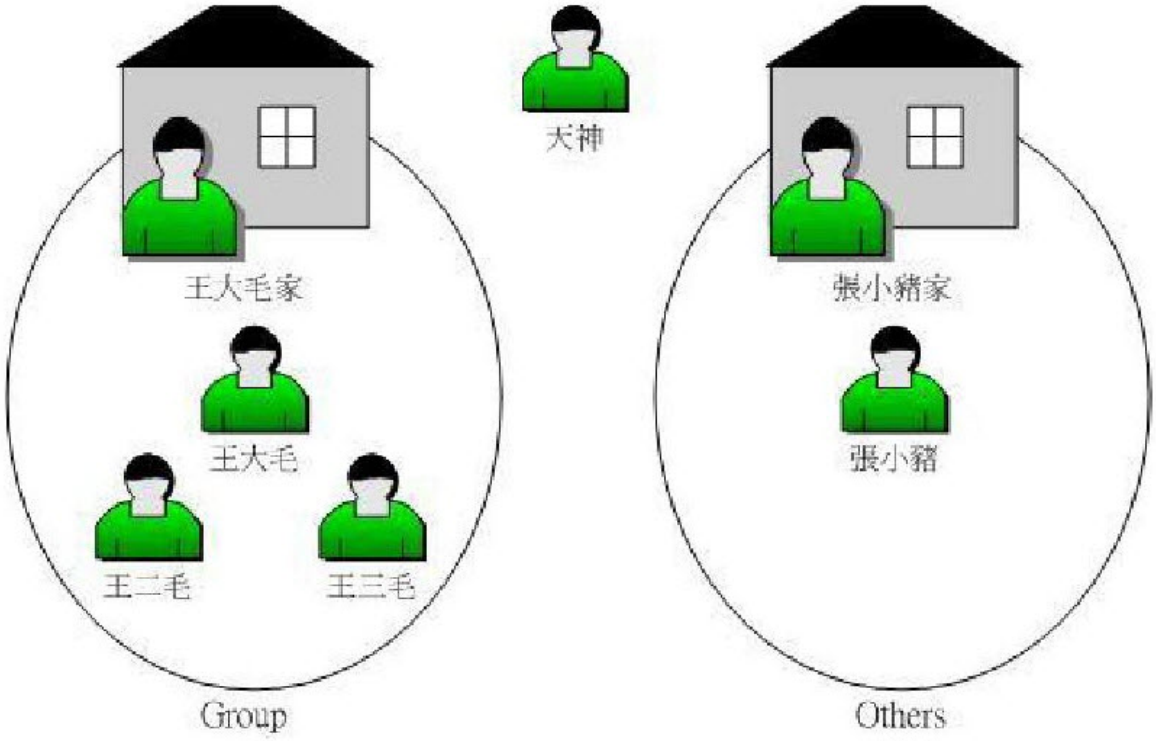
#1、用户 、用户组、其他人概述

介绍以下三个内容：

1. 用户
2. 用户组
3. 其他人的概念



文件所有者、用户组与其他人的示意图

有个人叫张小猪，他是张小猪家的人，和王大毛家没有关系，此时无法进入王大毛家；

如果张小猪和王三毛成了好朋友，他就可以通过王三毛进入王家。那么这个张小猪就是王家所谓的其他人（Others），若某一用户对于一个与他毫无关系的用户组而言,就可以被视作其他用户。

在这有一个特殊用户需要提及一下。Linux中的root用户（上图中的天神）,在整个Linux系统中拥有最大的权限,理论上能干任何事情。

##1.1 用户

假如当你将的给你心意的女神写了封Email情书转存成了文件之后，放在你自己的主文件夹中，你总不希望被其他人看见自己的情书吧？这个时候你就可以把该文件设置成只有所有者才能查看和修改该文件的内容，那么即使其他人知道你这个相当有趣的文件，不过由于你设置适当权限，所以其他人自然不知道该文件的具体内容。

由于Linux是多用户,多任务的操作系统,为此,会经常有多个用户同时使用某一台主机。为了考虑每个用户的隐私安全以及每个用户特殊的工作环境,设计了文件所有者这个概念。而文件所有者就是文件所属的用户。

Linux系统是一个多用户多任务的分时操作系统，任何一个要使用系统资源的用户，都必须首先向系统管理员申请一个账号，然后以这个账号的身份进入系统。

用户的账号一方面可以帮助系统管理员对使用系统的用户进行跟踪，并控制他们对系统资源的访问；另一方面也可以帮助用户组织文件，并为用户提供安全性保护。

每个用户账号都拥有一个惟一的用户名和各自的口令。用户在登录时键入正确的用户名和口令后，就能够进入系统和自己的主目录。

1.2 用户组

那么用户组呢？为何要配置文件所属的用户组呢?

用户组是为了团队开发资源而设计的。举例来说:某一台主机上的资源有两个团队共同使用,team A（曹操、许褚、夏侯渊）和teamB（刘备、关羽、诸葛亮）。

-rwxrwx--- caocao teamA wei.doc

-rwxrwx--- liubei teamB shu.doc

-rwx------ zhugeliang teamB qingshutohyy.txt

这两个团队之间是竞争关系,但却要提交同一份报告《如何获得天下》，每个团队的成员需要有权修改该团队其他成员的数据,同时另一个团队的成员无权查看本组自己的文件内容,此时用户组就起到了关键作用。

在Linux下我们可以的进行简单的文件权限设置,就能限制非自己团队的用户权限,同时,我们还可以设置个人私密文件,使团队内其他成员无法获取私人的文件内容。除此之外,若是有项目主管想监督这两个团队的开发进度,需要查看两个团队文件的权限,你便可以设置主管账号,同时支持这两个用户组。换句话说:每个账号都可以有多个用户组的支持。

用户组和用户关系分析案例：

可以使用“家庭”和家庭成员的关系进行分析，比如王大毛家的三兄弟：大毛、二毛、三毛，而家庭登记的王大毛的名下，他们分别有自己的房间。所以“王大毛家”（用户组）有3个人（用户）：大毛、二毛、三毛；而且这三个人分别有自己的房间，并且共同拥有一个客厅。

用户的意义：由于王大毛家3个人拥有自己的房间，所以二毛虽然可以进入三毛的房间，但是二毛不能随便翻三毛的抽屉，因为抽屉里可能有三毛的私有物品，比如情书、日记等；这是私人空间，所以不能让二毛随便动。

用户组的意义：由于共同拥有客厅，所以王家三兄弟可以在客厅看电视、看杂志、或望着天花板发呆等，总之只要是客厅的东西，三兄弟都可以使用，大家一家人嘛。

王大毛家->用户组；大毛、二毛、三毛—>分别对应一个用户.

