第 4 章

用户与用户组管理

用户和用户组是系统最为重要的组成之一，是系统资源的分配者。在本章中，我们将学习系统中用户和用户组的管理，通过用户与用户组的学习使大家对 Linux 系统中的用户和用户组有进一步的了解。

4.1 用户与用户组管理

用户一般是指使用计算机的人，计算机针对使用它的每一个人给了一个特定的名称，用户可以使用这些名称来登录使用计算机。除了人之外，一些系统服务也需要含有部分特权的用户账户运行。出于安全考虑，用户管理应运而生，其目的是明确限制各个用户账户的权限。root 在计算机中拥有至高特权，所以一般只做管理用，非特权用户可以通过 SU 来临时获得特权。要想实现用户账号的管理，需要完成的工作主要有如下几个方面：用户账号的添加、删除、修改以及用户密码的管理。

用户组（group）就是具有相同特性的用户（user）集合。有时我们需要让多个用户具有相同的权限，比如查看、修改某一个文件或目录，如果不用用户组，这种需求在授权时就很难实现。使用用户组就方便多了，只需要把授权的用户都加入同一个用户组里，然后通过修改该文件或目录对应的用户组权限，让用户组具有符合需求的操作权限，这样用户组下的所有用户对该文件或目录就会具有相同的权限，这就是用户组的用途。

4.1.1 系统用户的分类

对于“用户”这个概念，可以把它理解为能够获取系统资源权限相同特征的逻辑集合。Linux 系统下的用户分为管理员、虚拟用户和普通用户这三类。每个用户都有唯一的 UID。UID 的最大数量在 2.6 系列内核版本时已经达到 2^32-1 个，不管 UID 个数有多少，在同一个范围内 UID 的用户拥有相同的权限，但系统中每个用户的 UID 都具有唯一性。

在系统中一般把 UID 的范围划分为 3 个部分，按数值从小到大排列，0 作为一个部分，1~499 作为第二部分，500 之后作为第三部分。其中，UID 是 0 的用户拥有系统最高的权限，只属于 root 用户所有；1~499 属于系统的虚拟用户所有；500 之后的 UID 属于系统的普通用户所有。

1. 系统管理员用户账号

系统管理员的 UID 是 0，默认的用户名是 root（不要尝试更改名称，这会在执行命令时引起很多权限错误的问题）。系统管理员拥有除内核之外的系统最高权限，可对整个系统的所有文件、目录和进程及其他的资源进行控制。因此，它可以执行系统中的所有程序，任何文件的权限对于它都是无效的。由于系统管理员是所有用户中权限最高的，因此又称为超级管理员。

root 用户拥有非常大的权限，操作不当就可能会对整个系统造成灾难性的损失，因此在工作中除非必要，应该尽量避免使用 root 用户登录系统。在日常的系统检查工作和学习中，尽可能不使用 root 用户来操作。

2. 系统普通用户账号

普通用户是指那些可以登录系统、拥有属于自己独立目录且能够控制属于自己目录和文件的用户。普通用户受控于来自系统管理员的权限，而且只拥有极少数的系统级资源但具有能够独立执行属于自己任务的权限。

由于普通用户对系统造成的危害不大且基本可以满足日常的学习需要，因此日常的学习中建议使用普通账户。

3. 系统虚拟用户账号

虚拟用户又称伪用户，此类账号是在系统安装过程中创建的一些与安装包对应的账号，它们没有登录系统的权限且不属于任何人，主要用于特定的系统目的（如用来执行特定子系统完成服务所需要的进程等）。

虽然虚拟用户不具有登录系统的权限，但是这类用户都没有设置密码，因此它们常被非法人员作为获取系统登录权限的主要“跳板”，并利用/tmp 目录来使用（该目录对所有用户都开放使用权）。

4.1.2　用户和组的关系

Linux 系统本身就是一个支持多用户的系统，而且这些系统往往有着自己对应的组，如图 4-1 所示。

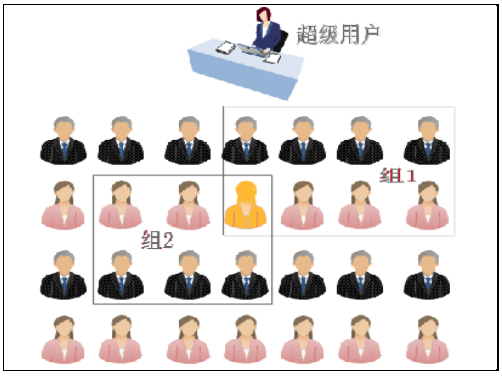


图 4-1　用户与组

用户与组之间的关系主要存在一对一和多对一的关系。

- 一对一：一个用户可以存在一个组中。
- 多对一：多个用户可以存在一个组中。
- 一对多：一个用户可以存在多个组中。
- 多对多：多个用户可以存在多个组中。

在使用用户与组时，有三个文件值得注意，如表 4-1 所示。

表4-1 用户与组相关的文件

名 称	账号信息	说 明
用户配置文件	/etc/passwd	记录了每个用户的一些基本属性，并且对所有用户可读。每行记录对应一个用户，并通过冒号进行分隔
用户组文件	/etc/group	用户组的所有信息存放地，并且组名不能重复
用户对应的密码信息	/etc/shadow	passwd 文件对所有用户是可读的，为安全起见把密码从 passwd 中分离出来放入这个单独的文件中。该文件只有 root 用户拥有读权限，以保证密码安全性

4.2 系统用户管理

用户是使用操作系统的主要手段之一，没有用户就无法使用系统，因此用户的存在是非常重要的。本节将学习用户的管理，以及对用户的基本添加和查询等操作。

4.2.1 添加用户

1. 语法

创建或向系统添加用户，可使用 useradd 命令，语法格式如下：

```
#useradd 选项 用户名
```

常用选项：

- -g: 指定用户的用户主（主要）组，选项的值既可以是用户组的 id，也可以是组名。
- -G: 指定用户的用户附加（额外）组，选项的值既可以是用户组的 id，也可以是组名。
- -u: uid，用户的 id（用户的标识符），系统默认会从 1000 之后按顺序分配 uid。如果不想使用系统分配的，就可以通过该选项自定义（类似于腾讯 QQ 的自选靓号情况）。
- -c: comment，添加注释。
- -s: 指定用户登录后所使用的 shell 解释器（专门的接待员）。
- -d: 指定用户登录时的启始目录（家目录位置）。
- -n: 取消建立以用户名称为名的群组。

2. 示例

创建用户 test，不带任何选项：

```
[root@test ~]# useradd test
```

创建新用户时，系统会在/home/目录下创建对应于用户名称的一个目录：

```
[root@test ~]# ll /home/
drwx------ 2 test test 62 Dec 20 09:44 test
```

验证是否成功：

① 验证/etc/passwd 的最后一行，查看是否有新建账号 test 的信息：

```
[root@test ~]# cat /etc/passwd | grep test
test:x:1000:1000::/home/test:/bin/bash
```

② 验证是否存在家目录（在 CentOS 下创建好用户之后产生一个同名家目录）。

3. 认识 passwd 文件

在 passwd 文件中，每个用户对应于一行记录信息，格式相同，但内容不一定相同。如图 4-2 所示是该文件的部分记录信息。

```
1 root:x:0:0:root:/root:/bin/bash
2 bin:x:1:1:bin:/bin:/sbin/nologin
3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
4 adm:x:3:4:adm:/var/adm:/sbin/nologin
5 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
6 sync:x:5:0:sync:/sbin:/bin/sync
7 shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
8 halt:x:7:0:halt:/sbin:/sbin/halt
```

图 4-2　用户对应信息

每行记录的信息格式为“用户名:密码:用户 ID:用户组 ID:注释:家目录:解释器 shell”，具体含义如下：

- 用户名：创建新用户名称，后期登录的时候需要输入。
- 密码：一般都是“x”，表示密码的占位。
- 用户 ID：用户的识别符。
- 用户组 ID：用户所属的主组 ID。
- 注释：解释该用户是做什么用的。
- 家目录：用户登录进入系统之后默认的位置。
- 解释器 shell：等待用户进入系统，用户输入命令之后，该解释器会收集用户输入的命令，传递给内核处理。如果解释器是/bin/bash，就表示用户可以登录到系统；如果解释器是/sbin/nologin 就表示该用户不能登录到系统。

4.2.2　添加登录用户

本小节演示在 Linux 环境下如何添加登录用的用户账号。

例如，添加一个名为 harry 的用户，并使用 bash 作为登录的 shell 系统。

```
[root@localhost ~]# useradd harry
[root@localhost ~]# tail -1 /etc/passwd
harry:x:1001:1001::/home/harry:/bin/bash
```

说明：此命令会自动创建harry组，并成为harry用户的默认主组，同时默认的登录shell是bash。

用户账户的全部信息被保存在/etc/passwd 文件中。这个文件以如下格式保存了每一个系统账户的所有信息（字段以“:”分割）。

```
harry:x:1001:1001::/home/harry:/bin/bash
```

- harry：用户名。
- x：密码占位符。
- 1001：用户的 uid，它都是用数字来表示的。
- 1001：用户所属组的 gid，都是用数字来表示的。
- 用户描述信息：对用户的功能或其他进行一个简要的描述。
- /home/harry：用户主目录（shell 提示符中“~”代表的那个）。
- /bin/bash：用户登录系统后使用的 shell。

用如下命令查看系统中支持哪些 shell：

```
[root@localhost ~]# cat /etc/shells    #查看系统中支持哪些 shell
/bin/sh
/bin/bash
/sbin/nologin
/usr/bin/sh
/usr/bin/bash
/usr/sbin/nologin
```

4.2.3 修改用户

语法：

```
#usermod 选项 用户名
```

其中，usermod 表示 user modify，即用户修改。

常用选项：

- -g：表示指定用户的用户主组，选项的值可以是用户组的 id，也可以是组名。
- -G：表示指定用户的用户附加组，选项的值可以是用户组的 id，也可以是组名。
- -u：uid，用户的 id（用户的标识符）。
- -l：修改用户名。
- -c<备注>：修改用户账号的备注文字。
- -d<登入目录>：修改用户登录时的目录。
- -s<shell>：修改用户登录后所使用的 shell。

示例：将 zhangsan 的用户名改为 wangerma。

```
[root@localhost ~]# usermod -l wangerma zhangsan        [新名字在前，旧名字在后]
```

4.2.4　设置密码

Linux 不允许没有密码的用户登录到系统，因此前面创建的用户目前都处于锁定状态，需要设置密码之后才能登录计算机。

语法：#passwd [用户名]（如果不指定用户名则修改自己的密码）

示例：设置 wangerma 用户的密码。

```
[root@localhost ~]# passwd wangerma
Changing password for user wangerma.
New password:
BAD PASSWORD: The password is a palindrome
Retype new password:
passwd: all authentication tokens updated successfully.
```

在设置密码的时候是没有任何输入提示的，请放心输入，但要确保两次输入的密码一致，按回车键即可。

也可以使用弱密码，但是不建议。设置密码之后的 shadow 文件如图 4-3 所示，说明 wangerma 用户是没有密码的。

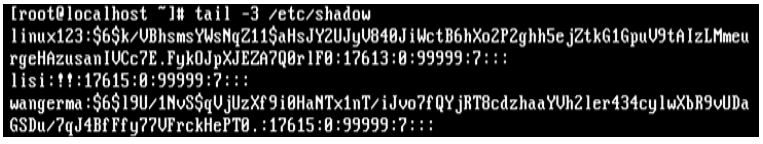


图 4-3　设置密码之后的 shadow 文件

在设置用户密码之后可以登录账号，例如登录 wangerma。

切换用户命令：#su [用户名]（switch user）

如果不指定用户名则表示切换到 root 用户。

例如，切换用户到 zhangsan，如图 4-4 所示。

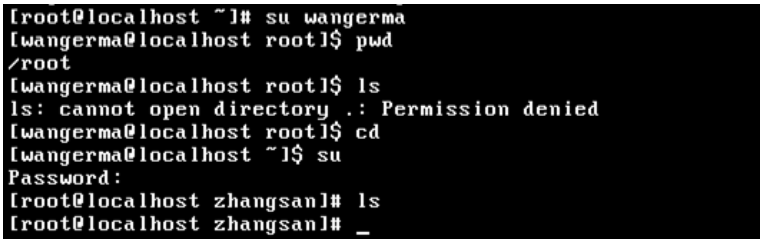


图 4-4　切换用户的操作

切换用户时需要注意以下事项：

- 从 root 向普通用户切换不需要密码，反之则需要 root 密码。
- 切换用户之后前后的工作路径是不变的。
- 普通用户没有办法访问 root 用户目录，反之则可以。

