基本核心概念，文件系统是在分区格式化的时候形成的，文件系统负责组织文件在分区上的存放的。

注意：

Inode不包含文件名。文件名是储存在目录的目录项中。

一个文件必须占用一个inode，至少占用一个block。

### 查看文件的inode信息：stat

**[root@localhost ~]# stat anaconda-ks.cfg**

**File: `install.log'**

**Size: 56380 Blocks: 120 IO Block: 4096 regular file**

**Device: fd00h/64768d Inode: 1179650 Links: 1**

**Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)**

**Access: 2016-04-13 04:23:48.019999873 +0800**

**Modify: 2016-04-13 04:49:16.057998559 +0800**

**Change: 2016-04-13 04:49:32.224998545 +0800**

**Atime、Mtime、Ctime详解：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 英文 | 别称 | 中文翻译 | 何时修改 | 查看命令 |
| Access | Atime | 访问时间 | 读取 | ls -lu |
| Modify | Mtime | 修改时间 | 写入、修改 | ls -l |
| Change/Create | Ctime | 改变时间/创建时间 | 修改文件名、写入、修改、改权限、做链接 | ls -lc |

#### 操作示例：

读取文件内容，查看Atime的改变：

**[root@localhost ~]# echo haha > a.txt**

**[root@localhost ~]# stat a.txt**

**File: `a.txt'**

**Size: 5 Blocks: 8 IO Block: 4096 regular file**

**Device: fd00h/64768d Inode: 1186522 Links: 1**

**Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)**

**Access: 2016-10-20 22:16:09.037571618 +0800**

**Modify: 2016-10-20 22:16:09.037571618 +0800**

**Change: 2016-10-20 22:16:09.037571618 +0800**

**[root@localhost ~]# cat a.txt**

**haha**

**[root@localhost ~]# stat a.txt**

**File: `a.txt'**

**Size: 5 Blocks: 8 IO Block: 4096 regular file**

**Device: fd00h/64768d Inode: 1186522 Links: 1**

**Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)**

**Access: 2016-10-20 22:16:34.920554817 +0800**

**Modify: 2016-10-20 22:16:09.037571618 +0800**

**Change: 2016-10-20 22:16:09.037571618 +0800**

文件写入内容，查看Mtime/Ctime的改变

**[root@localhost ~]# stat a.txt**

**File: `a.txt'**

**Size: 5 Blocks: 8 IO Block: 4096 regular file**

**Device: fd00h/64768d Inode: 1186522 Links: 1**

**Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)**

**Access: 2016-10-20 22:16:34.920554817 +0800**

**Modify: 2016-10-20 22:16:09.037571618 +0800**

**Change: 2016-10-20 22:16:09.037571618 +0800**

**[root@localhost ~]# echo hehe > a.txt**

**[root@localhost ~]# stat a.txt**

**File: `a.txt'**

**Size: 5 Blocks: 8 IO Block: 4096 regular file**

**Device: fd00h/64768d Inode: 1186522 Links: 1**

**Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)**

**Access: 2016-10-20 22:16:34.920554817 +0800**

**Modify: 2016-10-20 22:19:08.481558741 +0800**

**Change: 2016-10-20 22:19:08.481558741 +0800**

修改文件权限，查看Ctime的改变

**[root@localhost ~]# stat a.txt**

**File: `a.txt'**

**Size: 5 Blocks: 8 IO Block: 4096 regular file**

**Device: fd00h/64768d Inode: 1186522 Links: 1**

**Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)**

**Access: 2016-10-20 22:16:34.920554817 +0800**

**Modify: 2016-10-20 22:19:08.481558741 +0800**

**Change: 2016-10-20 22:19:08.481558741 +0800**

**[root@localhost ~]# chmod 777 a.txt**

**[root@localhost ~]# stat a.txt**

**File: `a.txt'**

**Size: 5 Blocks: 8 IO Block: 4096 regular file**

**Device: fd00h/64768d Inode: 1186522 Links: 1**

**Access: (0777/-rwxrwxrwx) Uid: ( 0/ root) Gid: ( 0/ root)**

**Access: 2016-10-20 22:16:34.920554817 +0800**

**Modify: 2016-10-20 22:19:08.481558741 +0800**

**Change: 2016-10-20 22:20:43.470565784 +0800**

创建硬链接，查看Ctime的修改