他们三个人在同一个用户组中，可以通过设置他们文件的权限，将一些文件设置为“私有的”（只能他自己访问和使用），同用户组的其他用户无法访问和使用。而通过设置用户组共享的，则可让大家共同分享。

d rwx r-x ---

#2、 用户和用户组管理

在Linux系统当中,默认情况下所有的系统上的账号信息都记录在/etc/passwd这个文件内(包括root用户)。而个人密码记录在/etc/shadow这个文件内。所有Linux的组名都记录在/etc/group内。这三个文件非常重要,不要轻易做变动。在后续内容中,我们还会详细做介绍。

综上,用户身份与用户组的概念,能够帮助我们的Linux多任务环境变得更为容易管理。

实现用户账号的管理，要完成的工作主要有如下几个方面：

1. 用户账号的添加、删除与修改。
2. 用户口令的管理。
3. 用户组的管理。

##2.1 用户管理

用户账号的管理工作主要涉及到用户账号的添加、修改和删除。 添加用户账号就是在系统中创建一个新账号，然后为新账号分配用户号、用户组、主目录和登录Shell等资源。刚添加的账号是被锁定的，无法使用。

###2.1.1 添加新的用户

useradd [选项] 用户名

选项说明:

- -c comment 指定一段注释性描述。
- -d 目录 指定用户主目录，如果此目录不存在，则同时使用-m选项，可以创建主目录。

- -g 用户组 指定用户所属的用户组。
- -G 用户组，用户组 指定用户所属的附加组。
- -s Shell文件 指定用户的登录Shell。
- -u 用户号 指定用户的用户号，如果同时有-o选项，则可以重复使用其他用户的标识号。

参数描述：用户名:指定新账号的登录名。

案例实战

实例1：添加用户lucy,并设置他的个人主目录练习

```
[root@node1 ~]# useradd -d /usr/lucy -m lucy
```

此命令创建了一个用户lucy，其中-d和-m选项用来为登录名lucy产生一个主目录/usr/lucy（/home为默认的用户主目录所在的父目录）。

实例2:创建用户gem，指定他属于主用户组“gtjin”，附加组“adm、root”，已经登录的Shell是/bin/sh

```
[root@node1 ~]# useradd -s /bin/sh -g gtjin -G adm,root gem
```

此命令新建了一个用户gem，该用户的登录Shell是 /bin/sh，它属于gtjin用户组，同时又属于adm和root用户组，其中gtjin用户组是其主组。

增加用户账号就是在/etc/passwd文件中为新用户增加一条记录，同时更新其他系统文件如/etc/shadow, /etc/group等。

Linux提供了集成的系统管理工具userconf，它可以用来对用户账号进行统一管理。

###2.1.2 修改帐号

修改用户账号就是根据实际情况更改用户的有关属性，如用户号、主目录、用户组、登录Shell等。

修改已有用户的信息使用usermod命令，其格式如下：

usermod [选项] 用户名

常用的选项包括-c, -d, -m, -g, -G, -s, -u以及-o等，这些选项的意义与useradd命令中的选项一样，可以为用户指定新的资源值。

这个选项指定一个新的账号，即将原来的用户名改为新的用户名。

案例实战：

实例1：将用户jinxf的登录Shell修改为bash，主目录改为/home/z，用户组改为root。

```
[root@node1 ~]# cat /etc/passwd|grep gem
gm:x:1003:1000::/home/gm:/bin/sh
[root@node1 ~]# id gem
uid=1003(gem) gid=1000(gtjin) 组=1000(gtjin),0(root),4(adm)
[root@node1 ~]# usermod -s /bin/bash -g root gem
[root@node1 ~]# usermod -d /home/z -m gem
[root@node1 ~]# cat /etc/passwd|grep gem
gem:x:1003:0::/home/z:/bin/bash

```

###2.1.3 删除帐号

如果一个用户的账号不再使用，可以从系统中删除。删除用户账号就是要将/etc/passwd等系统文件中的该用户记录删除，必要时还删除用户的主目录。

删除一个已有的用户账号使用userdel命令，其格式如下：

userdel [选项] 用户名

常用的选项是-r，它的作用是把用户的主目录一起删除。

```
[root@node1 ~]# userdel jinxf
```

此命令删除用户jinxf在系统文件中（主要是/etc/passwd, /etc/shadow, /etc/group等）的记录，同时删除用户的主目录。

###2.1.4 用户口令的管理

用户管理的一项重要内容是用户口令的管理。用户账号刚创建时没有口令，但是被系统锁定，无法使用，必须为其指定口令后才可以使用，即使是指定空口令。

指定和修改用户口令的Shell命令是passwd。超级用户可以为自己和其他用户指定口令，普通用户只能用它修改自己的口令。命令的格式为：

passwd 选项用户名

可使用的选项：

- -l（lock） 锁定口令，即禁用账号。
- -u（unlock） 口令解锁。
- -d 清除用户口令

如果默认用户名，则修改当前用户的口令。

例如，假设当前用户是jinxf，则下面的命令修改该用户自己的口令：

```
[root@node1 ~]# passwd

Old password:

New password:*

Re-enter new password:*
```

如果是超级用户，可以用下列形式指定任何用户的口令：

```
[root@node1 ~]#  passwd jinxf

New password:*

Re-enter new password:*
```

普通用户修改自己的口令时，passwd命令会先询问原口令，验证后再要求用户输入两遍新口令，如果两次输入的口令一致，则将这个口令指定给用户；而超级用户为用户指定口令时，就不需要知道原口令。

为了系统安全起见，用户应该选择比较复杂的口令，例如最好使用8位长的口令，口令中包含有大写、小写字母和数字，并且应该与姓名、生日等不相同。

为用户指定空口令时，执行下列形式的命令：

```
[root@node1 ~]#  passwd -d jinxf
```

此命令将用户jinxf的口令删除，这样用户jinxf下一次登录时，系统就不再询问口令。

passwd命令还可以用-l(lock)选项锁定某一用户，使其不能登录，例如：

```
[root@node1 ~]#  passwd -l jinxf
```

##2.2 用户组管理

每个用户都有一个用户组，系统可以对一个用户组中的所有用户进行集中管理。不同Linux 系统对用户组的规定有所不同，如Linux下的用户属于与它同名的用户组，这个用户组在创建用户时同时创建。

用户组的管理涉及用户组的添加、删除和修改。组的增加、删除和修改实际上就是对/etc/group文件的更新。

1、增加一个新的用户组使用groupadd命令。

其格式如下：

groupadd [选项] 用户组

可以使用的选项有：

- -g GID 指定新用户组的组标识号（GID）。
- -o 一般与-g选项同时使用，表示新用户组的GID可以与系统已有用户组的GID相同。

实例1：添加用户组group1

```
[root@node1 ~]# groupadd group1
```

此命令向系统中增加了一个新组group1，新组的组标识号是在当前已有的最大组标识号的基础上加1。

实例2：向系统中增加了一个新组group2，同时指定新组的组标识号是101

```
[root@node1 ~]# groupadd -g 101 group2
```

此命令向系统中增加了一个新组group2，同时指定新组的组标识号是101。

2、如果要删除一个已有的用户组，使用groupdel命令，其格式如下：

groupdel 用户组

实例1：从系统中删除组group1

```
[root@node1 ~]# groupdel group1
```

3、修改用户组的属性使用groupmod命令。

其语法如下：

groupmod 选项用户组

常用的选项有：

- -g GID 为用户组指定新的组标识号。
- -o 与-g选项同时使用，用户组的新GID可以与系统已有用户组的GID相同。
- -n新用户组 将用户组的名字改为新名字