4.2.5 用户密码管理

对于系统上的每个用户账号，如果系统是在生产环境下使用，那么这些账号就有必要设置密码，否则会出现账号被非法使用，危及系统的安全。当然，即使是在测试环境，也建议给账号配置密码，在一定程度上保障系统的安全。

给用户添加密码，使用 password 命令，如图 4-5 所示。

图 4-5 给用户添加密码

在日常使用系统的过程中，有时需要用户定期更改密码，这种变更密码状态的需求由 chage 命令实现：

命令：chage

常用参数 -d：表示上一次更改的日期，如-d 后跟 0 表示强制在下次登录时更新密码。

例如，让用户 Linux 首次登录系统时更改其密码：

```
[root@localhost ~]# chage -d 0 linux
[root@localhost ~]# ifconfig
[root@localhost ~]# ssh  linux@192.168.0.106
...
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.63' (ECDSA) to the list of known hosts.
mk@192.168.1.63's password: 123456
You must change your password now and login again!  # 提示必须改密码
更改用户 linux 的密码 。
为 Linux 更改 STRESS 密码。
（当前）UNIX 密码：
新的密码：
无效的密码：这个密码和原来的相同。
新的密码：
无效的密码：密码与原来的太相同。
新的密码：
重新输入新的密码：
passwd：所有的身份验证令牌已经成功更新。
```

4.2.6 删除用户

语法：#userdel 选项 用户名

Userdel：user delete（用户删除）

常用选项：

-r：删除用户的同时删除其家目录。

示例：删除 test 用户。

```
[root@localhost ~]# userdel -r test
```

如不使用-r 参数，在新建相同用户时会报错，示例如下：

```
[root@test ~]# useradd test
[root@test ~]# ls /home/
test
[root@test ~]# userdel test
[root@test ~]# ls /home/
test
[root@test ~]# useradd test
useradd: warning: the home directory already exists.
Not copying any file from skel directory into it.
Creating mailbox file: File exists
```

解决方案：

为了解决以上问题，我们新建账号 test2 来模拟以上问题的解决过程。

```
[root@test ~]# useradd test2
[root@test ~]# id test2
uid=1001(test2) gid=1001(test2) groups=1001(test2)
[root@test ~]# userdel test2
[root@test ~]# ls /home/
test2
```

删除用户账号后其主目录还存在，因此在创建同名用户账号时就会提示目录已经存在。为了解决这个问题，需要删除一些目录。

```
[root@test ~]# rm -rf /home/test2/
[root@test ~]# rm -rf /var/spool/mail/*
```

删除完毕后，就可以重新创建同名账号了。

```
[root@test ~]# useradd test2
[root@test ~]# id test2
uid=1001(test2) gid=1001(test2) groups=1001(test2)
```

4.3　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，实际上就是对/etc/group 文件的更新。以下为该文件的部分内容：

```
root:x:0:
bin:x:1:
daemon:x:2:
```

```
sys:x:3:
adm:x:4:
tty:x:5:
disk:x:6:
lp:x:7:
mem:x:8:
…
```

文件结构为“用户组名:密码:用户组 ID:组内用户名”。

- 密码：x 表示占位符，虽然用户组可以设置密码，但是绝大部分情况下不设置密码。
- 组内用户名：表示附加组是该组的用户名称。

4.3.1 用户组添加

语法：#groupadd 选项 用户组名
常用选项：

-g：类似用户添加里的“-u”，表示自己设置一个自定义的用户组 id 数字，如果自己不指定，则默认从 1000 之后递增。

示例：使用 groupadd 命令创建一个新的用户组，命名为 test。

```
[root@localhost ~]# groupadd test
```

验证：

```
[root@test ~]# cat /etc/group | grep test3
test3:x:1002:
```

4.3.2 用户组编辑

语法：#groupmod 选项 用户组名
常用选项：

- -g, --gid GROUP: 更新使用者新的起始登入群组，群组名必须已存在。
- -G, --groups GROUP1[,GROUP2,...[,GROUPN]]]: 定义使用者为一堆 groups 的成员。每个群组使用“,”分隔。
- -u, --uid UID: 使用者 id 值，必须为唯一的 id 值。
- -s, --shell SHELL: 指定新登入 shell。如果此栏留白，就将选用系统预设 shell。
- -l, --login NEW_LOGIN: 变更使用者 login 时的名称为 login_name。
- -g: 类似用户修改里的“-u”，表示自己设置一个自定义的用户组 id 数字。
- -n: 类似于用户修改“-l”，表示设置新的用户组名称。

示例：修改 test 用户组，将组 id 改成 1020，将名称改为 admin。

```
[root@test ~]# groupmod -g 1020 -n admin test
```

验证：
使用 vi 编辑器打开用户组的配置文件就可以编辑。

4.3.3　用户组删除

语法：#groupdel 用户组名

示例：删除 admin 组。

为了模拟如何删除用户组，先创建一个名称为 admin 的组，并确定该组已经创建。

```
[root@test ~]# groupadd admin
[root@test ~]# cat /etc/group | grep admin
admin:x:1003:
```

此时可以使用删除组的命令，删除后从/etc/group 文件中清除出去。

```
[root@test ~]# groupdel admin
```

扩展：创建一个用户，并指定该用户的 uid 和组。

先创建两个组 g1 和 g2。

```
[root@test ~]# groupadd g1
[root@test ~]# groupadd g2
```

创建用户 test4，并指定用户的 uid 和组。

```
[root@test ~]# useradd -r -u 520 -s /sbin/nologin -g g1 -Gg2 test4
```

确认该用户的信息是否为指定的信息。

```
[root@test ~]# id test4
uid=520(test4) gid=1003(g1) groups=1003(g1),1004(g2)
```

4.4　系统网络设置

网络功能是操作系统具备的功能之一，也是一个业务系统必须具备的功能。本节对 Linux 系统的网络设置进行介绍，内容涉及网络的基本配置和一些工具的使用，了解这些内容后就能够解决常见的网络问题。

4.4.1　网卡配置文件

网卡配置文件位于/etc/sysconfig/network-scripts 目录下，在该目录下能够找到网卡配置文件及相关的配置文件，包括网卡进程管理文件、路由配置的文件等。这些文件包括普通文本文件、可执行文件和链接文件。

```
    [root@test ~]# cd /etc/sysconfig/network-scripts/
    [root@test network-scripts]# ls
    ifcfg-ens32  ifdown-ipv6   ifdown-TeamPort ifup-ippp   ifup-routes
network-functions
    ifcfg-lo   ifdown-isdn  ifdown-tunnel   ifup-ipv6  ifup-sit
network-functions-ipv6
    ifdown      ifdown-post    ifup           ifup-isdn   ifup-Team
    ifdown-bnep  ifdown-ppp     ifup-aliases     ifup-plip   ifup-TeamPort
    ifdown-eth   ifdown-routes  ifup-bnep       ifup-plusb  ifup-tunnel
```

```
ifdown-ib    ifdown-sit    ifup-eth        ifup-post   ifup-wireless
ifdown-ippp  ifdown-Team   ifup-ib         ifup-ppp    init.ipv6-global
```

其中，普通文本文件中的 ifcfg-ens32 和 ifcfg-lo 是网卡配置文件。要配置系统的网卡，可以对 ifcfg-ens32 文件进行编辑，该文件中的配置信息（这是静态 IP 地址的默认配置信息）如下：

```
TYPE="Ethernet"
BOOTPROTO="none"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_FAILURE_FATAL="no"
IPV6_ADDR_GEN_MODE="stable-privacy"
NAME="ens32"
UUID="a03cadbe-e4d9-4bca-bd69-804660b5bea6"
DEVICE="ens32"
ONBOOT="yes"
IPADDR="10.0.3.104"
PREFIX="24"
GATEWAY="10.0.3.1"
IPV6_PEERDNS="yes"
IPV6_PEERROUTES="yes"
IPV6_PRIVACY="no"
```

下面对该文件中的部分参数进行说明。

- Device：设备名称。
- Type：网络类型，以太网。
- UUID：通用唯一标识符。
- ONBOOT：是否开机启动。
- BOOTPROTO：IP 地址分配方式，DHCP 表示动态主机分配协议。
- HWADDR：硬件地址，MAC 地址。

在更改网卡参数后需要重启才能生效，重启网卡可以使用以下命令：

```
[root@localhost ~]# systemctl restart network
```

查看网卡进程状态：

```
[root@tset ~]# systemctl status network
network.service - LSB: Bring up/down networking
   Loaded: loaded (/etc/rc.d/init.d/network; bad; vendor preset: disabled)
   Active: active (exited) since Fri 2019-12-20 21:59:55 CST; 15min ago
     Docs: man:systemd-sysv-generator(8)
  Process: 795 ExecStart=/etc/rc.d/init.d/network start (code=exited,
status=0/SUCCESS)

Dec 20 21:59:52 tset systemd[1]: Starting LSB: Bring up/down networking...
Dec 20 21:59:55 tset network[795]: Bringing up loopback interface:  [  OK  ]
Dec 20 21:59:55 tset network[795]: Bringing up interface ens32:  [  OK  ]
Dec 20 21:59:55 tset systemd[1]: Started LSB: Bring up/down networking.
```

扩展：如何重启单个网卡？

- 停止某个网卡：#ifdown 网卡名。
- 开启某个网卡：#ifup 网卡名。

例如，停止-启动（重启）ens33 网卡，则可以输入：

```
[root@localhost ~]# ifdown ens33
[root@localhost ~]# ifup ens33
连接已经成功激活 （(DBus 活动路径：/org/freedesktcet/ActiveConnection/21）
```

提示，在实际工作的时候不要随意禁用网卡。

4.4.2　Linux 自有服务——SSH 服务

SSH（secure shell，安全外壳）是一种用于远程连接的工具，有两个常用的作用：远程连接和远程文件传输。

该协议使用的端口号默认是 22（见图 4-6）。如果需要修改，则可以使用如下命令修改 SSH 服务的配置文件：

```
#/etc/ssh/ssh_config
```

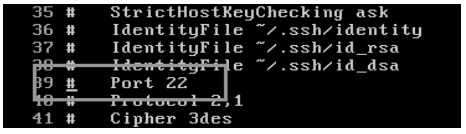


图 4-6　修改 SSH 服务的配置文件

修改端口号时注意以下事项：

（1）端口范围是 0~65535。

（2）不能使用别的服务已经占用的端口（常见的不能使用 20、21、23、25、80、443、3389、3306、11211 等）。

服务启动/停止/重启（服务名中的 d 全称 daemon，守护进程）：

```
[root@localhost ~]# systemctl start sshd
```

sshd 服务一般默认已经启动，可以在修改完其配置的情况下重启。

```
[root@localhost ~]# ps -ef | grep sshd
root      1052     1  0 21:59 ?       00:00:00 /usr/sbin/sshd
root     10236   1052  0 22:11 ?       00:00:00 sshd: root@pts/0
root     10262  10240  0 22:23 pts/0    00:00:00 grep --color=auto sshd
```

4.4.3　远程终端应用

终端工具主要帮助运维人员连接远程的服务器，常见的终端工具有 xshell、putty 等，本节以 putty 为例：

（1）获取服务器 IP 地址，可以通过 ifconfig 命令进行查看，然后顺手测试 IP 的连接相通性。

```
[root@tset ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

```
        inet 127.0.0.1/8 scope host lo
           valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
           valid_lft forever preferred_lft forever
    2: ens32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP qlen 1000
        link/ether 00:0c:29:fd:cd:d3 brd ff:ff:ff:ff:ff:ff
        inet 192.168.0.108/24 brd 192.168.0.255 scope global ens32
           valid_lft forever preferred_lft forever
        inet6 fe80::9117:962c:2d3e:706c/64 scope link
           valid_lft forever preferred_lft forever
```

此处获取的 IP 地址是：