**[root@localhost ~]# stat a.txt**

**File: `a.txt'**

**Size: 5 Blocks: 8 IO Block: 4096 regular file**

**Device: fd00h/64768d Inode: 1186522 Links: 1**

**Access: (0777/-rwxrwxrwx) Uid: ( 0/ root) Gid: ( 0/ root)**

**Access: 2016-10-20 22:16:34.920554817 +0800**

**Modify: 2016-10-20 22:19:08.481558741 +0800**

**Change: 2016-10-20 22:20:43.470565784 +0800**

**[root@localhost ~]# ln /root/a.txt /tmp/a.txt**

**[root@localhost ~]# stat a.txt**

**File: `a.txt'**

**Size: 5 Blocks: 8 IO Block: 4096 regular file**

**Device: fd00h/64768d Inode: 1186522 Links: 2**

**Access: (0777/-rwxrwxrwx) Uid: ( 0/ root) Gid: ( 0/ root)**

**Access: 2016-10-20 22:16:34.920554817 +0800**

**Modify: 2016-10-20 22:19:08.481558741 +0800**

**Change: 2016-10-20 22:23:21.999560134 +0800**

修改文件名，查看Ctime的改变

**[root@localhost ~]# stat a.txt**

**File: `a.txt'**

**Size: 5 Blocks: 8 IO Block: 4096 regular file**

**Device: fd00h/64768d Inode: 1186522 Links: 2**

**Access: (0777/-rwxrwxrwx) Uid: ( 0/ root) Gid: ( 0/ root)**

**Access: 2016-10-20 22:16:34.920554817 +0800**

**Modify: 2016-10-20 22:19:08.481558741 +0800**

**Change: 2016-10-20 22:23:21.999560134 +0800**

**[root@localhost ~]# mv a.txt aa.txt**

**[root@localhost ~]# stat aa.txt**

**File: `aa.txt'**

**Size: 5 Blocks: 8 IO Block: 4096 regular file**

**Device: fd00h/64768d Inode: 1186522 Links: 2**

**Access: (0777/-rwxrwxrwx) Uid: ( 0/ root) Gid: ( 0/ root)**

**Access: 2016-10-20 22:16:34.920554817 +0800**

**Modify: 2016-10-20 22:19:08.481558741 +0800**

**Change: 2016-10-20 22:25:04.018557367 +0800**

Touch更新时间，三个时间全部修改

**[root@localhost ~]# stat aa.txt**

**File: `aa.txt'**

**Size: 5 Blocks: 8 IO Block: 4096 regular file**

**Device: fd00h/64768d Inode: 1186522 Links: 2**

**Access: (0777/-rwxrwxrwx) Uid: ( 0/ root) Gid: ( 0/ root)**

**Access: 2016-10-20 22:16:34.920554817 +0800**

**Modify: 2016-10-20 22:19:08.481558741 +0800**

**Change: 2016-10-20 22:25:04.018557367 +0800**

**[root@localhost ~]# touch aa.txt**

**[root@localhost ~]# stat aa.txt**

**File: `aa.txt'**

**Size: 5 Blocks: 8 IO Block: 4096 regular file**

**Device: fd00h/64768d Inode: 1186522 Links: 2**

**Access: (0777/-rwxrwxrwx) Uid: ( 0/ root) Gid: ( 0/ root)**

**Access: 2016-10-20 22:26:48.721870519 +0800**

**Modify: 2016-10-20 22:26:48.721870519 +0800**

**Change: 2016-10-20 22:26:48.721870519 +0800**

### 目录文件的结构

目录也是一种文件

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

文件系统内部不使用文件名引用文件，而是使用inode号码来识别文件。对于文件系统来说，文件名只是inode号码便于识别的别称，文件名是目录的数据。

### Inode的号码

#### 文件系统访问文件的步骤：

1．用户在目录中看到要访问的文件名

2．通过目录的数据找到这个文件名对应的inode号码

3．通过inode号码，获取inode信息（文件的元信息）

4．根据inode信息，找到文件数据所在的block，读出数据

#### 查看文件的inode号码

ls -i 文件名

stat 文件名

示例：

**[root@localhost ~]# ls -i aa.txt**

**1186522 aa.txt**

**[root@localhost ~]# stat aa.txt**

**File: `aa.txt'**

**Size: 5 Blocks: 8 IO Block: 4096 regular file**

**Device: fd00h/64768d Inode: 1186522 Links: 2**

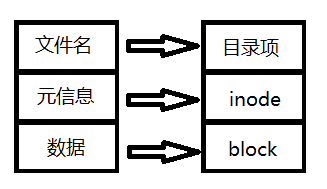
**Access: (0777/-rwxrwxrwx) Uid: ( 0/ root) Gid: ( 0/ root)**