实例1：将组group2的组标识号修改为102

```
[root@node1 ~]# groupmod -g 102 group2
```

此命令。

实例2：将组group2的标识号改为10000，组名修改为group3

```
[root@node1 ~]# groupmod -g 10000 -n group3 group2
```

4、如果一个用户同时属于多个用户组，那么用户可以在用户组之间切换，以便具有其他用户组的权限。

用户可以在登录后，使用命令newgrp切换到其他用户组，这个命令的参数就是目的用户组。例如：

```
[root@node1 ~]# newgrp root
```

这条命令将当前用户切换到root用户组，前提条件是root用户组确实是该用户的主组或附加组。类似于用户账号的管理，用户组的管理也可以通过集成的系统管理工具来完成。

##2.3 用户用户组相关系统文件详讲

完成用户管理的工作有许多种方法，但是每一种方法实际上都是对有关的系统文件进行修改。与用户和用户组相关的信息都存放在一些系统文件中，这些文件包括/etc/passwd, /etc/shadow, /etc/group等。

###2.3.1 passwd文件

/etc/passwd文件是用户管理工作涉及的最重要的一个文件。Linux系统中的每个用户都在/etc/passwd文件中有一个对应的记录行，它记录了这个用户的一些基本属性。这个文件对所有用户都是可读的。它的内容如下：

```
[root@node1 ~]# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
```

```
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:999:998:User for polkitd:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
chrony:x:998:996::/var/lib/chrony:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
```

从上面的例子我们可以看到，/etc/passwd中一行记录对应着一个用户，每行记录又被冒号(:)分隔为7个字段，其格式和具体含义如下：

用户名:口令:用户标识号:组标识号:注释性描述:主目录:登录*Shell

1）"用户名"是代表用户账号的字符串。

通常长度不超过8个字符，并且由大小写字母和/或数字组成。登录名中不能有冒号(:)，因为冒号在这里是分隔符。

为了兼容起见，登录名中最好不要包含点字符(.)，并且不使用连字符(-)和加号(+)打头。

2）“口令”一些系统中，存放着加密后的用户口令字。

虽然这个字段存放的只是用户口令的加密串，不是明文，但是由于/etc/passwd文件对所有用户都可读，所以这仍是一个安全隐患。因此，现在许多Linux 系统（如SVR4）都使用了shadow技术，把真正的加密后的用户口令字存放到/etc/shadow文件中，而在/etc/passwd文件的口令字段中只存放一个特殊的字符，例如“x”或者“*”。

3）“用户标识号”是一个整数，系统内部用它来标识用户。

一般情况下它与用户名是一一对应的。如果几个用户名对应的用户标识号是一样的，系统内部将把它们视为同一个用户，但是它们可以有不同的口令、不同的主目录以及不同的登录Shell等。

通常用户标识号的取值范围是0～65535。0是超级用户root的标识号，1～99由系统保留，作为管理账号，普通用户的标识号从100开始。在Linux系统中，这个界限是500。

4）“组标识号”字段记录的是用户所属的用户组。

它对应着/etc/group文件中的一条记录。

5)“注释性描述”字段记录着用户的一些个人情况。

例如用户的真实姓名、电话、地址等，这个字段并没有什么实际的用途。在不同的Linux 系统中，这个字段的格式并没有统一。在许多Linux系统中，这个字段存放的是一段任意的注释性描述文字，用做finger命令的输出。

6)“主目录”，也就是用户的起始工作目录。

它是用户在登录到系统之后所处的目录。在大多数系统中，各用户的主目录都被组织在同一个特定的目录下，而用户主目录的名称就是该用户的登录名。各用户对自己的主目录有读、写、执行（搜索）权限，其他用户对此目录的访问权限则根据具体情况设置。

7)用户登录后，要启动一个进程，负责将用户的操作传给内核，这个进程是用户登录到系统后运行的命令解释器或某个特定的程序，即Shell。

Shell是用户与Linux系统之间的接口。Linux的Shell有许多种，每种都有不同的特点。常用的有sh(Bourne Shell), csh(C Shell), ksh(Korn Shell), tcsh(TENEX/TOPS-20 type C Shell), bash(Bourne Again Shell)等。

系统管理员可以根据系统情况和用户习惯为用户指定某个Shell。如果不指定Shell，那么系统使用sh(或bash)为默认的登录Shell，即这个字段的值为/bin/sh。

用户的登录Shell也可以指定为某个特定的程序（此程序不是一个命令解释器）。

利用这一特点，我们可以限制用户只能运行指定的应用程序，在该应用程序运行结束后，用户就自动退出了系统。有些Linux 系统要求只有那些在系统中登记了的程序才能出现在这个字段中。

8)系统中有一类用户称为伪用户（psuedo users）。

这些用户在/etc/passwd文件中也占有一条记录，但是不能登录，因为它们的登录Shell为空。它们的存在主要是方便系统管理，满足相应的系统进程对文件属主的要求。

###2.3.2 shadow文件

由于/etc/passwd文件是所有用户都可读的，如果用户的密码太简单或规律比较明显的话，一台普通的计算机就能够很容易地将它破解，因此对安全性要求较高的Linux系统都把加密后的口令字分离出来，单独存放在一个文件中，这个文件是/etc/shadow文件。 有超级用户才拥有该文件读权限，这就保证了用户密码的安全性。

2、/etc/shadow中的记录行与/etc/passwd中的一一对应，它由pwconv命令根据/etc/passwd中的数据自动产生

它的文件格式与/etc/passwd类似，由若干个字段组成，字段之间用":"隔开。这些字段是：

登录名:加密口令:最后一次修改时间:最小时间间隔:最大时间间隔:警告时间:不活动时间:失效时间:标志

root:6QajDlW4L2KfJwl.J$nwHRZZjhghkSvAihGEp6cLtpHwg7/pZaXmI1cHTbI3iwFBM5rXqLGfqMEt3Po3bbUX9JX70ZJq.dfunpiR6HA0:18529:0:99999:7:::

1. "登录名"是与/etc/passwd文件中的登录名相一致的用户账号
2. "口令"字段存放的是加密后的用户口令字。如果为空，则对应用户没有口令，登录时不需要口令；如果含有不属于集合 { ./0-9A-Za-z }中的字符，则对应的用户不能登录。
3. "最后一次修改时间"表示的是从某个时刻起，到用户最后一次修改口令时的天数。时间起点对不同的系统可能不一样。例如在SCO Linux 中，这个时间起点是1970年1月1日。
4. "最小时间间隔"指的是两次修改口令之间所需的最小天数。
5. "最大时间间隔"指的是口令保持有效的最大天数。
6. "警告时间"字段表示的是从系统开始警告用户到用户密码正式失效之间的天数。
7. "不活动时间"表示的是用户没有登录活动但账号仍能保持有效的最大天数。
8. "失效时间"字段给出的是一个绝对的天数，如果使用了这个字段，那么就给出相应账号的生存期。期满后，该账号就不再是一个合法的账号，也就不能再用来登录了。

