##### 认识Linux下用户的分类

//宏观上面可以分作两类 root：超级用户 普通用户 x

//这两个的区分就是 root基本不受权限的约束，普通用户是受权限约束的

1. 两个用户之间的切换

//首先提示一点，Linux中所有的用户，都要有密码。即便有多个用户，也都要设置密码。 尽量把root密码和普通账号的密码设置的不一样

1. 如果我们是普通用户，在命令行输入su，再去输入root的密码，就能让普通用户拥有root权限.
2. 如果输入的是su -，再去输入root密码。 这样做是切换到root账户上面，而不是提高权限 ps： 注意这个 - 很重要
3. 如果我们想要退回到输入密码之前的普通用户，在命令行上面输入exit，或者ctrl+d就行了

//上面的三点是普通用户切换成为root用户的，下面的是root用户切换成普通用户的的方法

1. 如果我们是root用户，在命令行输入 su 用户名，就能指定的切换到那个普通用户
2. 从这个切换的普通用户退出到root用户，还是可以输入exit，或者ctrl+d就可以了

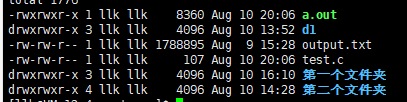
//普通用户切换成普通用户，还是在命令行输入su，但是会提示输入密码，输完想要切换用户的密码就能切换成另一个普通用户

1. 暂时的对一条命令进行权限的提高
2. 使用 sudo 相应的指令，此时sudo后面的指令就能以管理员的权限进行执行
3. 但是新创建的用户可能无法使用sudo，因为没有颁发执行sudo，除非在root用户下将这个普通用户添加进系统的信任白名单

##### 什么叫做权限

1. 笼统的来说就是一件事情是否允许被做
2. 权限认证的是身份（权限和“人”有关）
3. 权限也和事物的“属性”有关

##### 文件的属性

1. 
2. 解读一下这十位字母都什么意思

//Linux系统中文件名后缀没有直接的意义，但是不代表不用，一般情况下就是用第一位来区分是什么类型

//即使将.out的文件后缀改成.exe也能照样运行这个程序。 但是将.c改成.txt又不能进行编译

//gcc是一款编译器，软件，但是不代表其他Linux系统运行的其他软件不看后缀  


1. 第一位 d ：代表文件的类型

//- 横杆代表普通文件，但是普通文件的定义有很多，比如文本文件（txt）,源文件，头文件等

//d:目录文件

//b： 块设备文件 比如磁盘文件

//c：字符设备文件 比如键盘、显示器文件等

//p：管道文件 一般是用来通信的 后面的三种知道就行

1. 其余9位三三为一组，但是都是文件的权限属性

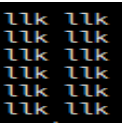
//r :可读 w ： 可写 x:可执行 - :对应的权限位置，-就代表没有权限