**Access: 2016-10-20 22:26:48.721870519 +0800**

**Modify: 2016-10-20 22:26:48.721870519 +0800**

**Change: 2016-10-20 22:26:48.721870519 +0800**

**文件存储位置对照**



SuperBlock 里面存储的文件系统所有Inode、block的相关信息。

当一个用户在Linux系统中试图访问一个文件时，系统会先根据文件名去查找它的inode，看该用户是否具有访问这个文件的权限。如果有，就指向相对应的数据block； 如果没有过，就返回Permission denied（拒绝访问）

#### 删除指定inode号所对应的文件

格式：find ./ -inum inode号 -exec rm -i {} \;

示例：当文件名中包含特殊符号时我们可以删除其inode号达到删除文件的目的

**[root@localhost ~]# touch 'a b`\_c de ! 4|'.txt**

**[root@localhost ~]# ls -li**

**total 112**

**1186522 -rwxrwxrwx 2 root root 5 Oct 20 22:26 aa.txt**

**1186524 -rw-r--r-- 1 root root 0 Oct 20 22:48 a b`\_c de ! 4|.txt**

**1190719 -rw-------. 1 root root 2423 Apr 13 2016 anaconda-ks.cfg**

**1179650 -rw-r--r--. 1 root root 56380 Apr 13 2016 install.log**

**1179651 -rw-r--r--. 1 root root 11161 Apr 13 2016 install.log.syslog**

**[root@localhost ~]# find ./ -inum 1186524 -exec rm -i {} \;**

**rm: remove regular empty file `./a b`\_c de ! 4|.txt'? y**

**[root@localhost ~]# ls**

**aa.txt anaconda-ks.cfg install.log install.log.syslog**

从/root/目录下查找3天内修改的文件，并复制到/tmp下

**[root@agent ~]# find /root/ -mtime -3 -a -type f -exec cp {} /tmp \;**

**补充：find的以时间为条件查找**

**-amin +n 查找n分钟以前被访问过的所有文件。**

**-atime -n 查找n天以内被访问过的所有文件。**

**-cmin +n 查找n分钟以前文件状态被修改过的所有文件。**

**-ctime +n 查找n天以前文件状态被修改过的所有文件。**

**-mmin +n 查找n分钟以前文件内容被修改过的所有文件。**

**-mtime +n 查找n天以前文件内容被修改过的所有文件。**

**————————————————**

#### 查看文件系统的inode与block的信息

df -i 设备名（文件系统已挂载时查询，查询inode总数与已用数量）

dumpe2fs -h 设备名（文件系统无需挂载）ext有效

tune2fs -l 设备名（文件系统无需挂载）ext有效

示例：

**[root@localhost ~]# df -i /data**

**Filesystem Inodes IUsed IFree IUse% Mounted on**

**/dev/sdb1 66384 11 66373 1% /data**

**[root@localhost ~]# umount /data**

**[root@localhost ~]# dumpe2fs -h /dev/sdb1**

**----------------部分忽略----------------**

**Inode count: 66384**

**Block count: 265064**

**Reserved block count: 13253**

**Free blocks: 252369**

**Free inodes: 66373**

**First block: 0**

**Block size: 4096**

**----------------部分忽略----------------**

**First inode: 11**

**Inode size: 256**

**----------------部分忽略----------------**

**[root@localhost ~]# tune2fs -l /dev/sdb1**

**----------------部分忽略----------------**

**Inode count: 66384**

**Block count: 265064**

**Reserved block count: 13253**

**Free blocks: 252369**

**Free inodes: 66373**

**First block: 0**

**Block size: 4096**

**----------------部分忽略----------------**

**First inode: 11**

**Inode size: 256**

**----------------部分忽略----------------**

格式化时指定文件系统的inode个数与block大小

mkfs.ext4 -N inode数 -b 块大小（单位字节） 设备名

示例：inode数设定为60000个，block大小为1KB

**[root@localhost ~]# mkfs.ext4 -N 60000 -b 1024 /dev/sdb1**

**[root@localhost ~]# dumpe2fs -h /dev/sdb1 | grep -i "inode count"**

**dumpe2fs 1.41.12 (17-May-2010)**

**Inode count: 60320**

**[root@localhost ~]# dumpe2fs -h /dev/sdb1 | grep -i "block size"**

**dumpe2fs 1.41.12 (17-May-2010)**

**Block size: 1024**

**Xfs文件系统用xfs\_info查看，bsize就是block大小**

**[root@localhost ~]# xfs\_info /dev/sda1|grep bsize**

**data = bsize=4096 blocks=262144, imaxpct=25**

**naming =version 2 bsize=4096 ascii-ci=0 ftype=1**

**log =internal bsize=4096 blocks=2560, version=2**

#### 解决inode耗尽导致的磁盘故障

1. 删除不使用的文件，配置磁盘配额