下面是/etc/shadow的一个例子：

```
[root@node1 ~]# cat /etc/shadow
root:$6$fx49SSwdulgqt9OP$gvu2YGlSE5.7KdFum7R5XXr1Od0NdL3n5VUEnmnVd8AuBY2kVHkQHHQO562VoIkJkSsUciwKlfDZ8ue54we7b.::0:99999:7:::
bin:*:18353:0:99999:7:::
daemon:*:18353:0:99999:7:::
adm:*:18353:0:99999:7:::
lp:*:18353:0:99999:7:::
sync:*:18353:0:99999:7:::
shutdown:*:18353:0:99999:7:::
halt:*:18353:0:99999:7:::
mail:*:18353:0:99999:7:::
operator:*:18353:0:99999:7:::
```

```
games:*:18353:0:99999:7:::
ftp:*:18353:0:99999:7:::
nobody:*:18353:0:99999:7:::
systemd-network:!!:18855::::::
dbus:!!:18855::::::
polkitd:!!:18855::::::
sshd:!!:18855::::::
postfix:!!:18855::::::
chrony:!!:18855::::::
ntp:!!:18855::::::
```

###2.3.3 group文件

用户组的所有信息都存放在/etc/group文件中。

将用户分组是Linux 系统中对用户进行管理及控制访问权限的一种手段。

每个用户都属于某个用户组；一个组中可以有多个用户，一个用户也可以属于不同的组。

当一个用户同时是多个组中的成员时，在/etc/passwd文件中记录的是用户所属的主组，也就是登录时所属的默认组，而其他组称为附加组。

用户要访问属于附加组的文件时，必须首先使用newgrp命令使自己成为所要访问的组中的成员。

用户组的所有信息都存放在/etc/group文件中。此文件的格式也类似于/etc/passwd文件，由冒号(:)隔开若干个字段，这些字段有：

组名:口令:组标识号:组内用户列表

1. "组名"是用户组的名称，由字母或数字构成。与/etc/passwd中的登录名一样，组名不应重复。
2. "口令"字段存放的是用户组加密后的口令字。一般Linux 系统的用户组都没有口令，即这个字段一般为空，或者是*。
3. "组标识号"与用户标识号类似，也是一个整数，被系统内部用来标识组。

4. "组内用户列表"是属于这个组的所有用户的列表，不同用户之间用逗号(,)分隔。这个用户组可能是用户的主组，也可能是附加组。

/etc/group文件的一个例子如下：

```
[root@node1 ~]# cat /etc/group
root:x:0:
bin:x:1:
daemon:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
disk:x:6:
lp:x:7:
```

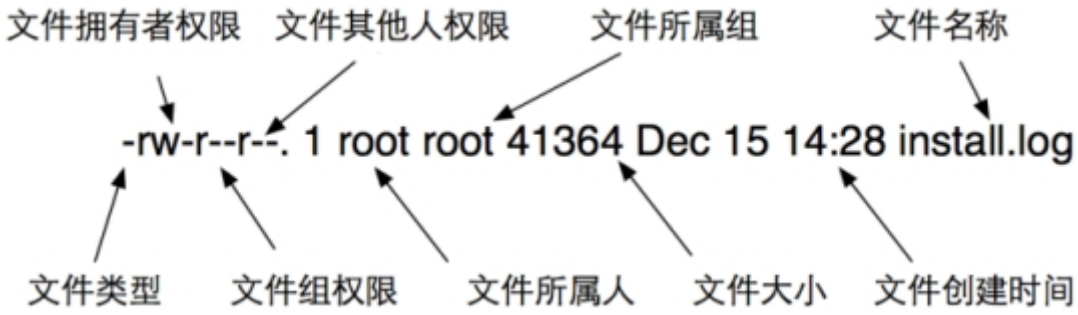
#3、Linux文件属性及权限

##3.1 Linux文件属性与权限概述

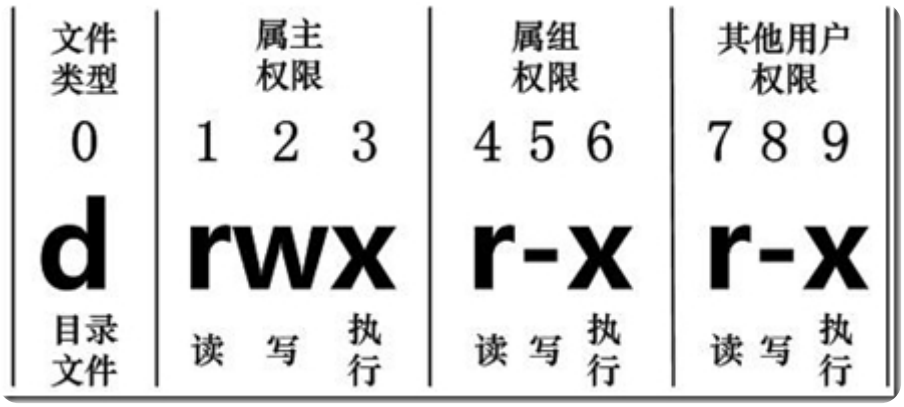
首先我们以root用户的身份登陆Linux,执行ll -a查看文件:

```
[root@node1 ~]# ll -a
总用量 108
dr-xr-x---.  4 root root  4096 8月  20 20:04 .
dr-xr-xr-x. 17 root root   244 8月  18 14:43 ..
-rw-------   1 root root  1201 8月  18 14:53 anaconda-ks.cfg
-rw-------.  1 root root 12052 8月  20 20:31 .bash_history
-rw-r--r--.  1 root root    18 12月 29 2013 .bash_logout
-rw-r--r--.  1 root root   176 12月 29 2013 .bash_profile
-rw-r--r--.  1 root root   176 12月 29 2013 .bashrc
```

ls是list的缩写,能显示文件的文件名和相关属性。而-al表示列出所有的文件详细的权限与属性(包含隐藏文件,诸如文件名以“.”开头的文件)。显示信息详细含义如下:



1. 第一列代表这个文件的类型与权限(permission),仔细看可以发现其中总共有10个字符。



(1). 第一个字符代表这个文件的具体类型:

任何设备在Linux下都是文件，不仅如此，连数据库的同学接口也有专门的文件负责。

若是[d]则是目录（directory）

若是[-]则是文件

若是[l]则是链接文件（link）,类似window系统下的快捷方式。

设备*与设备文件*[*b*]、*[c]*:

*与系统外设和存储等相关的一些文件，*通常*都*集中在/dev这个*目录中。分为*两种*:

A. 块（block）设备文件[b]：就是一些存储数据，以提供系统随机访问的接口设备，例如硬盘、软盘等。你可以随机的在硬盘的不同块读写，这种设备就是成组设备。

```
[root@node1 ~]# ll  /dev/sda
brw-rw---- 1 root disk 8, 0 8月  23 09:08
/dev/sda
```

B. 字符（character）设备文件[c]：是一些串行端口的接口设备,例如键盘,鼠标。这些设备的特征就是“一次性读取”的，不能够截断输出。

```
[root@node1 ~]# ll /dev/vcs
crw-rw---- 1 root tty 7, 0 8月  23 09:08
/dev/vcs
```