1. 文件角色/权限身份 分为三类 拥有着 所属组 其他人 。文件的拥有着可以设置权限这个文件都谁能查看，可以设置后两者对此文件的权限

//此时从左向右前三个对应着拥有着的权限 中间三位对应着所属组的权限 最后三位对应着其他人的权限（既不是拥有着，又不是所属组的人，包括管理员）

（4）并且这3位的每一位的位置都有固定的权限 比如前三个rwx 第一位只能r或- 第二位只能是w或- 第三位只能是x或-。 总结就是位置都代表什么权限都是固定的

（）

2.两个llk都是什么意思  


1. 第一个llk 代表文件的拥有着
2. 第二个llk 代表所属组

//至于再后面的就是字节大小和修改文件的时间了

##### 四、文件权限的更改

1. chmod 指令 change：改变 mode：方法模式

//在文件被创建之后，创建者和root都能通过chmod进行文件权限的修改，设置文件能被谁读写执行

//r :可读 w ： 可写 x:可执行 - :对应的权限位置，-就代表没有权限 文件就只能增加这几种权限

//首先对字母进行说明： u：拥有着（user） g：所属组（group） o:其他人（other） a：所有人

（1）对文件权限进行增加: chmod u+rwx test.txt 这个指令代表拥有着对此文件增加了读写执行的权限。 可以只增加其中一个

ps：对这个文件增加了可执行权限，和这个文件能不能执行并没什么关系，哪怕是这个文件不能执行，这个权限照样能够进行修改

（2）对文件权限进行减少： chmod u-rw test.txt 这个指令就是删除了拥有着读写的权限

（3）对不同身份的用户进行权限的更改： chmod u+rwx,g-rwx,o-rwx test.txt 这个指令能同时对这三种身份的权限进行修改，中间使用逗号间隔

（4）对所有身份进行权限的修改： chmod a+rwx test.txt

2.第二种权限修改的方式

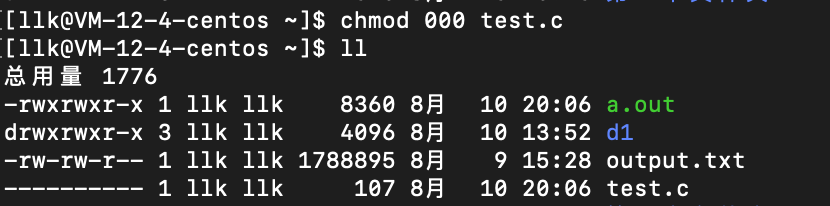
//首先我们知道三三一组，每一组的第一二三位置都是代表着对应的权限信息。其权限要么有，要么没有，所以每个位置都能用1代替

//比如拥有着位置的 rwx 就可以看做111 -rx就可以看做011

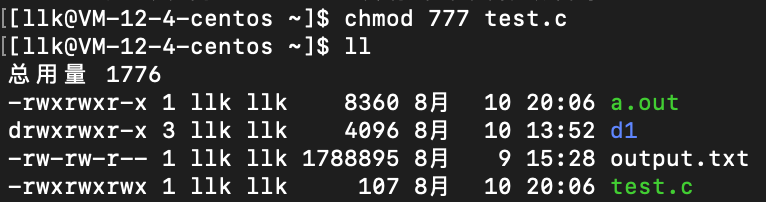
//刚好111所能代表的最大数字是7，所以还能采用8进制的形式对不同身份的权限进行修改。三个数字就能代表三个身份拥有的权限

111 111 111 ---》 rwx rwx rwx ----》7 7 7

（1）所有身份全部权限取消



（2）所有身份全部权限的增加



##### **五、**文件拥有着以及所属组的更改

//更改文件拥有着的指令是 chown （也就是将文件交给别人），但是给别人之前要征得别人的同意。当然除非是管理员，或者使用sudo提升权限

//使用格式 chown 用户名 文件名 此处的用户名是你想给的那个人的用户名

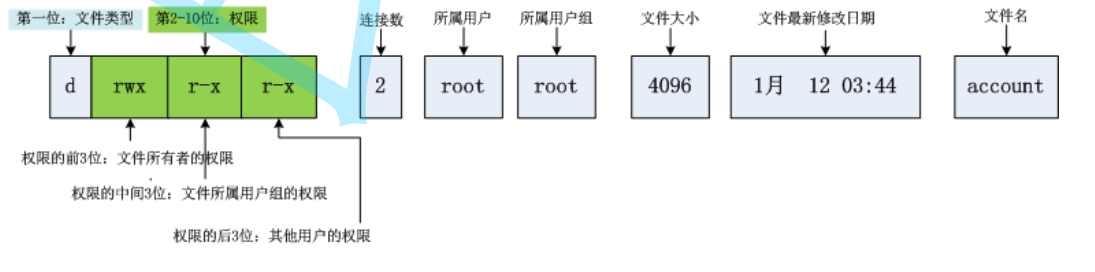
//其实将文件给别人也就是将文件拥有着的权限给别人，既然是权限，所属组的权限也能给别人。