**[root@localhost ~]# find /data -mtime -1 -exec rm -rf {} \;**

**[root@localhost ~]# find /data -mtime -1 |xargs rm -rf**

2．将文件备份，重新格式化此文件系统，指定较多的inode个数

**[root@localhost ~]# df -i /test**

**Filesystem Inodes IUsed IFree IUse% Mounted on**

**/dev/sdb2 28224 11 28213 1% /test**

**[root@localhost ~]# touch /test/{1..28213}.txt**

**[root@localhost ~]# touch /test/hello.txt**

**touch: cannot touch `/test/hello.txt': No space left on device**

使用第二种方法：

**[root@localhost ~]# mkdir /backup**

**[root@localhost ~]# mv /test/\* /backup/**

**[root@localhost ~]# umount /test**

**[root@localhost ~]# mkfs.ext4 -N 50000 /dev/sdb2 &> /dev/null**

**[root@localhost ~]# mount /dev/sdb2 /test**

**[root@localhost ~]# mv /backup/\* /test**

**mv: overwrite `/test/lost+found'? y**

**[root@localhost ~]# touch /test/hello.txt**

## Linux文件系统软硬链接原理

### 硬链接（hard link）

方法：ln 源文件 目标文件

特点：

硬链接指向的文件inode号，新生成的硬链接文件的inode号与源文件的inode号相同，不可针对目录进行硬链接，必须在同一文件系统内。删除一个文件名，不影响另外一个的访问。

示例：

**[root@localhost ~]# df -h /data**

**Filesystem Size Used Avail Use% Mounted on**

**/dev/sdb1 988M 2.8M 933M 1% /data**

**[root@localhost ~]# cd /data**

**[root@localhost data]# > a.txt**

**[root@localhost data]# ln a.txt hardlink-a.txt**

**[root@localhost data]# ls -li**

**total 12**

**12 -rw-r--r-- 2 root root 0 Oct 20 23:19 a.txt**

**12 -rw-r--r-- 2 root root 0 Oct 20 23:19 hardlink-a.txt**

**11 drwx------ 2 root root 12288 Oct 20 23:06 lost+found**

**[root@localhost data]# echo "****hard test" > a.txt**

**[root@localhost data]# rm -rf a.txt**

**[root@localhost data]# cat hardlink-a.txt**

**hard test**

### 软链接（soft link）

方法：ln -s 源文件或目录 目标文件或目录

特点：

软连接又被称为符号链接

软链接指向的文件名，新生成的软链接文件的inode号与源文件不同，目录也可以生成软链接，软链接文件与源文件可以不在同一文件系统内，软链接文件的内容是源文件的路径，读取时系统会自动导向源文件路径，根据源文件找到文件内容，但当源文件移动或重命名时，软链接将报错。

示例：

**[root@localhost data]# echo "soft test" > a.txt**

**[root@localhost data]# ln -s a.txt softlink-a.txt**

**[root@localhost data]# ll -i**

**total 13**

**12 -rw-r--r-- 1 root root 10 Oct 20 23:24 a.txt**

**11 drwx------ 2 root root 12288 Oct 20 23:06 lost+found**

**13 lrwxrwxrwx 1 root root 5 Oct 20 23:25 softlink-a.txt -> a.txt**

重命名源文件，硬链接不受影响，但软链接失效

**[root@localhost data]# mv a.txt b.txt**

**[root@localhost data]# ls -li**

**total 13**

**12 -rw-r--r-- 1 root root 10 Oct 20 23:24 b.txt**

**11 drwx------ 2 root root 12288 Oct 20 23:06 lost+found**

**13 lrwxrwxrwx 1 root root 5 Oct 20 23:25 softlink-a.txt -> a.txt**

**[root@localhost data]# cat softlink-a.txt**

**cat: softlink-a.txt: No such file or directory**

硬链接与软链接的对比

|  |  |  |
| --- | --- | --- |
|  | 硬链接 | 软链接 |
| 指向 | inode号 | 文件名 |
| inode号是否相同 | 相同 | 不同 |
| 是否可以针对目录 | 不可针对目录 | 可以针对目录 |
| 可否跨文件系统 | 不可跨文件系统 | 可跨文件系统 |
| 删除、重命名、移动源文件 | 不受影响 | 失效 |
| 创建命令 | ln 源文件 目标文件 | ln -s 源文件或目录 目标文件或目录 |