*总结：除了设备文件是我们系统很重要的文件，最好不要*随意修改*之外（通常它也不会让你修改的）；另外一个比较有*趣的*文件就是连接文件，类似window的桌面快捷方式，同样可以将linux下的连接文件简单的视为一个文件或目录的快捷方式。至于*socket与FIFO文件比较难以理解，因为他们与进程比较有关系，这个等到将来了解进程（*process）之后再*进行理解。

(2)． 余下的字符,三个一组,且均为[rwx]的3个参数组合,

其中[r]代表可读(read),[*w*]代表可写(*write*),[*x*]代表可执行(*excute*)。这三个参数的出现顺序不会改变,若没有某个权限,则会以[-]代替。这三组参数中,第一组是文件所有者的权限;第二组是同用户组的权限;第三组是其他用户的权限。这三组权限均是针对某些账号而言的权限。另外,文件权限和目录权限意义不同,这是因为文件与目录记录的数据内容不相同,后面我们会详细叙述。

1. 第二列表示有多少文件名链接到此节点(i-node)

   每个文件都会将他的权限与属性记录到文件系统的i-node中,但是Linux所使用的目录树却是使用文件名来记录,因此每个文件名就会链接到一个i-node。这个属性记录的就是有多少不同的文件名链接到相同的一个i-node号码。

2. 第三列表示这个文件(或目录)的所有者账号

3. 第四列表示这个文件的所属用户组

在Linux系统下,每个账号会附属于一个或多个用户组中。

1. 第五列表述这个文件的容量大小,默认单位为byte
2. 第六列为这个文件的创建文件日期或者是最近的修改日期(“ls -l” 等价于“ll”) ,如果想要显示完整的时间格式,可以使用ls参数,即”ll --full-time”,这样做就可以显示出完整的时间格式。
3. 第七列为该文件名

注意,我们之前提到了前缀为”.”的是隐藏文件。

Linux文件权限最大的用途实在数据安全性上,它能根据不同用户的不同权限实现对不同文件的操作。为此,在我们设置Linux文件与目录的属性之前,需要弄清到底什么数据是可变的,什么数据是不可变的。

##3.2 如何改变文件属性与权限

接下来,我们介绍几个常用于用户组,所有者,各种身份的权限的修改的命令:

chgrp:改变文件所属用户组

chown:改变文件所有者

chmod:改变文件的权限

###3.2.1 改变所属用户组

语法:

chgrp [-R] 用户组 dirname/filename

参数:

-R:如果为目录递归修改组。

作用:

使用chgrp命令可以改变一个文件的用户组,它是changegroup的简称。

注意：需要注意的是,要被改的组名必须要在/etc/group文件内存在才行,否则会报错。

案例实战：

我们使用root的身份登陆Linux,那么在/root下有一个anaconda-ks.cfg的文件,并且在/etc/group里已经存在一个名为gtjin的用户组,但是testing这个用户组并不存在于/etc/group中,我们做以下操作:

```
[root@node1 ~]# ll |grep anac
-rw------- 1 root root 1201 8月  18 14:53 anaconda-ks.cfg
[root@node1 ~]# chgrp gtjin anaconda-ks.cfg
[root@node1 ~]# ll |grep anac
-rw------- 1 root gtjin 1201 8月  18 14:53 anaconda-ks.cfg
[root@node1 ~]# chgrp testgrp anaconda-ks.cfg
chgrp: 无效的组："testgrp"
#还原所属的用户组
[root@node1 ~]# chgrp root anaconda-ks.cfg
[root@node1 ~]# ll |grep anac
-rw------- 1 root root 1201 8月  18 14:53 anaconda-ks.cfg
```

我们发现文件的用户组被改成了gtjin,但是要改成testing的时候就会发生错误。

修改目录的所属用户组:

```
[root@node1 ~]# ll |grep etc
drwxr-xr-x 75 root root 8192 8月  23 14:53 etc
[root@node1 ~]# ll etc/ | head -n 5
```

```
总用量 1056
-rw-r--r--  1 root root     16 8月  23 14:53 adjtime
-rw-r--r--  1 root root   1529 8月  23 14:53 aliases
-rw-r--r--  1 root root  12288 8月  23 14:53 aliases.db
drwxr-xr-x  2 root root    236 8月  23 14:53 alternatives
[root@node1 ~]# chgrp gtjin etc/
[root@node1 ~]# ll |grep etc
drwxr-xr-x 75 root gtjin 8192 8月  23 14:53 etc
[root@node1 ~]# ll etc/ | head -n 5
总用量 1056
-rw-r--r--  1 root root     16 8月  23 14:53 adjtime
-rw-r--r--  1 root root   1529 8月  23 14:53 aliases
-rw-r--r--  1 root root  12288 8月  23 14:53 aliases.db
drwxr-xr-x  2 root root    236 8月  23 14:53 alternatives
[root@node1 ~]# chgrp -R ntp etc
[root@node1 ~]# ll |grep etc
drwxr-xr-x 75 root ntp  8192 8月  23 14:53 etc
[root@node1 ~]# ll etc/
-rw-r--r--  1 root ntp     16 8月  23 14:53 adjtime
-rw-r--r--  1 root ntp   1529 8月  23 14:53 aliases
```

```
-rw-r--r--  1 root ntp  12288 8月  23 14:53 aliases.db
drwxr-xr-x  2 root ntp    236 8月  23 14:53 alternatives
-rw-------  1 root ntp    541 8月  23 14:53 anacrontab
......
```

###3.2.2 改变文件所有者

语法：

chown [-R] 用户名 dirname/filename

或

chown [-R] 用户名:用户组名 dirname/filename

作用：

使用chown命令可以改变一个文件的所有者,还可以直接修改群组的名称；它是changeowner的缩写。

注意：用户必须是已经存在于系统中的账号,也就是在/etc/passwd这个文件中有记录的用户名称才能改变。如果要将目录下的所有子文件或目录同时改变文件所有者,加-R参数即可。

案例实战：

1. 将.bashrc这个文件复制成为.bashrc_test文件名,修改该文件的所有者为gtjin

```
[root@node1 ~]# cp .bashrc .bashrc_test
[root@node1 ~]# ll -a .bashrc_test
-rw-r--r-- 1 root root 176 8月  23 16:22 .bashrc_test
[root@node1 ~]# chown gtjin .bashrc_test
[root@node1 ~]# ll -a .bashrc_test
-rw-r--r-- 1 gtjin root 176 8月  23 16:22 .bashrc_test
```

