# 目录和文件权限

## 1.1权限对文件

**文件是实际存储数据的地方**

r:可以读取文件的内容

w:可以修改文件的内容,但是不具备删除该文件的权限,由文件所属的目录的w权限决定

x:可以执行该文件

## 1.2权限对目录

**目录主要的内容是记录文件名称列表**

r:可以读取目录结构列表,即可以在目录中ls等操作

w:可以修改目录结构列表,即新建文件,删除文件和目录,文件重命名,mv 操作

x:表示可以进入到该目录,cd,如果没有x权限,则目录下的任何东西都看不见

## 1.3文件特殊权限:SUID,SGID,SBIT

SUID:4

SGID:2

SBIT:1

例如:chmod 4755 01.txt 就给赋给01.txt SUID

**SUID(文件所有者x上):例如**

[lw@www ~]$ ll /usr/bin/passwd

-rwsr-xr-x. 1 root root 27832 Jun 10 2014 /usr/bin/passwd

[lw@www ~]$ ll /etc/passwd

-rw-r--r-- 1 root root 2071 Sep 5 09:53 /etc/passwd

首先:suid出现在了/usr/bin/passwd的所有者的x权限上,并且是/usr/bin/passwd是二进制

其次:lw对/usr/bin/passwd具有x权限

结果:lw在执行/usr/bin/passwd时,会暂时获取root的权限,所以/etc/passwd可以被lw通过passwd命令修改

**SGID(文件或目录用户组x上):**

1. SGID在文件中和SUID很像,只是获得的是二进制文件的用户组的权限
2. SGID在目录上

如果目录包含SGID

用户对此目录含有x权限,则用户在该目录下的有效用户将变为该目录的用户组

**SBIT(目录其他用户x上):**

例如:

[lw@localhost ~]$ ll -d /tmp

drwxrwxrwt. 27 root root 4096 Nov 12 09:11 /tmp 注意“t”

在此目录用户建立的文件只有该用户和root用户可以删除,

当然你可能会说,tmp目录用户，用户组,其他人都有w权限啊,应该都可以删除tmp里面的文件/目录啊,但这就是”t”的效果

## 1.4绝对权限

这个权限是除了 rwx SUIG SGID SBIT之外的权限,这个权限是输入所有者的,这个权限能改变文件/目录的权限

一个文件的绝对权限是属于root用户或者文件所有者的

例如:

[lw@localhost ~]$ ll 02.txt

-r--r--r-- 1 lw lw 0 Nov 12 09:38 02.txt

但是怎么修改文件的权限呢？rwx貌似没有对应的权限是能够修改用户的权限的,

但是lw这个用户确实能chmod 777 02.txt来修改用户权限

1. 如果root用户修改文件所有者

-r--r--r-- 1 root lw 0 Nov 12 09:38 02.txt

此时lw用户变不再可以改变02.txt的权限了

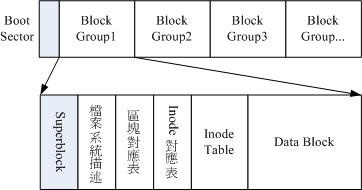
1. root能改变所有文件/目录的权限

# 2.inode和block

注意:下面的内容是基于ext2/ext3/ext4文件系统来讲的

dumpe2fs 只对ext系列的文件系统有效

文件系统格式化后就成了下面的(ext2)



**superblock(超级块)**:记录整个文件系统相关信息的地方

1. block和inode总量
2. 未使用和已使用的inode/block总量
3. block（1k,2k,4k）和inode（128Bytes,256 Bytes）的大小
4. 文件系统的挂载时间,最近一次写入数据的时间,最近一次检验磁盘(fsck)的时间
5. 一个validbit数值,如果此文件系统已被挂载,则valid bit 为0,未被挂载则为1

注意:superblock肯定是在第一个Block Group(块组)中的,但是其他的Block Group可能含有superblock(备份)

**文件系统描述说明**:描述每个block group(块组)的开始和结束的block号码,以及 superblock,bitmap,inodemap,datablock分别属于哪个block号码之间

**block bitmap(块对照表):**记录了block的号码和是否使用的状态,新建文件时就能找到未被使用的block号码

block号码: 是否使用

1. 使用
2. 没有使用
3. 没有使用

**inode bitmap(inode对照表)**:

和block bitmap类似

inode号码 是否使用

1. 使用
2. 没有使用
3. 使用

**inode table(inode表格):**

记录文件的属性stat 01.txt 所包含的属性还有该文件的真正内容的指向,innode的数量在文件系统被格式化时就固定了

inode号码 所有这 所属组 权限 ctime atime mtime blockId1 blockId2 blockId3(间接) blockId4(双间接) blockId5(三间接) ...

...............................

注意:

1. inode大小为128bytes/256bytes,记录一个blockId需要花4bytes
2. 每个文件占用一个inode
3. 系统读取文件时,先找到inode,并分析inode所记录的各种权限,最后去读inode所指向的block
4. 总额:以block=1k,将直接,间接,双间接,三间接加总:12k+256\*1k(间接)+256.256k(双间接) + 256.256.256k(三间接)=16GB