```
[root@localhost ~]# ping 192.168.0.108
```

测试连通性：

```
[root@localhost ~]# ping 192.168.0.108
PING 192.168.0.108 (192.168.0.108) 55(84) bytes of data.
64 bytes from 192.168.0.108: icmp_seq=1 rrl=64 time=0.040 ms
^C
64 bytes from 192.168.0.108: icmp_seq=1 rrl=64 time=0.040 ms
--- 192.168.0.108 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.040/0.040/0.040/0.000 ms
```

（2）打开 putty，输入相关的信息，如图 4-7 所示。

更改 putty 字体大小：https://blog.csdn.net/u012810488/article/details/41597395

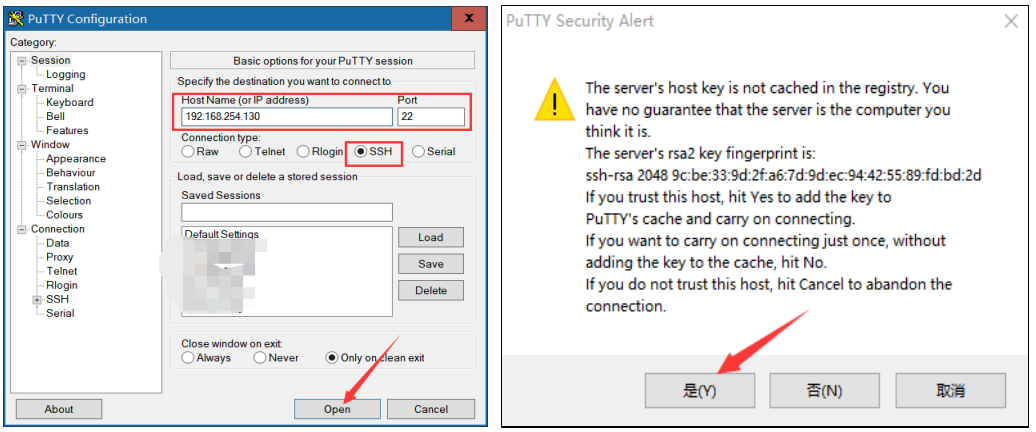


图 4-7　在 putty 工具中输入相关的信息

（3）在弹出 key 确认的对话框时单击“是”按钮，以后不会再提示。

（4）输入登录信息，如图 4-8 所示。

图 4-8　输入登录信息

在虚拟机 CentOS 中的全部命令在远程终端中都可以执行。

4.4.4　Filezilla 工具

FileZilla 是一个免费开源的 FTP 软件，有客户端和服务器端版本，并且具备所有 FTP 软件的功能，具体的特点包括但不限于以下几点：

- 可以断点续传但需要服务器支持。
- 自定义命令。
- 可进行站点管理。
- 防发呆功能（工作界面在一定的时间内没有操作，就会自动断开连接）。
- 超时侦测。
- 支持防火墙。
- 支持 FTP、FTPS（FTP over SSL/TLS）、SFTP（SSH File Transfer Protocol）等多种协议。
- 支持远程文件搜索。

需要使用时可下载安装，安装好的 FileZilla 能够在桌面上看到图标，其工作界面如图 4-9 所示。

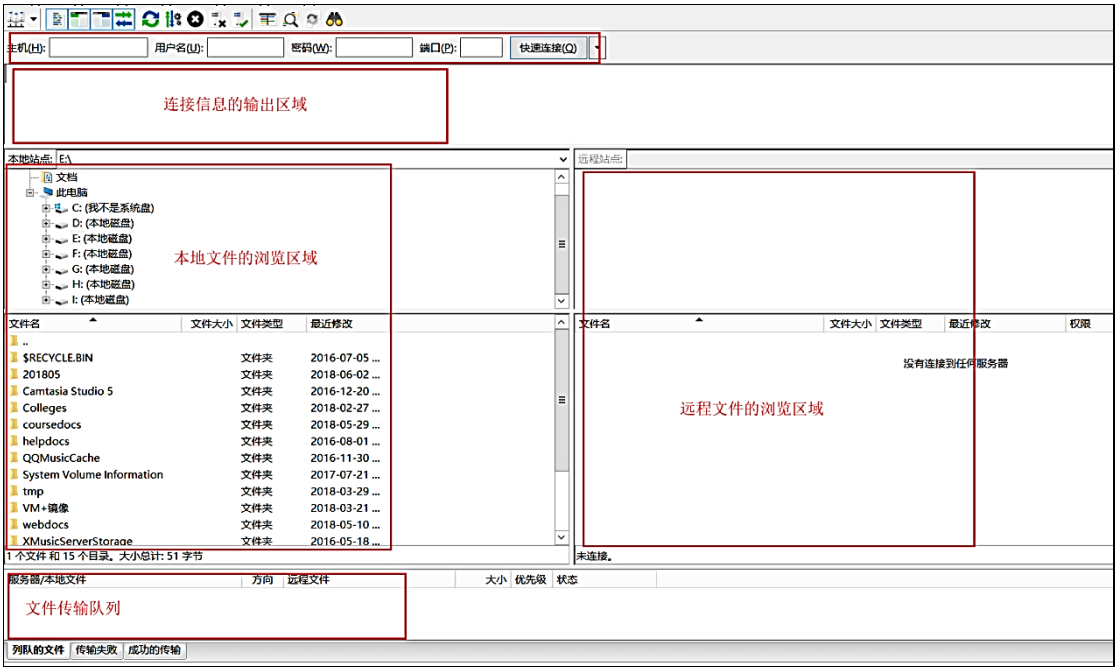


图 4-9　Filezilla 的工作界面

如果使用 Filezilla 远程登录服务器，则可按照以下步骤进行：

（1）选择“文件”→“站点管理器”，打开“站点管理器”对话框，如图 4-10 所示。

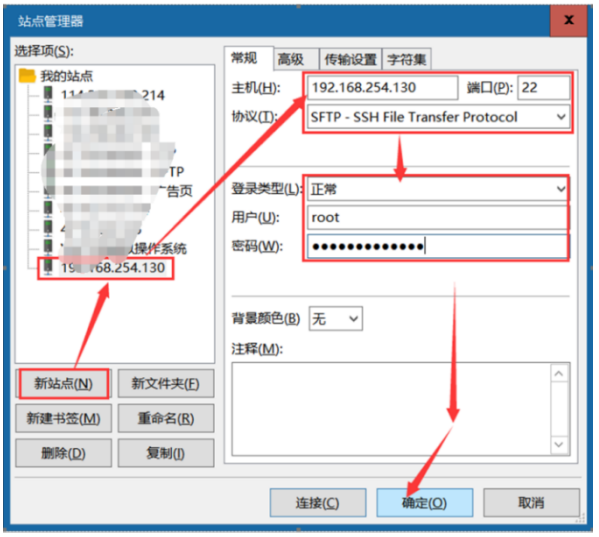


图 4-10　站点管理器

Filezilla 支持保存 IP 地址之类的信息，保存后下次就不必再次输入，如图 4-11 所示。

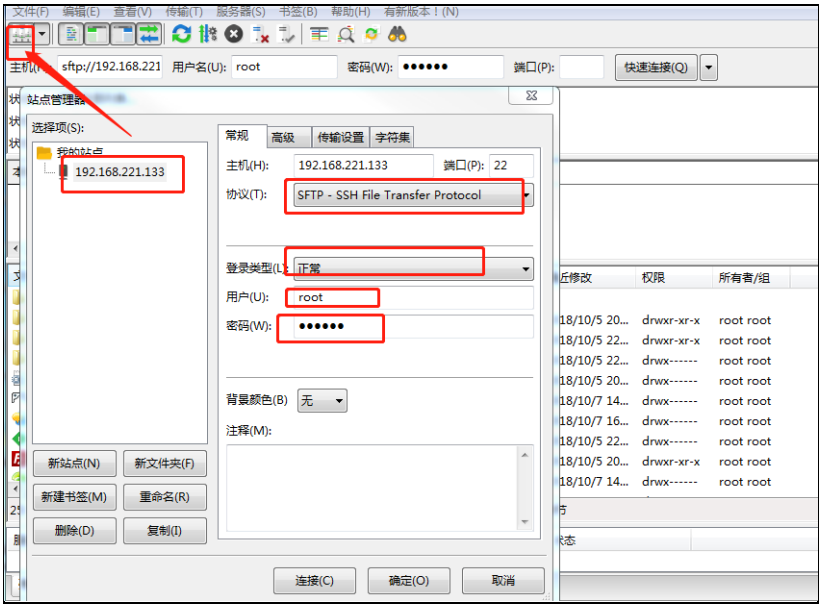


图 4-11　自动保存站点信息

（2）单击“文件”菜单下方的“▽”，选择需要连接的服务器，连接好之后的工作环境如图 4-12 所示。

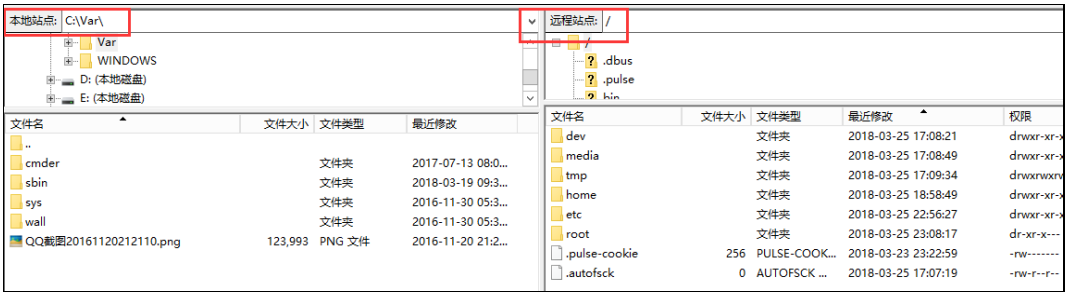


图 4-12　连接服务器的工作环境

（3）从本地 Windows 上传文件到 Linux 中的方式：支持直接拖曳文件，也可以右击本地需要上传的文件，然后选择“上传”按钮，如图 4-13 所示。

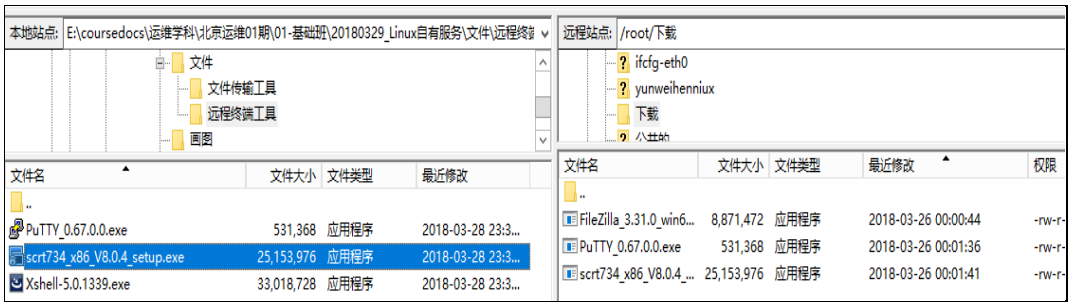


图 4-13　上传的文件

第 5 章

权限管理

权限是 Linux 系统下的一个重要概念，它是保证文件内容安全的另外一道屏障，因此应该对权限的管理有一定的了解。本章主要介绍 Linux 权限管理操作，包括什么是权限、有什么作用和如何设置。

5.1 概 述

Linux 系统一般将文件可存/取访问的身份分为 3 个类别，即 owner（拥有者）、group（和所有者同组的用户）、others（其他人，除了所有者、同组的用户及超级管理员外），并且 3 种身份各有 read（读）、write（写）、execute（执行）的权限。

5.1.1 用户权限介绍

在多用户（可以不同时）计算机系统的管理中，权限是指某个特定的用户具有特定的系统资源的使用权力，比如文件夹、特定系统命令的使用或存储量的限制。

在 Linux 系统中，用户对文件的权限分别有读权限、写权限和执行权限。

- 读权限：对于目录来说，读权限影响用户是否能够进入目录；对于文件来说，读权限影响用户是否可以查看文件内容。
- 写权限：对于目录来说，写权限影响用户是否可以在文件夹下“创建/删除/复制到/移动到”文档；对于文件来说，写权限影响用户是否可以编辑文件内容。
- 执行权限：对于目录来说，执行权限影响用户是否可以执行 cd 操作；对于文件（特别是脚本文件）来说，执行权限影响文件是否可以运行。

5.1.2 用户身份介绍

1. owner 身份（文件所有者，默认为文档的创建者）

由于 Linux 是多用户、多任务的操作系统，因此可能常常有多人同时在某台主机上工作，但是每个人均可在主机上设置文件的权限，让其成为个人的“私密文件”，即个人所有者。因为设置了适当的文件权限，所以除本人（文件所有者）之外的用户无法查看文件内容。

例如，某个 MM 给你发了一封邮件，你将邮件转为文件之后存在自己的主文件夹中。为了不让别人看到邮件的内容，你就能利用所有者的身份去设置文件的适当权限，这样即使别人想偷看内容也是做不到的。

2. group 身份（与文件所有者同组的用户）

与文件所有者同组最有用的功能体现在多个团队在同一台主机上开发资源的时候。例如，主机上有 A、B 两个团体（用户组），A 中有 a1、a2、a3 三个成员，B 中有 b1、b2 两个成员，这两个团体要共同完成一份报告 F。由于设置了适当的权限，A、B 团体中的成员都能互相修改对方的数据，团体 C 的成员则不能修改 F 的内容，甚至连查看的权限都没有。同时，团体的成员也能设置自己的私密文件，让团队的其他成员读取不了文件数据。在 Linux 中，每个账户支持多个用户组，比如用户 a1、b1 既可属于 A 用户组也可属于 B 用户组（主组和附加组）。

3. others 身份（其他人，相对于所有者与同组用户）

这是一个相对概念。打个比方，大明、二明、小明三兄弟同住一栋房，房产证上的登记者是大明（owner），那么大明一家就是一个用户组，这个组有大明、二明、小明三个成员。另外，有个人叫张三，和他们都没有关系，那么张三就是其他人（Others）。大明、二明、小明有各自的房间，三者虽然能自由进出各个房间，但是小明不能让大明看到自己的情书、日记等，这就是文件所有者（用户）的意义。

4. root 用户（超级用户）

在 Linux 系统中，root 用户拥有最大的权限，管理着普通用户。

5.1.3　Linux 的权限介绍

要设置权限，需要知道文件的一些基本属性和权限的分配规则。在 Linux 中，常用 ls 命令来查看文件的属性，例如：

#ls -l 路径（ls -l　等价于 ll）

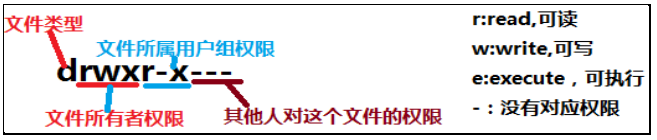
具体示例如图 5-1 所示。