2. 将/etc目录拷贝到/root下，将目录的所有者改为gtjin

```
#只修改当前目录的所有者
[root@node1 ~]# ll |grep etc
drwxr-xr-x 75 root ntp  8192 8月  23 14:53 etc
[root@node1 ~]# chown gtjin etc/
[root@node1 ~]# ll |grep etc
drwxr-xr-x 75 gtjin ntp  8192 8月  23 14:53 etc
[root@node1 ~]# ll etc/ |head -n 10
总用量 1056
-rw-r--r--  1 root ntp     16 8月  23 14:53 adjtime
-rw-r--r--  1 root ntp   1529 8月  23 14:53 aliases
-rw-r--r--  1 root ntp  12288 8月  23 14:53 aliases.db
drwxr-xr-x  2 root ntp    236 8月  23 14:53 alternatives
-rw-------  1 root ntp    541 8月  23 14:53 anacrontab
-rw-r--r--  1 root ntp     55 8月  23 14:53 asound.conf
```

```
drwxr-x---  3 root ntp     43 8月  23 14:53 audisp
drwxr-x---  3 root ntp     83 8月  23 14:53 audit
drwxr-xr-x  2 root ntp     22 8月  23 14:53 bash_completion.d
#修改当前目录以及它的“后代目录和文件”的所有者
[root@node1 ~]# chown root etc/  #首先将所有者还原
[root@node1 ~]# ll |grep etc
drwxr-xr-x 75 root ntp  8192 8月  23 14:53 etc
[root@node1 ~]# chown -R gtjin etc/
[root@node1 ~]# ll |grep etc
drwxr-xr-x 75 gtjin ntp  8192 8月  23 14:53 etc
[root@node1 ~]# ll etc/ |head -n 10
总用量 1056
-rw-r--r--  1 gtjin ntp     16 8月  23 14:53 adjtime
-rw-r--r--  1 gtjin ntp   1529 8月  23 14:53 aliases
-rw-r--r--  1 gtjin ntp  12288 8月  23 14:53 aliases.db
drwxr-xr-x  2 gtjin ntp    236 8月  23 14:53 alternatives
-rw-------  1 gtjin ntp    541 8月  23 14:53 anacrontab
-rw-r--r--  1 gtjin ntp     55 8月  23 14:53 asound.conf
drwxr-x---  3 gtjin ntp     43 8月  23 14:53 audisp
```

```
34 drwxr-x---  3 gtjin ntp     83 8月  23 14:53 audit
35 drwxr-xr-x  2 gtjin ntp     22 8月  23 14:53 bash_completion.d
```

3. 将/root/etc目录的文件（包括该目录内部所有目录和文件）所有者和所属用户组都改为bin

```
[root@node1 ~]# chown -R bin:bin etc/
[root@node1 ~]# ll |grep etc
drwxr-xr-x 75 bin  bin  8192 8月  23 14:53 etc
[root@node1 ~]# ll etc/ |head -n 10
总用量 1056
-rw-r--r--  1 bin bin    16 8月  23 14:53 adjtime
-rw-r--r--  1 bin bin  1529 8月  23 14:53 aliases
-rw-r--r--  1 bin bin 12288 8月  23 14:53 aliases.db
drwxr-xr-x  2 bin bin   236 8月  23 14:53 alternatives
-rw-------  1 bin bin   541 8月  23 14:53 anacrontab
-rw-r--r--  1 bin bin    55 8月  23 14:53 asound.conf
drwxr-x---  3 bin bin    43 8月  23 14:53 audisp
drwxr-x---  3 bin bin    83 8月  23 14:53 audit
drwxr-xr-x  2 bin bin    22 8月  23 14:53 bash_completion.d
```

###3.2.3 改变权限

语法：

chmod [-R] mode dirname/filename

作用：

文件或目录权限的改变使用的是chmod(change file mode bits)这个命令。

注意：但是权限的设置方法分两种,可以通过数字或符号进行修改。

####3.2.3.1 数字类型改变文件权限

Linux的基本权限有9个,分别是owner,group,others三种身份各自的read,write,excute权限,各个权限对应的数字如下： **r:4、w:2、x:1**

0 ---、1 --x、2 -w-、3 -wx、4 r--、5 r-x、6 rw-、7 rwx 理论情况下一共8中情况，实际情况时2和3的情况几乎没有出现过。

为此每种身份各自的三个权限数字相加即可得出数字表示的权限...例如[-rwxr-x---]可以表示为:

owner = rwx = 4+2+1 = 7
group = r-x = 4+0+1 = 5
others= --- = 0+0+0 = 0

案例实战：

```
#1.修改.bashrc文件权限,改为rwxr-x---
[root@node1 ~]# chmod 750 .bashrc
[root@node1 ~]# ll .bashrc
-rwxr-x---. 1 root root 176 12月 29 2013 .bashrc
#2.修改/root/etc目录(以及目录中的所有子目录或文件)的权限，改为rwxrw-r--
[root@node1 ~]# ll |grep etc
```

```
drwxr-xr-x 75 bin  bin  8192 8月  23 14:53 etc
[root@node1 ~]# chmod -R 764 etc/
[root@node1 ~]# ll |grep etc
drwxrw-r-- 75 bin  bin  8192 8月  23 14:53 etc
[root@node1 ~]# ll etc/ | head -n 8
总用量 1056
-rwxrw-r--  1 bin bin    16 8月  23 14:53 adjtime
-rwxrw-r--  1 bin bin  1529 8月  23 14:53 aliases
-rwxrw-r--  1 bin bin 12288 8月  23 14:53 aliases.db
drwxrw-r--  2 bin bin   236 8月  23 14:53 alternatives
-rwxrw-r--  1 bin bin   541 8月  23 14:53 anacrontab
-rwxrw-r--  1 bin bin    55 8月  23 14:53 asound.conf
drwxrw-r--  3 bin bin    43 8月  23 14:53 audisp

#3.将/root/etc目录(以及目录中的所有子目录或文件)的权限，还原为rwxr-xr-x
[root@node1 ~]# chmod -R 755 etc/
[root@node1 ~]# ll |grep etc
drwxr-xr-x 75 bin  bin  8192 8月  23 14:53 etc
[root@node1 ~]# ll etc/ | head -n 8
总用量 1056
```

```
-rwxr-xr-x  1 bin bin     16 8月  23 14:53 adjtime
-rwxr-xr-x  1 bin bin   1529 8月  23 14:53 aliases
-rwxr-xr-x  1 bin bin  12288 8月  23 14:53 aliases.db
drwxr-xr-x  2 bin bin    236 8月  23 14:53 alternatives
-rwxr-xr-x  1 bin bin    541 8月  23 14:53 anacrontab
-rwxr-xr-x  1 bin bin     55 8月  23 14:53 asound.conf
drwxr-xr-x  3 bin bin     43 8月  23 14:53 audisp
```

理考：user - 4、group - 4、other -4，为何使用的拼接的方式，而不是继续使用求和的方式?