1. 当你或者root把属于你的文件给了别的用户，文件的拥有着就是别人了，文件的所属组就是你自己。

//只有文件的拥有着和root才能对文件的权限进行修改

1. 将所属组的权限给别人使用的指令是 chgrp （change group）,使用格式是 chgrp 用户名 文件名
2. 当然也能一次性的将拥有着和所属组权限一次性给别人，使用的格式是 chown 用户名：用户名 文件名
3. 所属组也能正常添加，可以自己查看

ps：管理员命令行提示符是#，普通用户的命令行提示符是$



1. 如果拥有着

##### 创建文件、文件夹时候的默认权限

1. 
2. 创建的普通文件的最终权限：664 但是系统默认给的的是666
3. 目录文件的最终权限：775 但是系统默认给的是777

//造成这种情况的原因就是权限掩码 umask

1. umask 权限掩码

//权限掩码：凡是在umask中出现的权限，不会在最终的文件权限中出现

//User File Creation Mask 用户创建文件掩码

1. 权限掩码的值



//权限掩码还是8进制的，最前面的0不用去管，可以理解为8进制的标明，权限掩码只关注后面的三位

//可以更改umask的值， umask 0003这样的格式去更改

1. 初始权限的计算

//假设权限掩码是 002 其他人的默认权限是666 初始的普通文件的权限是怎么变成664呢？

//强调一点，并不是减法造成的。 最终权限（664）=起始权限（666） & ~umask（002） &同时真才是真 ~按位取反的位运算符

//其实也就权限掩码 002（000 000 010）先按位取反成为111 111 101，再将111 111 101和默认权限进行位运算

1. 变更新创建文件的最终权限。使用umask，然后后面跟上不想让出现的权限就行
2. 文件本身w权限被禁止之后，文件就不能被删除了

##### 创建的目录中的权限r w x 都代表什么意思

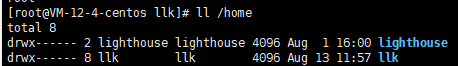
1.目录中的权限具体代表的意义



1. r（读）权限：是否允许我们查看指定目录下的文件内容，但是能够看见目录中文件的名字列表
2. w（写）权限：是否允许我们在当前目录下进行创建文件、更改、删除。
3. x（执行）权限：决定着用户能够通过路径进入这个目录
4. 也就是说文件能不能被查看修改删除，是被两个方面决定的，目录权限和文件本身的权限

##### 家目录权限

//ps:小小的提示，创建一个用户之后，home文件夹中就会创建出一个这个用户名字的文件夹

//，这个文件集中会包含各个用户的文件，各用户文件夹，只有拥有着才能够进行rwx，其他人没有权限，也就是说，我在自己的家目录创建的文件，任何人都不能进入。 每个用户都拥有一个自己专属的home文件夹

1. 很多用户想要文件的共享怎么处理？

从上面的提示可知，两个人在自己home文件夹内创建的文件是不可能进行共享的

1. 在根目录下面创建一个文件夹，这个文件中的东西都是共享的
2. 一个文件能否被删除，并不由文件本身决定
3. 但是如果去掉了文件的w权限，其他人又无法进行更改了。怎么让别人能够在这个共享目录中除了删除其他都能进行干？
4. 粘滞位 x换成t权限
5. 这个是在目录权限设置中，一般情况下是共享文件，大家可以在共享文件夹中创建的自己的文件进行各种操作。这个文件只有拥有着或者root能够删除，t就是强制你无法进行删除。
6. 

//唯一的系统自带的粘滞位文件，也就是说用户之间想要互相传递信息的时候，可以临时通过这个文件进行传递信息

##### 共享文件创建的限制

1. 共享文件夹只能由root用户创建
2. 这个文件夹只能被root删除