```
[root@localhost ~]# touch {1..3}.txt
[root@localhost ~]# ls -l
总用量 8
-rw-r--r--. 1 root root    0 5月    9 07:39 1.txt
-rw-r--r--. 1 root root    0 5月    9 07:39 2.txt
-rw-r--r--. 1 root root    0 5月    9 07:39 3.txt
-rw-------. 1 root root 1536 5月    7 23:47 anaconda-ks.cfg
-rw-r--r--. 1 root root 1584 5月    8 08:50 initial-setup-ks.cfg
drwxr-xr-x. 2 root root    6 5月    8 21:40 公共
drwxr-xr-x. 2 root root    6 5月    8 21:40 模板
drwxr-xr-x. 2 root root    6 5月    8 21:40 视频
drwxr-xr-x. 2 root root    6 5月    8 21:40 图片
drwxr-xr-x. 2 root root    6 5月    8 21:40 文档
```

图 5-1　显示文件信息

下面对 Linux 的文档权限属性信息组成进行简单介绍。

Linux 中存在用户（owner）、用户组（group）和其他人（others）的概念，各自有不同的权限。对于一个文档来说，其权限具体分配如图 5-2 所示。



d（第 1 位）	rwx（第 2~4 位）	r-x（第 5~7 位）	---（第 8~10 位）	user1	user1	time	FILENAME
文件类型	拥有者的权限	所属组的权限	其他人的权限	拥有者	属组	最后修改时间	对象

图 5-2　文档权限分配

文档权限有 10 位字符，其含义如下：

- 第 1 位：表示文档类型，取值常见的有“d”（表示文件夹）、“-”（表示文件）、“l”（表示软连接）、“s”（表示套接字）、“c”（表示字符设备）、“b”（表示块状设备）等。
- 第 2~4 位：表示文档所有者的权限情况。第 2 位表示读权限的情况，取值有“r”“-”；第 3 位表示写权限的情况，“w”表示可写，“-”表示不可写；第 4 位表示执行权限的情况，取值有“x”和“-”。
- 第 5~7 位：表示与所有者同在一个组的用户的权限情况。第 5 位表示读权限的情况，取值有“r”“-”；第 6 位表示写权限的情况，“w”表示可写，“-”表示不可写；第 7 位表示执行权限的情况，取值有“x”和“-”。
- 第 8~10 位：表示除了前两部分的用户之外的其他用户的权限情况。第 8 位表示读权限的情况，取值有“r”和“-”；第 9 位表示写权限的情况，“w”表示可写，“-”表示不可写；第 10 位表示执行权限的情况，取值有“x”和“-”。

下面可以通过查看/dev/目录下文件的权限组成来进行深入了解。

```
[root@yunwei ~]# ll /dev/
总用量 0
crw-rw----.    1  root video    10, 175 10 月 14 09:09 agpgart
crw-rw----.    1  root root     10,  57 10 月 14 09:09 utofs
…
```

可见，权限分配均是由 r、w、x 三个参数组合实现，且位置顺序不会变化，如果没有对应权限就用“–”代替。

例如，有一个文档，权限分配情况如下：

```
drwxr-xr-x.  2  root root  6  Nov  5  2016  /media/
```

该文档类型是文件夹，所有者拥有读写执行权限，同组用户拥有读执行权限，其他人拥有读执行权限，管理员拥有全部权限。

其他文档权限的解读如下：

- - rwx --- ---：文件所有者对文件具有读取、写入和执行的权限。
- - rwx r-- r--：文件所有者具有读、写与执行的权限，用户组里的用户及其他用户具有读取的权限。
- - rw- rw- r-x：文件所有者与同组用户对文件具有读写的权限，其他用户仅具有读取和执

行的权限。

- drwx--x--x：目录所有者具有读写与进入目录的权限，其他用户只能进入该目录却无法读取任何数据。
- drwx------：除了目录所有者具有完整的权限之外，其他用户对该目录完全没有任何权限。

当前是什么用户，在该用户下创建文件或目录时就自动将所属主设置为该用户所有，下面通过例子来说明。如图 5-3 所示是在两个系统上做一样的操作，即使用相同的用户创建相同的目录和文件，以此来验证文件的权限所属是系统默认，而不因系统不同而改变。为了演示本次操作，分别在两个系统上以 root 用户创建 yw 用户，并分别以这两个用户各创建一个文件，以此来对文件的所属主进行对比。

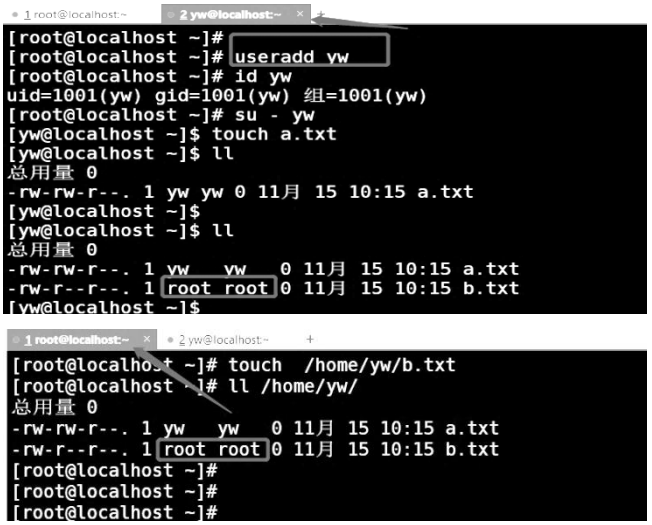


图 5-3　权限分配示例

以上图是分别在不同的系统上操作截取的，在两个不同的系统上以 root 用户先创建 yw 用户，然后使用 root 创建 b.txt 文件，yw 用户创建 a.txt 文件。最后在两个不同的系统上将文件的权限做对比，发现它们所属的属主是相同的。这就说明，权限是系统根据用户的级别并在创建文件时默认分配的属主。

通过下面的例子来了解文件权限的组成。

```
dr-xr-xr-x.   2  root root   12288   8 月   5  00:00  sbin
drwxr-xr-x.   7  root root       0   8 月   11  08:54  selinux
```

- 文件的类型：d，表示是一个目录。
- 所有者：r 和 x 表示读和执行权限，-表示没有权限。
- 同组用户：r 和 x 表示读和执行权限，-表示没有权限。
- 其他人：r 和 x 表示读和执行权限，-表示没有权限。

如果文件的权限是 rwx，说明该用户对文件具有可读、可写和可执行权。

5.2 权限的设置

本节来学习权限的设置、如何给用户权限和如何取消权限。我们将通过数字和字母两种形式来讲解。

语法：#chmod 选项 权限模式 文档

- 选项：-R，递归设置权限（当文档类型为文件夹的时候）。
- 权限模式：就是该文档需要设置的权限信息。
- 文档：既可以是文件也可以是文件夹，既可以是相对路径也可以是绝对路径。

注意，如果想要给文档设置权限，那么操作者要么是 root 用户要么是文档的所有者。

5.2.1 字母形式的权限

在权限设置中既有用字符形式也有用数字形式，如果是字符形式，其涉及的字符主要有如图 5-4 所示的几个。

选项	字母	介绍
(谁)	u	用户
(谁)	g	所属群体
(谁)	o	其他人
(谁)	a	所有人 (“全部”)
(作用)	+	增加权限
(作用)	-	减少权限
(作用)	=	确定权限
(权限)	r	可读
(权限)	w	可写
(权限)	x	执行

图 5-4 权限设置字符

这些字符按用户群体、作用、权限分为以下 3 类。

（1）用户群体字符

- u：表示所有者身份 owner（user）。
- g：表示给所有者同组用户（group）设置权限。
- o：表示 others，给其他用户设置权限。
- a：表示 all，给所有人（包含 ugo 部分）设置权限。

如果在设置权限的时候不指定给谁设置，那么默认给所有用户设置。

（2）权限字符

- r：表示读。
- w：表示写。
- x：表示执行。
- -：表示没有权限。

（3）权限分配方式字符

- +：表示给具体的用户新增权限（相对当前）。
- -：表示删除用户的权限（相对当前）。
- =：表示将权限设置成具体的值（注重结果），即赋值。

例如，需要给/root/anaconda-ks.cfg 文件（-rw-------.）设置权限，要求所有者拥有全部权限（rwx），同组用户拥有读和写权限（rw），其他用户拥有只读权限（r）。

```
 [root@localhost ~]# chmod u+x,g+rw,o+r anaconda-ks.cfg
-rw------.  1  root  root  1716  10月  30  06:37 anaconda-ks.cfg
```

还原：

```
[root@localhost ~]# chmod u=rw,g-rwx,o-rwx anaconda-ks.cfg
-rw------.  1  root  root  1716  10月  30  06:37 anaconda-ks.cfg
```

提示，当文档拥有执行权限（任意部分）时，则其颜色在终端中是绿色的。

如果有两部分权限一样则可以合在一起写，例如“#chmod ug=rwx”等价于“#chmod u=rwx,g=rwx”。

如果 anaconda-ks.cfg 文件什么权限都没有，那么可以使用 root 用户设置所有人都有执行权限：

```
#chmod a+x anaconda-ks.cfg    等价于  #chmod +x anaconda-ks.cfg
#chmod a=x anaconda-ks.cfg
#chmod ugo=x anaconda-ks.cfg
#chmod u+x,g+x,o+x anaconda-ks.cfg
```

设置文件“~/yunwei/yunwei.txt”的权限，要求所有者拥有全部权限，同组用户拥有读写权限，其他人拥有读权限，如图 5-5 所示。

```
[root@localhost ~]# mkdir yunwei
[root@localhost ~]# touch yunwei/yunwei.txt
[root@localhost ~]# ll yunwei/yunwei.txt
-rw-r--r--. 1 root root 0 11月 15 10:39 yunwei/yunwei.txt
[root@localhost ~]# chmod u=rwx,g=rw,o=r yunwei/yunwei.txt
[root@localhost ~]# ll  yunwei/yunwei.txt
-rwxrw-r--. 1 root root 0 11月 15 10:39 yunwei/yunwei.txt
```

图 5-5　设置权限

将其他人的权限更改为读写权限，其他用户权限不变：