答案分析：如果使用求和的话的，4+4+4=12，但是对12进行拆解时，有多个拆解结果：444、660、750。所以不能使用求和的方式，只能通过拼接的方式表示。

####3.2.3.2符号类型改变文件权限

另一种改变权限的方法就是通过符号了,上文提到,Linux总共9种权限,对应着三种身份,为此我们可以通过u,g,o代表三种身份,另外a代表全部身份。对应的权限可以写为r,w,x,如下图所示:

chmod	u g o a	+(加入) -(除去) =(設定)	r w x	檔案或目錄

案例实战:

1.我们要将一个文件的权限改做[-rwxr-xr-x],具体来说就是:

user(u):具有可读可写可执行权限;group与others(go):具有可读可执行权限

```
#不考虑之前的权限，直接使用设定的方式
[root@node1 ~]# chmod u=rwx,g=rx,o=rx .bashrc
#还以省略为如下：
[root@node1 ~]# chmod u=rwx,go=rx .bashrc
```

2.若要改为[-rwxr-xr--],可以使用[chmod u=rwx,g=rx,o=r filename]来设置:

```
#不考虑之前的权限，直接使用设定的方式
[root@node1 ~]# chmod u=rwx,g=rx,o=r profile
```

3.要去掉全部的执行权限，但不修改其他权限:

```
[root@node1 ~]# chmod a-x .bashrc
#或
[root@node1 ~]# chmod -x .bashrc
[root@node1 ~]# ll .bashrc
-rw-r--r--. 1 root root 176 12月 29 2013 .bashrc
```

4.添加全部的可执行权限:

```
[root@node1 ~]# chmod a+x .bashrc
# 或者
[root@node1 ~]# chmod +x .bashrc
[root@node1 ~]# ll .bashrc
-rwxr-xr-x. 1 root root 176 12月 29 2013 .bashrc

```

5.还原默认的权限rw-r--r--

```
[root@node1 ~]# ll .bashrc
-rwxr-xr-x. 1 root root 176 12月 29 2013 .bashrc
#以下三种方式都可以实现
[root@node1 ~]# chmod a-x .bashrc
[root@node1 ~]# chmod u=rw,go=r .bashrc
[root@node1 ~]# chmod u-x,go=r .bashrc

```

#4、 目录与文件权限的意义

##4.1 权限对文件的重要性

文件是实际含有数据的地方,包括一般文本文件,数据库内容文件,二进制可执行文件(binary program)等等,因此,文件权限有如下意义:

- r（read）：可读取此文件的实际内容，如读取文本文件的文字内容等;
- w（write）：可以编辑、新增或者是修改该文件的内容
- x（execute）：该文件具有可以被系统执行的权限。

注意:在Linux中,文件是否能被执行是由是否具有"*x*"这个权限来决定,与拓展名无关。

```
[root@node1 ~]# echo "hello sxt" >> myfile
[root@node1 ~]# ll myfile
-rw-r--r-- 1 root root 10 8月  23 17:56 myfile
[root@node1 ~]# cat myfile
hello sxt
#既可以读，也可以写修改
[root@node1 ~]# vim myfile
[root@node1 ~]# cat myfile
Hello sxt
[root@node1 ~]# chmod u=r myfile
#修改后文件变为只读文件，不能进行修改
[root@node1 ~]# ll myfile
-r--r--r-- 1 root root 10 8月  23 17:57 myfile
```

##4.2 权限对目录的重要性

- r（read contents in directory）：表示具有读取目录结构清单的权限，所以当你具有读取（r）一个目录的权限时，表示你可以查询该目录下的文件名数据。所以你就可以利用ls这个指令将该目录的内容列表显示出来。
- w（modify contents of directory）：这个可写入的权限对目录来说，表示你具有改变目录结构清单的权限，也就是底下这些权限：
  - 建立新的文件或子目录； /opt/
  - 删除已经存在的文件或子目录（不论该文件的权限）
  - 将已存在的文件或目录进行重命名；
  - 移动该目录内的文件、目录位置。

- x（access directory）：在Linux中,目录不可以被执行,目录的x代表的是使用者能否进入该目录成为工作目录的用途。所谓的工作目录（work directory）就是你目前所在的目录。举例来说，当你登入Linux时，你所在的Home目录就是你当下的工作目录。

```
[root@node1 opt]# chmod 750 apps
[root@node1 opt]# ll
总用量 12
drwxr-x---  2 root root    6 8月  23 18:08 apps
drwxr-xr-x 75 root root 8192 8月  20 19:12 etc
[root@node1 opt]# cd apps/
[root@node1 apps]# ls
[root@node1 apps]# touch test
[root@node1 apps]# mkdir mydir
[root@node1 apps]# mv mydir mydir1
[root@node1 apps]# pwd
/opt/apps
[root@node1 apps]# su gtjin
[gtjin@node1 apps]$ pwd
/opt/apps
[gtjin@node1 apps]$ ls
ls: 无法打开目录.: 权限不够
[gtjin@node1 apps]$ cd
[gtjin@node1 ~]$ cd /opt/apps/
bash: cd: /opt/apps/: 权限不够
[gtjin@node1 ~]$ ls /opt/apps
ls: 无法打开目录/opt/apps: 权限不够
[gtjin@node1 ~]$ mkdir /opt/apps/a
mkdir: 无法创建目录"/opt/apps/a": 权限不够
```

```
[gtjin@node1 ~]$ exit
exit
[root@node1 apps]# chmod 755 /opt/apps
[root@node1 apps]# cd
[root@node1 ~]# su gtjin
[gtjin@node1 root]$ cd /opt/apps/
[gtjin@node1 apps]$ ls
mydir1  test
[gtjin@node1 apps]$ pwd
/opt/apps
[gtjin@node1 apps]$ touch newfile
touch: 无法创建"newfile": 权限不够
[gtjin@node1 apps]$ mv test test_new
mv: 无法将"test" 移动至"test_new": 权限不够
```

#5、 Linux中软件安装

- 绿色软件：解压、配置，比如：zookeeper、hadoop、hive、hbase等等
- 安装的三种方式：
  - 源码编译安装，代表软件Nginx
  - rpm安装
  - yum安装

##5.1 rpm安装软件

###5.1.1 rpm概述

RPM（RedHat Package Manager）安装管理， 这个机制最早是由Red Hat开发出来,后来实在很好用,因此很多 distributions（发行版）就使用这个机制来作为软件安装的管理方式 。包括Fedora,CentOS,SuSE等等知名的开发商。

RPM的优点

1. RPM内含已经编译过的程序与配置文件等数据,可以让用户免除重新编译的困扰
2. RPM在被安装之前,会先检查系统的硬盘容量、操作系统版本等,可避免文件被错误安装
3. RPM文件本身提供软件版本信息、相依属性软件名称、软件用途说明、软件所含文件等信息,便于了解软件
4. RPM管理的方式使用数据库记录 RPM 文件的相关参数,便于升级、移除、查询与验证

rpm默认安装的路径

1. /etc　一些配置文件放置的目录,例如/etc/crontab
2. /usr/bin 一些可执行文件
3. /usr/lib　一些程序使用的动态链接库
4. /usr/share/doc 一些基本的软件使用手册与说明文件
5. /usr/share/man 一些man page（Linux命令的随机帮助说明）文件

手动下载rpm的网址：

http://www.rpmfind.net/linux/rpm2html/search.php

5.1.2 rpm安装

基本格式

rpm [选项] package_name

选项说明

- -i :install的意思
- -v :察看更细部的安装信息画面
- -h :以安装信息列显示安装进度

安装单个rpm包

```
rpm -ivh package_name
```

安装多个rpm包

```
rpm -ivh a.rpm b.rpm *.rpm
```

安装网上某个位置rpm包

```
rpm -ivh http://website.name/path/pkgname.rpm
```

rpm安装jdk：

1. 将上传到/opt/apps目录下

```
[root@node1 apps]# ls
jdk-7u80-linux-x64.rpm
```

1. 安装当前目录下的jdk-7u80-linux-x64.rpm

```
[root@node1 apps]# rpm -ivh jdk-7u80-linux-x64.rpm
准备中...
 ################################# [100%]
正在升级/安装...
   1:jdk-2000:1.7.0_80-fcs
 ################################# [100%]
Unpacking JAR files...
    rt.jar...
    jsse.jar...
    charsets.jar...
    tools.jar...
    localedata.jar...
    jfxrt.jar...