```
[root@localhost ~]# ll yunwei/yunwei1.txt
-rwxrw-r---.  1  root  root  0  11月  15  10:39  yunwei/yunwei1.txt
[root@localhost ~]# chmod o+w yunwei/yunwei1.txt
-rwxrw-rw--.  1  root  root  0  11月  15  10:39  yunwei/yunwei1.txt
```

设置文件夹/tmp/yunwei 的权限（如果文件夹不存在就自行创建），要求权限为递归权限，并且所有者拥有全部权限，同组用户拥有读执行权限，其他用户拥有只读权限；

```
[root@yunwei ~]# chmod -R u=rwx,g=rx,o=r yunwei/
```

设置文件/tmp/yunwei/class03.txt 权限（文件如果不存在就自行创建），要求权限为所有者拥有全部权限，同组用户拥有读和执行权限，其他用户没有权限。

```
[root@localhost ~]# touch yunwei/class03.txt
-rw-r--r--.  1  root  root  0  11月  15  10:43  yunwei/class03.txt
```

```
[root@localhost ~]# chmod u=rwx,g=rx,o-r ./yunwei/class03.txt
-rwxr-r---.  1  root  root  0  11月  15  10:43  yunwei/class03.txt
```

5.2.2 数字形式

经常会在一些技术性的网页上看到类似于“#chmod 777 a.tx”这样的权限，这种称为数字形式的权限（777）。其与字符形式的权限效果一致，各数字与字符有一定的对应关系，如下所示：

- 读：r，4。
- 写：w，2。
- 执行：x，1。
- 没有任何权限：0，---。

具体如图 5-6 所示。

数值	权限	目录列表
0	不能读，不能写，不能执行	---
1	不能读，不能写，可执行	--x
2	不能读，可写，不能执行	-w-
3	不能读，可写，可执行	-wx
4	可读，不能写，不能执行	r--
5	可读，不能写，可执行	r-x
6	可读，可写，不能执行	rw-
7	可读，可写，可执行	rwx

图 5-6 各数字对应的权限

例如，需要给 anaconda-ks.cfg 设置权限，要求所有者拥有全部权限，同组用户拥有读执行权限，其他用户拥有只读权限。

- 所有者权限=全部权限=读权限+写权限+执行权限=4 +2 +1=7。
- 同组用户权限=读权限+执行权限=4+1=5。
- 其他用户权限=读权限=4。

最终得出的结果是 754，具体权限设置命令如下：

```
[root@localhost ~]# chmod 754 anaconda-ks.cfg
-rwxr-xr---.  1  root  root  1716  10月  30  06:37 anaconda-ks.cfg
```

面 试 题

用超级管理员设置文档的权限命令是“#chmod -R 731 aaa”，请问这个命令有没有什么不合理的地方？

解答：

所有者 = 7 = 4 + 2 + 1 = 读 + 写 + 执行

同组用户 = 3 = 2 + 1 = 写 + 执行

其他用户 = 1 = 执行

在权限 731 中，3 表示“写+执行”权限，但是必须要打开之后才可以写，因此必须具备读权限，这里的权限不合理。

注意，单独出现 2、3 的权限数字一般都是有问题的权限。在写权限的时候千万不要设置类似于上面的这种“奇葩权限”。

在 Linux 中，如果要删除一个文件，不是看文件有没有对应的权限，而是看文件所在的目录是否有写权限，如果有才可以删除。

5.3　属主与属组设置

本节主要学习属主和属组的设置，在学习之前大家一定要清楚两者的区别：

- 属主：所属的用户（文件的主人），文档所有者。
- 属组：所属的用户组。

在图 5-7 中，前面的 root 是属主，后面的 root 是属组。这两项信息在文档创建的时候会使用创建者的信息（用户名放在前面的 root 中、用户所属的属组名称放在后面的 root 中）。这样设置的原因是，有时候删除某个用户，则该用户对应的文档属主和属组信息也需要修改（类似离职之前的工作交接），以达到关联修改的目的。

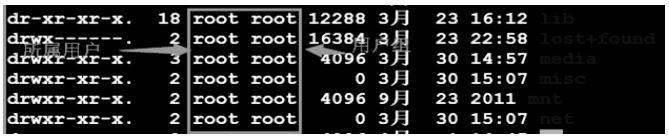


图 5-7　属主与属组的区别

5.3.1　chown 命令应用

chown 命令的作用是更改文档的所属用户（change owner），语法如下：

```
#chown  -R  新的 username 文档路径
```

其中，-R 表示选项，文件不需要-R，目录需要加-R。

示例 1：更改目录的所有者。

有一个原目录，如图 5-8 所示。

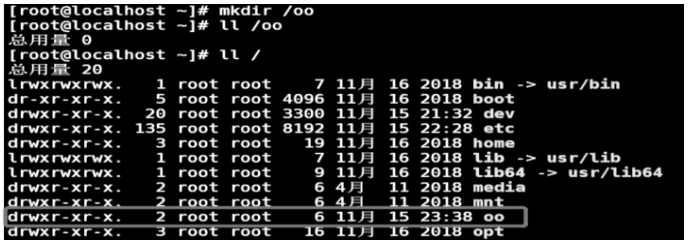


图 5-8　原目录

执行下述命令更改所有者：

```
# chown -R yw03 /oo/
```

结果如图 5-9 所示。

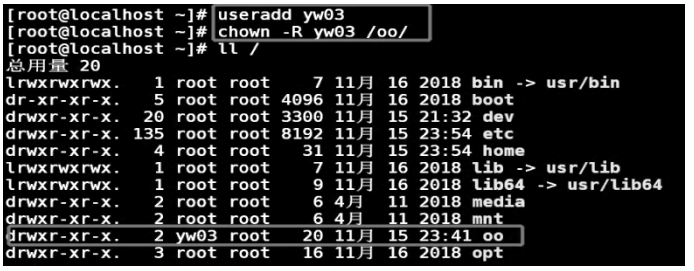


图 5-9　更改结果

示例 2：更改 hr，san 用户的属主和属组。

chown user:group filename　比如：chown hr:san a.txt　把文件的属主和属组改为 hr,san

chown user filename　比如：chown san a.txt　把文件的属主改为 san 用户

chown :group filename　比如：chown :miao a.txt　把文件的属组改为 miao 这个组

chown user: filename　比如：chown san: a.txt　自动继承这个用户所有的组

示例 3：文件组关系的演变。

下面通过三个文件来演示文件的组关系的转变过程。首先，创建 a.txt、b.txt 和 c.txt 三个文件：

```
[root@test ~]# ll *.txt
-rw-r--r-- 1 root root 0 Dec 20 17:26 a.txt
-rw-r--r-- 1 root root 0 Dec 20 17:26 b.txt
-rw-r--r-- 1 root root 0 Dec 20 17:26 c.txt
```

然后创建用户 yw，并确认用户已经成功创建。

```
[root@test ~]# useradd yw
[root@test ~]# cat /etc/passwd | grep yw
yw:x:1002:1006::/home/yw1:/bin/bash
```

接着进行的用户和用户组变更操作。先更改 a.txt 文件的所有者（由 root 更改为 yw）。

```
[root@test ~]# chown yw a.txt
[root@test ~]# ll a.txt
-rw-r--r-- 1 yw root 0 Dec 20 17:26 a.txt
```

再更改文件的用户组（将 b.txt 文件的用户组更改为 yw）。

```
[root@test ~]# chown :yw b.txt
[root@test ~]# ll b.txt
-rw-r--r-- 1 root yw 0 Dec 20 17:26 b.txt
```

最后更改文件的所有者和用户组（将 c.txt 文件的所有者和用户组更改为 yw）。

```
[root@test ~]# chown yw:yw c.txt
[root@test ~]# ll c.txt
-rw-r--r-- 1 yw yw 0 Dec 20 17:26 c.txt
```

5.3.2　chgrp 命令应用

chgrp 命令的作用是更改文档的所属用户组（change gro up），语法如下：

```
#chgrp  -R  groupname  文档的路径
```

例如，将上一小节的 oo 目录改为 yw03（见图 5-10）：

```
[root@localhost ~]# chgrp -R yw03 /oo/
```

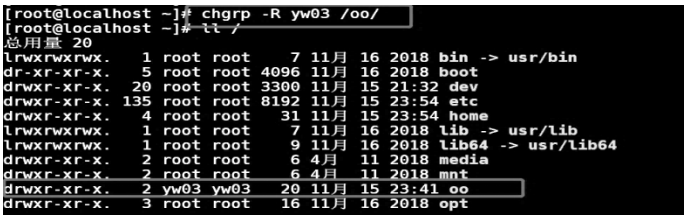


图 5-10　将 oo 目录更改为 yw03

前面所讲的 chown 命令既可以更改所属的用户，也可以用于修改所属的用户组，语法如下：

```
#chown  -R  username:groupname   文档路径
```

例如，将 oo 文档所属组的所属用户修改为 root：

```
#chown -R root:root /oo
```

执行结果如图 5-11 所示。

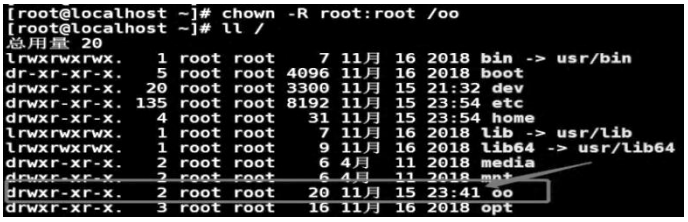


图 5-11　执行结果

5.4　文件的特殊权限

在 Linux 系统中，常见的权限是读、写和可执行，其实文件与目录可设置的权限不止这些，还有一种特殊的权限，即 suid 和 sgid，下面分别进行介绍。

5.4.1　suid（set uid，设置用户 ID）权限应用

suid 的功能：程序运行时的权限从执行者变更成程序所有者。

suid 一般用在二进制可执行文件上，当用户执行该文件时，会临时拥有该执行文件的所有者权限，但对目录无效。使用“ls –l”或者“ll”命令浏览文件时，如果可执行文件所有者权限的第三位是一个小写的“s”，就表明该执行文件拥有 suid 权限。比如，/usr/bin/passwd 文件就有“s”的权限参数：

```
[root@test ~]# ll /usr/bin/passwd
-rwsr-xr-x. 1 root root 27832 Jun 10  2014 /usr/bin/passwd
```

示例：提升到用户权限。

本示例是针对“s”权限基本应用的介绍，目的就是通过对文件授予“s”权限后再使用不同的用户来执行文件，以验证拥有该权限的文件是否能被其他非所属用户拥有临时权限。以下是普通用户 yw 和 root 用户用 less 命令查看文件的内容，并对比添加“s”权限先后的操作结果，如图 5-12 所示。

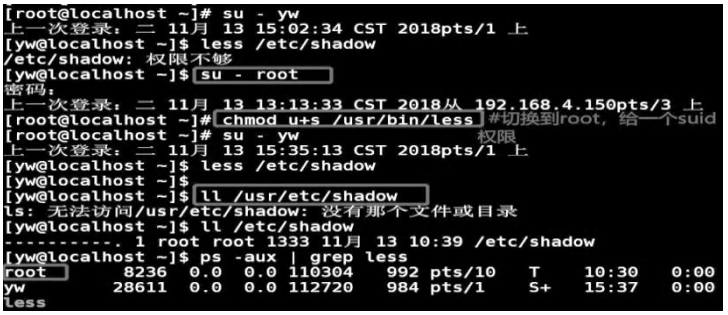


图 5-12　权限提升

从命令执行的先后顺序来看，先从 root 切换到 yw 用户，再使用 less 命令查看/etc/shadow 文件的内容，最终发现 yw 用户对该 less 命令没有查看内容的权限。

重新切换回 root 用户，给 less 命令授予“s”权限（/usr/bin/less 文件），添加后再次切换到 yw 用户并执行 less 命令查看/etc/shadow 文件的内容，这次执行的结果显示的是空。这就意味着 yw 用户已经能够执行 less 命令，看不到内容则与 yw 对/etc/shadow 文件的权限有关。

使用 ll 命令查看/etc/shadow 文件的权限组成，就会发现该文件中的权限参数全都是“-”，也就是说 yw 用户没有查看该文件内容的权限，因此 less 命令执行后没有获取到相关内容。当然，因文件大小为 1333KB，所以肯定是有内容的。

5.4.2　sgid 权限应用

sgid 权限的功能：该权限既可以对二进制可执行程序进行设置，也可以对目录进行设置。在设置了 sgid 权限的目录下建立文件时，新创建的文件所属组会继承上级目录的所属组。

为了验证上述说法，下面通过实例来演示。图 5-13 演示的是给目录所属组添加“s”权限，并验证目录所属组被添加“s”后在其下所创建文件自动添加“s”权限的问题。

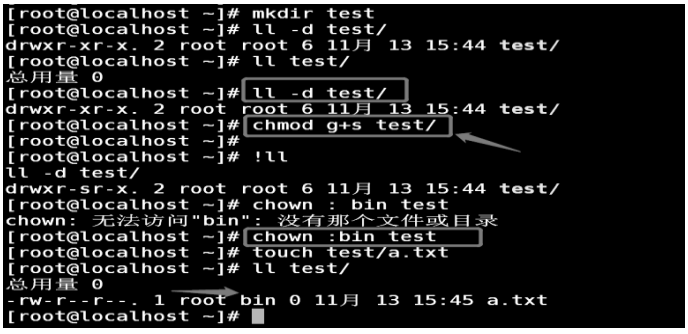


图 5-13　目录所属组 s 权限应用

这里首先创建一个目录 test 并确定该目录 test 的权限组成，通过带有-d 选项的 ll 命令输出的信息可以确定没有“s”权限。

接着执行 chmod 命令给 test 目录添加“s”权限，添加后确认一下，之后使用 chown 命令更改 test 目录的所属组为 bin。至此，准备工作已经完成，然后验证在拥有“s”权限的目录下创建文件时该文件是否集成其父目录中所属组 bin 组。最后使用 touch 命令在 test 目录下新建 a.txt 文件，并使用 ll 命令查看新建的文件，能够看到该文件继承了其父目录中的所属组。

5.4.3　案例：文件扩展权限 acl

acl（access control list，访问控制列表）主要用于针对某些目录或文件授予某个用户 rwx 权限的细节设置。

示例 1：设置用户 yw 对文件 a.txt 拥有 rwx 权限；yw 不属于 a.txt 的所属主和组，是 other。

先以 root 身份创建文件 a.txt：