```

1. 查找java安装目录的位置：

```
[root@node1 apps]# whereis java
java: /usr/bin/java
您在 /var/spool/mail/root 中有新邮件
[root@node1 apps]# ll /usr/bin/java
lrwxrwxrwx 1 root root 26 8月  24 10:42 /usr/bin/java -> /usr/java/default/bin/java
[root@node1 apps]# cd /usr/java/
[root@node1 java]# ll
总用量 0
lrwxrwxrwx 1 root root  16 8月  24 10:42 default -> /usr/java/latest
drwxr-xr-x 8 root root 233 8月  24 10:42 jdk1.7.0_80
lrwxrwxrwx 1 root root  21 8月  24 10:42 latest -> /usr/java/jdk1.7.0_80

```

1 配置环境变量

```
[root@node1 java]# vim /etc/profile
#在文件的最后，加入以下两行代码：
export JAVA_HOME=/usr/java/default
export PATH=$PATH:$JAVA_HOME/bin
```

1 让配置生效，使用.命令，或者source命令

```
[root@node1 java]# source /etc/profile
```

1 测试安装配置是否成功

```
[root@node1 java]# java -version
java version "1.7.0_80"
Java(TM) SE Runtime Environment (build 1.7.0_80-b15)
Java HotSpot(TM) 64-Bit Server VM (build 24.80-b11, mixed mode)
[root@node1 java]# jps
3926 Jps #表示安装配置成功!
```

rpm安装总结:

- redhat提供了rpm管理系统
- 已经编译的软件包：针对不同的平台系统编译目标软件包
- 操作系统维护安装信息
- 软件包包含依赖检查，这是人需要参与处理的。你得下载它需要的所有依赖包并安装，才能安装一个rpm软件。

###5.1.3 rpm查询:

rpm查询已安装软件,选项说明:

- -q :仅查询,后面接的软件名称是否有安装
- -qa :列出所有的,已经安装在本机Linux系统上面的所有软件名称！！！
- -qi :列出该软件的详细信息,包含开发商、版本和说明等 !!
- -ql :列出该软件所有的文件与目录所在完整文件名 !!
- -qc :列出该软件的所有配置文件 !
- -qd :列出该软件的所有说明文件
- -qR :列出和该软件有关的相依软件所含的文件
- -qf :由后面接的文件名,找出该文件属于哪一个已安装的软件

案例实战

案例1：查找是否安装jdk

```
[root@node1 ~]# rpm -qa |grep jdk
```

案例2：查询jdk所包含的文件及目录

```
[root@node1 ~]# rpm -ql jdk
```

案例3：查看jdk包的相关说明

```
[root@node1 ~]# rpm -qi jdk
```

案例4：列出iptables的配置文件

```
[root@node1 ~]# rpm -qc iptables
/etc/sysconfig/ip6tables-config
/etc/sysconfig/iptables-config
```

（无显示说明不需要配置文件）

案例5：查看apr需要的依赖

```
[root@node1 ~]# yum install apr -y
[root@node1~]# rpm -qR apr
```

###5.1.4 rpm卸载

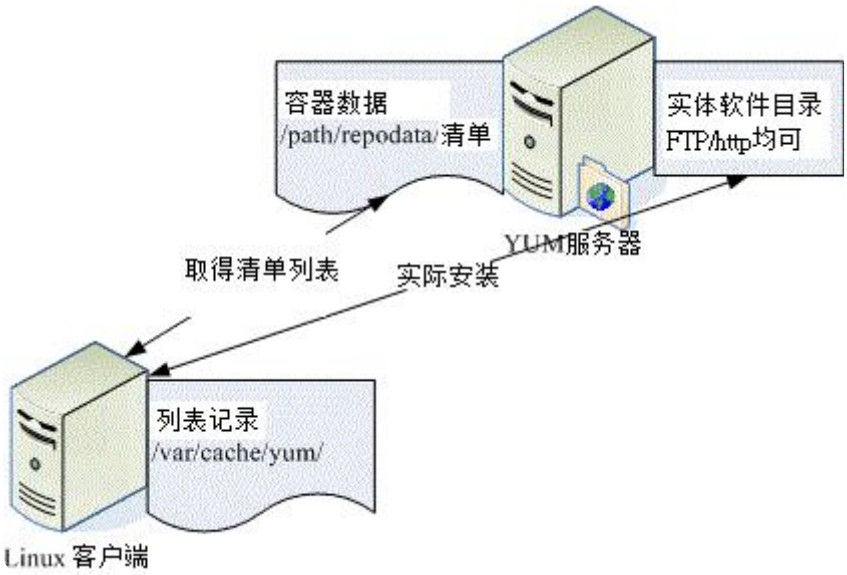
找出与apr有关的软件名称,并尝试移除apr这个软件

```
[root@node1 ~]# rpm -qa|grep apr
apr-1.3.9-5.el6_9.1.x86_64
[root@node1 ~]# rpm -e apr-1.3.9-
5.el6_9.1.x86_64
[root@node1 ~]# rpm -qa|grep apr
#如果某软件A被别的软件B所依赖，无法通过rpm -e直接删除软件A;可以通过rpm -e --nodeps xxx
[root@node1 ~]# rpm -e --nodeps apr-1.3.9-
5.el6_9.1.x86_64
```

5.2 yum安装

5.2.1 yum概述

YUM（全称为 Yellow dog Updater, Modified）是一个在Fedora和RedHat以及CentOS中的Shell前端软件包管理器。基于RPM包管理，能够从指定的yum源服务器自动下载RPM包并且安装，可以自动处理依赖性关系，并且一次安装所有依赖的软件包，无须繁琐地一次次下载和安装。



###5.2.2 yum命令

基本语法

yum [options] command [package ...]

options选项说明

选项	功能
-y	对所有询问都回答“yes”

command说明

参数	功能
install	安装rpm软件包
update	更新rpm软件包
check-update	检查是否有可用的更新rpm软件包
remove	删除指定的rpm软件包
list	显示软件包信息
clean all	清除yum的所有缓存
makecache	生成新的缓存

案例实战

1. yum源相同命令

```
#查看yum源
yum repolist
#清空yum缓存
yum clean all
#重新生成yum缓存
yum makecache
yum update  #更新系统使用该命令
```

1. 查询:

```
 #列出系统中已经安装的和可以安装的包
yum list
#列出系统中已经安装的和可以安装的包中包含lrzsz关键字的
yum list | grep lrzsz
#yum search在yum源搜索指定的包
yum search lrzsz
#打印指定包的描述信息
yum info lrzsz.x86_64
```

1. 安装和卸载:

```
yum -y install lrzsz
yum remove|erase lrzsz
```

1. yum分组命令：

```
#查询yum源中rpm包的组信息
yum grouplist
#查看指定组的信息
yum groupinfo "开发工具"
#安装指定组的所有包
yum groupinstall "development"
#更新指定软件组
yum groupupdate "development"
#删除指定软件组
yum groupremove "development"
```