```
[root@localhost ~]# touch /tmp/a.txt
```

再通过 getfacl 命令查看 a.txt 的权限（拥有者和属组都是 root）：

```
[root@localhost ~]# getfacl /tmp/a.txt
```

然后使用 setfacl 命令让 yw 用户对 a.txt 拥有 rwx 权限：

```
[root@localhost ~]# setfacl -m u:yw:rwx /tmp/a.txt
```

其中，-m 表示修改文件或目录的扩展 acl 设置信息；u 表示设置某个用户拥有的权限。再次查看 acl 权限，yw 用户已经有 a.txt 文件的 rwx 权限了，如图 5-14 所示。

```
[root@localhost ~]# touch /tmp/a.txt
[root@localhost ~]# getfacl /tmp/a.txt
getfacl: Removing leading '/' from absolute path names
# file: tmp/a.txt
# owner: root
# group: root
user::rw-
group::---
other::---

[root@localhost ~]# setfacl -m u:yw:rwx /tmp/a.txt
[root@localhost ~]# getfacl /tmp/a.txt
getfacl: Removing leading '/' from absolute path names
# file: tmp/a.txt
# owner: root
# group: root
user::rw-
user:yw:rwx
group::---
mask::rwx
other::---
```

图 5-14　yw 用户对 a.txt 文件具备 rwx 权限

切换到 yw 用户，发现其具备编辑并且查看 a.txt 的权限，如图 5-15 所示。

```
[root@localhost ~]# su - yw
[yw@localhost ~]$ echo "aaaaaaaaaa">>/tmp/a.txt
[yw@localhost ~]$ echo "bbbbbbbbb">>/tmp/a.txt
[yw@localhost ~]$ cat -n /tmp/a.txt
     1  aaaaaaaaaa
     2  bbbbbbbbb
[yw@localhost ~]$
```

图 5-15　yw 用户对 a.txt 文件有编辑和查看权限

示例 2：给目录下所有文件都加扩展权限。注意，-R 选项（操作递归到所有子目录和文件）一定要在-m 前面或者授权目录的前面（否则会报语法错误），表示目录下所有文件。

```
    [root@localhost ~]#  setfacl -R -m
u:yw:rw- testdirectory/或者 setfacl  -m
u:lee:rw- -R testdirectory/
    [root@localhost ~]#  setfacl -x u:yw
/tmp/a.txt       # 去掉单个权限
    [root@localhost ~]#  setfacl -b
/tmp/a.txt           # 去掉所有 acl 权限
```

在图 5-16 中，先给 a.txt 文件分别设置了 readonly、work、yw 三个用户的 acl 权限，然后演示去掉单个 yw 用户的 acl 权限的效果，接着演示去掉 yw、work、readonly 三个用户的 acl 权限的效果。

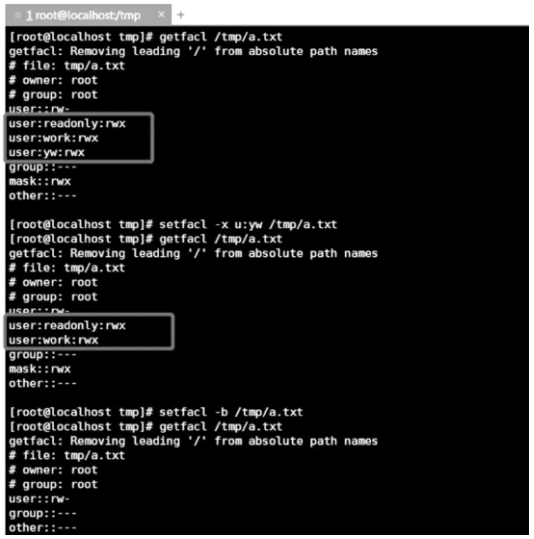


图 5-16　去掉单个 acl 权限和去掉所有 acl 权限

5.5　实战 sudo 命令

普通用户身份无法操作 reboot、shutdown、init、halt、user 这些管理命令，但某些特殊情况下又需要有执行权限，此时可以使用 sudo（switch user do）命令来进行权限设置。sudo 可以让管理员（root）事先定义某些特殊命令谁可以执行。

默认情况下，sudo 命令的配置文件中仅配置 root 用户的规则，规则所定义的参数在/etc/sudoers 这个配置文件中，该文件也是 sudo 的配置文件。配置文件默认为只读（不允许修改），但可以使用 cat 等命令来查看其内容，如下是部分内容：

```
    …
    ## Allow root to run any commands anywhere
    root    ALL=(ALL)       ALL

    ## Allows members of the 'sys' group to run networking, software,
    ## service management apps and more.
    # %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES,
LOCATE, DRIVERS

    ## Allows people in group wheel to run all commands
    %wheel  ALL=(ALL)       ALL

    ## Same thing without a password
    # %wheel        ALL=(ALL)       NOPASSWD: ALL

    ## Allows members of the users group to mount and unmount the
    ## cdrom as root
    # %users  ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom

    ## Allows members of the users group to shutdown this system
    # %users  localhost=/sbin/shutdown -h now
```

```
## Read drop-in files from /etc/sudoers.d (the # here does not mean a comment)
#includedir /etc/sudoers.d
```

对于该配置文件的修改，建议使用 visudo 命令，不过用户对该文件仅有可读权，因此要编辑该文件须先授予可写权，并在打开该文件后使用和 vim 编辑器一样的命令来操作。

之前讲过，/etc/sudoers 默认情况下仅配置 root 用户的规则，对于规则的配置格式可以在该文件中找到，以下是配置格式和信息：

```
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root    ALL=(ALL)       ALL
```

下面按照先后顺序对这些信息进行说明：

- root：表示用户名，如果是用户组，则可以写成“%组名”。
- ALL：表示允许登录的主机（地址白名单）。
- (ALL)：表示以谁的身份执行，ALL 表示 root 身份。
- ALL：表示当前用户可以执行的命令，多个命令可以使用“,”分隔。

如果需要配置其他用户的 sudo 规则，可参考 root 用户的配置规则进行。

示例：创建 test 用户，并以 test 用户的身份来新建用户。

```
[root@localhost ~]# useradd test
[root@localhost ~]# passwd test
更改用户 test 的密码
新的 密码：                  # 密码为 123456
无效的密码： 密码少于 8 个字符
重新输入新的 密码：
passwd：所有的身份验证令牌已经成功更新。
```

实际上，平台用户 test 并不能添加用户，这是因为它没有创建用户的权限，因此使用它来创建 test1 用户时会提示权限不够的问题。

```
[root@localhost ~]# su - test
[root@localhost ~]# useradd test1
-bash: /user/sbin/useradd: 权限不够
```

要是需要使用普通用户 test 来创建用户，需要在 sudo 的配置文件中进行配置，使其具备添加新用户的权限。

注意，在写 sudo 规则的时候不推荐以相对路径的方式设置命令，建议以绝对路径的方式设置（直接指定命令的完整路径）。

路径可以使用 which 命令来查看，语法如下：

```
#which 命令名称
```

接着在 visudo 打开的文件中配置普通用户的权限，配置信息如下：

```
test    ALL+(ALL)           /usr/sbin/useradd
```

在添加好对应的规则之后，就可以切换到普通用户 test，然后再执行：

```
[root@yunwei /]# su - test
```

```
[test@yunwei /]# useradd test1
bash: /user/sbin/useradd: 权限不够
```

此时要想使用刚才的规则，则以以下命令进行：

```
#sudo 需要执行的命令
[root@localhost ~]# sudo useadd test1
[sudo] test 的密码：
```

在输入 sudo 命令之后需要输入当前的用户密码进行确认操作（不是 root 用户密码），输入之后在接下来的 5 分钟内再次执行 sudo 命令则不需要密码。

如果要删除用户则会提示：

```
[test@localhost ~]# sudo userdel test1
对不起,用户 test 无权以 root 的身份在 localhost.localdomian 上执行/sbin/userdel test1
```

因此，要想实现删除则必须先配置 sudo 规则，即打开/etc/sudoers 配置文件或直接执行 visudo 命令打开配置文件（建议使用 visudo 命令打开），并在该文件中以绝对路径的方式增加命令/usr/sbin/userdel，添加如下行：

```
test   ALL=(ALL)       /usr/sbin/useradd, /usr/sbin/userdel
```

保存后退出，此时使用 test 用户来删除 test1 用户就不会提示无权限的问题了。

扩展：在普通用户下可以使用下述命令查看自己具有哪些特殊权限。

```
#sudo  -l            表示 list
```

已经添加 sudo 权限后的 test 普通用户可以查看自己具有的特殊权限，如图 5-17 所示。

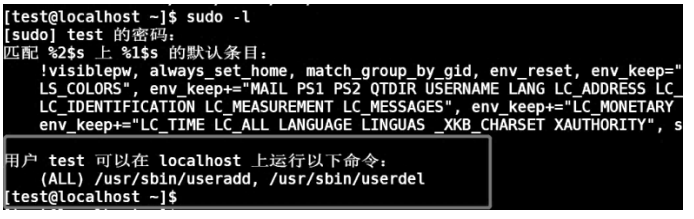


图 5-17　普通用户查看自己的特殊权限

注意，sudo 不是任何 Linux 系统都有的命令，但常用的 CentOS 与 Ubuntu 都有 sudo 命令。

第 6 章

文件归档

本章我们开始学习文件的归档处理，因为 Linux 系统都是以文件的形式存在的，所以在处理文件时有时会因为文件太多导致传输速度慢等问题，为了方便使用并提高效率，常把文件归档（就是把多个文件变成一个归档文件）。

6.1 文件的类型

在 Linux 系统下，常见的文件类型主要有目录文件、普通文件、链接文件等，它们相互协作以完成各项工作，使系统能够正常工作，本节我们针对 Linux 这些文件进行详细介绍。

6.1.1 目录文件

形象地说，目录更像是一个数据库，在其下存储着文本文件、设备文件及目录文件等各类文件。目录是至今为止依然不能直接写入数据的文件，因此要在目录下存储数据时需要先创建普通文件，再以向普通文件写入数据的形式存储数据。

在 Linux 系统下的目录以层级的等级关系组成一种“分布式”的结构，这种“分布式”的结构就像是一棵倒置的树，树的最顶层是根目录（用“/”来表示），其下的子目录可以一直向下延伸，理论上没有终点。

文件是组成 Linux 系统的主要成分，是系统不可缺少的。其中，目录是文件体系中的基础，系统的文件以目录为依托而存在。

根（/）目录是整个 Linux 系统最顶层的目录，是系统其他文件存在的基础和入口，因此根目录是一个非常重要的目录。系统以根目录为起点，并与其他目录或文件形成一个层次等级严格的文件系统，不过默认情况下与根目录有直接关系的这层等级中只存在目录（根目录的子目录），可以在根目录下执行 tree 命令来获取根目录与子目录之间这种严格的等级关系图。

6.1.2 普通文件

在 Linux 系统下，普通文件主要包含文本文件和特殊文件这两类。普通文件是系统中数量最多、使用频率最高的文件，是系统中相关数据存放的最终归宿。

1. 文本文件

文本文件以“-”作为标识符，包括纯文本文件（ASCII，其内容可以直接读取，如自编的 shell 脚本文件）、二进制文件（这些都经过编译且内容不可读）和数据格式的文件（在程序运行中使用特定格式的文件）这三类。

其中，纯文本文件是 Linux 系统中数量最多的一种类型，之所以称为纯文本文件，是因为可以直接读懂它的内容；二进制文件主要以可执行的状态存在，是一些经过特殊处理的文件，处理过后的内容仅有系统能读懂，系统大多数的命令其实对应的就是一个二进制文件；数据格式文件是系统中某些程序在运行的过程中会读取的某些特定格式的文件，这些文件以不可读的方式来记录数据，但所记录的数据可以通过特定的命令来读取，如使用 last 命令读取/var/log/wtmp 的内容。

2. 特殊文件

特殊文件包括块设备文件、套接字文件、字符设备文件及命名管道文件，这类文件不包含任何数据，而是提供一种用于在文件系统中建立一个物理设备与文件名之间映射的机制，系统必须支持每个设备且每个设备至少与一个特殊文件相关联。

特殊文件是通过 mknod 命令或系统调用来创建的，在创建这类文件时必须提供相应的驱动程序并在创建后集成到系统内核中。特殊文件被 Linux 系统用于作为用户与 I/O 设备间的接口，使用户能够像读写普通文件一样使用设备 I/O 接口。

- 字符设备文件：这类文件以“c”作为标识符，是一些如鼠标、键盘等串口设备的接口，这些设备通常是一次性读取且不能截断输出。
- 块设备文件：块设备文件以“b”作为标识符，是文件系统的组成单位，是储存数据以供系统随机存取的接口设备。
- 套接字文件：或称数据接口文件，以 s 作为标识符。这种文件通常用于网络上的数据承接，具体说就是应用程序用于给客户端进行数据的沟通。
- 命名管道文件：或称数据输送文件，用于解决多个程序同时存取一个文件所造成的错误问题，以“p”为标识符。

块设备文件、套接字文件等与我们日常工作的内容交集较少，仅做展示。例如，进入/dev 目录后，会看到如下文件：

```
[root@localhost ~]# ls -la /dev/tty
crw-rw-rw- 1 root tty 5, 0 04-19 08:29 /dev/tty
[root@localhost ~]# ls -la /dev/hda1
brw-r----- 1 root disk 3, 1 2006-04-19 /dev/hda1
```

其中，/dev/tty 的展示结果为 crw-rw-rw-，第一个字母“c”表示字符设备文件，如猫等串口设备；/dev/hda1 的展示结果为 brw-r-----，第一个字母“b”表示块设备，如硬盘、光驱等。

还有一种特殊的文件，就是/dev/null，是被称为“黑洞”的空设备文件，任何进入到该设备的数据都会被“吞并”，基本读不出来，可以说是有进无出。

6.1.3　链接文件

在 Linux 系统中，链接文件有软、硬链接两种。其中，软链接文件相当于一个文件的影子，比如快捷方式文件；硬链接文件相当于一个文件的副本，就好比是对一个文件进行复制后得到的文件，但这并不是独立存在的文件。

软/硬链接文件有各自的特点，下面对这两种文件的特点进行简单介绍。

（1）文件创建后，源文件、软/硬链接文件均可以查看到文件内容。

（2）编辑源文件，软/硬链接文件的内容也会随着变化。

（3）删除源文件，软链接失效，硬链接无影响。再重新建一个与源文件同名的文件，软链接就直接链接到新的文件，而硬链接不变，因为软链接是按名称进行链接的。

软链接文件可以在一定的程度上解决配置文件管理的问题，比如说有些服务有多个分布在不同位置的配置文件，通过软链接方式就可以对这些文件做一个软链接文件到统一的目录下，能够在很大程度上减轻查找文件所需的时间。

软链接文件就好比是 Windows 系统下的快捷文件，在 Windows 系统下创建快捷文件时需要先选择目标文件（创建快捷文件的源文件）再创建快捷文件。在 Linux 系统下要创建软链接文件，同样需要先指定源文件后才能创建软链接文件。

创建软链接文件使用的是 ln 命令且常带-s 选项，软链接文件的名称可以自定义。下面是创建/a/source.txt 文件的软链接文件/b/des.txt 的过程。

```
[root@localhost ~]# ln -s /a/source.txt /b/des.txt
[root@localhost ~]# ll /b/des.txt
Lrwxrwxrwx  root root  13  10 月   8  19:04  /b/des.txt -> /a/source.txt
```

这种软链接文件的源文件和目标文件内容上是一致的。为了验证这个结论，对目标文件/b/des.txt 进行编辑后保存，接着查看源文件/a/source.txt 的内容，会发现源文件的内容就是刚改过的/b/des.txt 文件的内容。也就是说，这类文件的数据是自动同步的，更改其中一个文件的内容相当于同时更改两个文件的内容。

同理，要对源文件/a/source.txt 的内容进行更改，目标文件/b/des.txt 的内容也会被同步成与源文件的内容一致。因此，从本质上可以把这类文件看作是一个文件。

源文件相当于是“根基”所在，把它删除后目标文件因找不到“根基”其内容也不存在了，因此目标文件仅是以一个名字存在而没有任何实质性的内容。反过来讲，删除目标文件就相当于删除文件的影子，对源文件并不会造成丝毫影响。

以上内容通过下面的例子来验证。直接删除目标文件并查看源文件：

```
[root@localhost ~]# rm -rf /b/des.txt
[root@localhost ~]# ll /a/source.txt
-rw-r--r--.  1  root  root  25  10 月  8  19:18   /a/source.txt
```

此时发现源文件还存在，内容还是没有发生改变，因此可以说删除目标文件不会对源文件造成实质性的影响。

反过来，删除源文件时尽管目标文件没有被删除，但此时目标文件只是以名字的形式存在，实际上已经没有内容了。

```
[root@localhost ~]# rm -rf /a/source.txt
[root@localhost ~]# ll /b/des.txt
Lrwxrwxrwx. 1 root root 13 10月 8 19:21 /b/des.txt —> /a/source.txt
```

以上示例输出的信息以很显眼的颜色显示，如果查看它的内容，就会提示文件不存在。

6.2 文件归档和归档技术

文件归档就是把多个文件变成一个归档文件，集中在一起，归档的目的是为了方便备份、还原及文件的传输操作，简单讲就是便于管理。

6.2.1 用 tar 命令归档文件

对文件的归档打包管理使用的是 tar 命令（工具），格式是：

```
tar cf 存储路径 打包文档
```

tar 命令的功能是将多个文件（也可能包括目录，因为目录本身也是文件）放在一起，存放到一个磁带或磁盘归档文件中，并且将来可以根据需要只还原归档文件中某些指定的文件。

tar 命令有以下几个常用选项：

- c: 创建一个新的 tar 文件。
- t: 列出 tar 文件中目录的内容。
- x: 从 tar 文件中抽取文件。
- f: 指定归档文件或磁带（也可能是软盘）设备（一般都要选）。
- v: 显示所打包的文件的详细信息，v 是 verbose 的第 1 个字母。
- z: 使用 gzip 压缩算法来压缩打包后的文件。
- j: 使用 bzip2 压缩算法来压缩打包后的文件。

例如，需要将"/etc"下的文件归档则需要输入"tar cf etc.tar /etc"。

示例：归档文件。为便于操作，先创建目录 test 并创建 1~5 这 5 个 txt 文件。

```
[root@localhost ~]# mkdir test
[root@test ~]# cd test/
[root@test test]# touch {1..5}.txt
[root@test test]# ll
total 0
-rw-r--r-- 1 root root 0 Jan 10 17:02 1.txt
-rw-r--r-- 1 root root 0 Jan 10 17:02 2.txt
-rw-r--r-- 1 root root 0 Jan 10 17:02 3.txt
-rw-r--r-- 1 root root 0 Jan 10 17:02 4.txt
-rw-r--r-- 1 root root 0 Jan 10 17:02 5.txt
```

使用 tar 命令归档文件：

```
[root@test test]# tar -vcf test.tar 1.txt 2.txt 3.txt 4.txt 5.txt
1.txt
2.txt
3.txt
4.txt
5.txt
```

其中，test.tar 是最终归档后的文件的名字，所有的.txt 后缀的文件都是被归档的文件，命令中使用的 v、c 和 f 三个选项分别表示输出详细信息、创建归档文件和指定要归档的文件。

此时，查看当前目录会发现多出了一个 test.tar 的归档文件。

```
[root@localhost test]# ls
1.txt  2.txt  3.txt  4.txt  5.txt  test.tar
```

实际上，通过这种方式进行归档后，归档文件会占用更多的磁盘空间。本来磁盘空间资源就有限，因此如何将存放在系统上的数据在保证数据完整性的条件下减少占据磁盘空间是一个非常重要的问题。

对于 tar 命令，它有一个选项是可以对归档文件进行压缩的，这个选项就是-z。在归档时使用该选项能够对归档文件进行压缩，示例如下：

```
[root@test test]# tar -zvcf test2.tar 1.txt 2.txt 3.txt 4.txt 5.txt
1.txt
2.txt
3.txt
4.txt
5.txt
[root@test test]# ll -h
total 16K
-rw-r--r-- 1 root root    0  Jan 10 17:02 1.txt
-rw-r--r-- 1 root root    0  Jan 10 17:02 2.txt
-rw-r--r-- 1 root root    0  Jan 10 17:02 3.txt
-rw-r--r-- 1 root root    0  Jan 10 17:02 4.txt
-rw-r--r-- 1 root root    0  Jan 10 17:02 5.txt
-rw-r--r-- 1 root root  132  Jan 10 17:51 test2.tar
-rw-r--r-- 1 root root  10K  Jan 10 17:03 test.tar
```

其中，-h 选项是以 K 单位的形式归档文件大小的。

对比是否使用-z 选项后的归档文件就会发现，使用-z 选项后的归档文件大小都不到 1K，远比不使用-z 选项归档的文件小很多，这样就能够节省大量的磁盘空间。

6.2.2　解压 tar 格式归档文件

对于.tar（.tar.gz）后缀归档的（解压）文件，使用带有选项的 tar 命令就可以解压，常用的选项及功能说明如下：

- -x: 从 tar 文件中抽取文件。
- -f: 指定归档文件或磁带（也可能是软盘）设备（一般都要选）。
- -c: 表示指定目录。

为了能够更加直接地反映出解压的效果，先将当前目录（/root/test/）下所有以.txt 为后缀的文

件全部删除（其中，*符号是通配符，意思就是匹配当前目录下所有符合条件的文件），之后通过 ll 命令查看删除结果。

```
[root@test test]# rm -rf *.txt
[root@test test]# ll
total 16
-rw-r--r-- 1 root root   132 Jan 10 17:51 test2.tar
-rw-r--r-- 1 root root 10240 Jan 10 17:03 test.tar
```

根据输出的信息可以确定所有的.txt 文件已经被删除，接下来开始对压缩文件进行解压，所解压的文件是 test2.txt，并查看操作结果。执行下面的命令：

```
[root@test test]# tar -vxf test2.tar
1.txt
2.txt
3.txt
4.txt
5.txt
[root@test test]# ll
total 16
-rw-r--r-- 1 root root     0 Jan 10 17:02 1.txt
-rw-r--r-- 1 root root     0 Jan 10 17:02 2.txt
-rw-r--r-- 1 root root     0 Jan 10 17:02 3.txt
-rw-r--r-- 1 root root     0 Jan 10 17:02 4.txt
-rw-r--r-- 1 root root     0 Jan 10 17:02 5.txt
-rw-r--r-- 1 root root   132 Jan 10 17:51 test2.tar
-rw-r--r-- 1 root root 10240 Jan 10 17:03 test.tar
```

通过输出的信息可以确定，test2.tar 文件已经被解压。

6.2.3 压缩/解压缩多种格式文件

对于压缩文件，除了.tar、.gz 格式之外，还有 zip、gz 及 xz 等格式，对于这类（压缩）归档文件，通常被使用的概率比较少，但在系统迁移、数据还原等场合常使用，因此有必要了解这类文件的压缩和解压缩过程。

接下来将介绍 gz 和 bz2 这两种格式文件的压缩和解压缩，首先介绍.gz 格式文件的压缩和解压缩。

为了演示对文件的压缩和解压缩操作，先做一下准备工作，即使用 cat 命令把/etc/passwd 文件的内容复制一份出来并重命名文件为 test.txt（其中的>用于重定向，相当于新建一个文件，也具备清空文件内容的功能），然后使用 gzip 命令对它进行压缩。

```
 [root@test ~]# cat /etc/passwd > test.txt
[root@test ~]# gzip test.txt
[root@test ~]# ll
total 4
-rw-r--r-- 1 root root 960 Jan 10 20:50 test.txt
[root@test ~]# gzip test.txt
[root@test ~]# ll
total 4
-rw-r--r-- 1 root root 448 Jan 10 20:50 test.txt.gz
```

从输出的信息来看，确实出现了 gz 格式的文件，但源文件已经不存在了，说明该命令在压缩文件的时候就把源文件删除了。另外，压缩文件比源文件小一些（源文件为 960，压缩后为 448），有利于数据的存储。

对于 gz 格式文件的解压缩，其命令为 gunzip，且在执行解压缩时不需要选项的辅助，因此使用也是比较简单的。需要注意的是，解压缩时会把源文件删除。对此类压缩文件的解压缩过程如下：

```
[root@test ~]# gunzip test.txt.gz
[root@test ~]# ll
total 4
-rw-r--r-- 1 root root 960 Jan 10 20:50 test.txt
```

上面介绍的是.gz 类压缩文件的操作，接下来介绍.bz2 类格式的文件的压缩与解压的。这类文件的压缩使用的命令是 bzip2，比较简单。

```
[root@test ~]# bzip2 test.txt
[root@test ~]# ll test.txt.bz2
-rw-r--r-- 1 root root 713 Jan 10 20:50 test.txt.bz2
```

从输出的信息可以确定，bz2 格式的压缩文件已经生成，仅从现在的信息看该命令对文件的压缩率并没有 gzip 命令的压缩率高。

对于 bz2 类格式的压缩文件，要解压缩时需要使用 bunzip2 命令。下面对 test.txt.bz2 压缩文件进行解压缩，并在解压后查看该文件的信息。

```
[root@test ~]# bunzip2 test.txt.bz2
[root@test ~]# ll test.txt
-rw-r--r-- 1 root root 1621 Jan 10 21:50 test.txt
```

第 7 章

磁盘空间管理

系统中的数据主要储存在磁盘空间上，每块磁盘通常被划分成一个或以上的分区，每个分区都是磁盘中独立的空间，在这些分区上需要创建文件系统后才可以储存数据，也就是说在磁盘上先创建分区，再创建文件系统，每层之间协同来保证数据的安全。

7.1 磁盘分区的概念

硬盘属于随机存储设备，比较常见的接口可归纳为串行接口和并行接口这两大类。由于硬盘的类型不同，因此在系统下命名的方式也不同。通过命名的组成就可以判断出硬盘的类型，在 Linux 系统下硬盘通常是以“/dev/sd+字母”的方式命名，并以数字来表示分区的数目。

7.1.1 硬盘的物理结构

磁盘是一片以坚硬金属材料制成并被永久密封在固定位置上的磁性介质，通常每个盘有两面并且每面都可记录信息。要了解磁盘的物理结构，就需要对磁道、扇区、柱面等这几个概念有所了解。注意，磁盘只是硬盘的一部分，不同容量硬盘的盘片数也有所不同。

磁盘是计算机最为宝贵的资源之一，计算机及相关应用所产生的数据绝大部分都被永久保存在磁盘的盘片之上。对于磁盘的盘片，可以视每个盘片由若干个磁道和扇区组成，这些磁道和扇区在盘片的表面空间上形成多个同心圆柱面，在读写数据时需要磁盘片不断高速旋转，如果数据具有一定的量，数据的查找时间就消耗更多。也就是说，磁盘的转速和数据储存量都会对数据的读写速度和系统运行的速度有所影响。

磁盘中的磁道、扇区等都具有一定的空间限制，这些限制可以为磁盘中数据的正常读写提供可行性，否则会引起磁道间的相互干扰。

- 磁道：磁盘片被划分为等分的同心圆，这些同心圆就是磁道。磁道是被磁头磁化的同心

圆，为了避免磁盘单元之间相互干扰，每个磁道之间都要存在一定的间隔。

- 扇区：磁盘的磁道中等份的区域，是数据存储的最小单位。
- 柱面：具有相同磁道编号的同心圆组成的面。柱面是磁盘分区的最小单位，数量与磁盘上的磁道数相等。

7.1.2　分区的基本组成方式

分区是指在磁盘上建立用于储存数据和文件的独立空间，要了解磁盘分区的结构，需要对 MBR、主分区、扩展分区、逻辑分区的相关概念有所掌握。系统下的磁盘分区（安装系统的磁盘，不包括后来新添加的磁盘）由主分区（Primary Partition）和扩展分区（Extended Partition）组成。其中，主分区可直接使用，扩展分区则需要先划分成逻辑分区（Logical Partition）才可以使用。

对于磁盘的分区，其划分的组合方式有多种，但不管怎么划分都要至少存在一个主分区。对于扩展分区，可以不划分，由于扩展分区不能直接使用，因此需要在扩展分区上再创建逻辑分区，理论上逻辑分区的数量是没有上限的。

每个磁盘上最多有 4 个主分区（整个磁盘都划分成主分区），实际上不建议将整个磁盘划分成 4 个主分区（由于磁盘本身的原因，划分成 4 个主分区并不能把全部的空间都使用到，而且剩下的空间也没有多余的分区表来记录），这会造成空间的浪费。

实际上扩展分区并不能直接使用，而是需要在其上再创建逻辑卷才可以。那么何必要先创建扩展分区再创建逻辑卷而不是直接创建逻辑卷呢？主要是因为在扩展分区上创建分区时不受限制，即可以创建无限个逻辑分区。

除了主分区和扩展分区之外，系统会自动保留一个只有 512 字节大小的主引导（Master Boot Recorder，MBR）分区（图 7-1 所示的是 MBR 的位置及结构示意图）。这个分区是实际文件数据放置的地方，保存着磁盘系统启动的引导信息、磁盘分区表等重要信息。这些数据非常重要，如果 MBR 物理实体损坏，那么这块磁盘基本无法使用了。

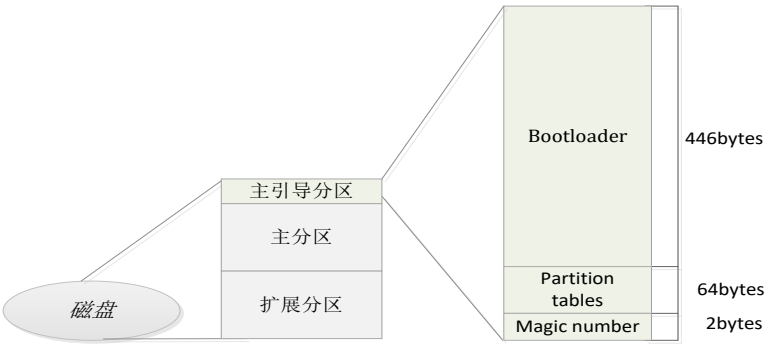


图 7-1　磁盘分区

主引导分区中主要有 Bootloader 和 Partition tables 这两个部分：Bootloader 占据 446 字节，用于存放引导代码；Partition tables 占据 64 字节，用于存放磁盘分区表，因为磁盘每个分区的信息需要用 16 个字节来记录，所以最多只能记录 4 个分区的信息。

另外，在每个分区表中记录着每个分区的大小（始终点）、所处磁盘的位置、柱面等信息。如果重新分区，实际上就是重新更改分区表的记录信息，比如分区表中定义了第 n 个分区是从“第

x 个柱面到第 y 个柱面”，当系统要读取第 n 个磁盘时就根据分区表中定义的信息去操作。

7.1.3 磁盘分区的命名规则

在 Linux 系统下有 SCSI 和 IDE 这两种类型的硬盘，其中，SCSI 是以/dev/sdx 格式命名的，IDE 是以/dev/hdx（x 表示的是硬盘的块数，比如第 1 块为 a，则第 2 块则为 b）的格式命名。相对来说，SCSI 类硬盘使用较为广泛。

硬盘中的磁盘空间虽然可以划分成多个分区，但是分区之间不能重叠（也就是分区号不能相同），而且每个分区的名称总与该分区所属的（磁盘）位置有关（如/dev/sda1 表示第一块磁盘的第一个分区），而对于不同类型的硬盘它所支持的分区数量也有所不同。磁盘设备（包括 U 盘、光盘等）在系统下都被视为文件系统并被映射到/dev/目录下。不同类型的设备被映射后的名称存在差异，可通过表 7-1 来说明。

表7-1 不同设备映射后的名称

设 备 名	相关说明
/dev/hdc	第三块 IDE 磁盘
/dev/sda	第一块 SCSI 磁盘
/dev/sda1	第一块 SCSI 磁盘上的第一个分区
/dev/sda2	第一块 SCSI 磁盘上的第二个分区
/dev/sda3	第一块 SCSI 磁盘上的第三个分区
/dev/sda5	第一块 SCSI 磁盘上的第一个逻辑分区

磁盘的分区编号等于或超过 5（如/dev/sda5、/dev/sda6 等）时就说明这块磁盘一定存在逻辑分区，如/dev/sda5 表示第一块 SCSI 类磁盘上的第一个逻辑分区，/dev/sda6 则表示第二块逻辑分区。

另外，在 Linux 主机上做 XEN 或 KVM 虚拟化时，这些虚拟机所用磁盘的命名格式也有所不同。XEN 类虚拟机的磁盘格式采用的是/dev/xvdb 的命名方式，KVM 则采用的是/dev/vdb 的命名方式，光盘采用的是/dev/sr0 的命名方式。

7.2 使用 fdisk 管理分区

在实际工作环境下对磁盘分区的划分通常是把整块新增的磁盘划分为一个分区，并对磁盘进行格式化。本节就从磁盘分区的获取、分区的创建以及磁盘挂载和卸载几个方面介绍硬盘分区的管理。

7.2.1 获取磁盘分区信息

要对系统的磁盘空间进行维护，首先要做的就是获取磁盘空间的相关信息。要获取怎么样的信息应该根据实际的维护需要而定（例如要扩展磁盘空间则应先了解是哪个分区空间不足，分区是否支持扩容，即分区的文件系统类型是否为 Linux LVM），通过对信息的获取就可以对需要维护的磁盘空间进行维护。

对于系统中的磁盘分区，最基本的应该有根（/）分区、/boot 分区和/swap 分区。根分区是一个特殊的分区，是系统根目录所在的区，储存系统正常运行时所需的数据；/boot 分区储存的是用于引导系统启动时所需的数据（如果这些数据被损坏，那么系统将无法正常启动）；swap 分区是磁盘上一个用于暂存数据的虚拟内存区，所储存的数据是从物理内存中调出来的，并在系统需要时再调出该区。

在系统上要获取磁盘有多少块，可以在/dev/目录下或通过 fdisk 命令来查看。如果只是获取磁盘的块数和每块磁盘上有多少个分区，可以通过下面的 ll 命令来查看，比如查看在/dev/下 SCSI 类磁盘和其分区：

```
[root@system ~]# ll /dev/sd*
brw-rw---- 1 root disk 8,  0 May  9 22:05  /dev/sda
brw-rw---- 1 root disk 8,  1 May  9 22:05  /dev/sda1
brw-rw---- 1 root disk 8,  2 May  9 22:05  /dev/sda2
brw-rw---- 1 root disk 8, 16 May  9 22:05  /dev/sdb
brw-rw---- 1 root disk 8, 32 May  9 22:05  /dev/sdc
```

从输出的结果可以看到，当前系统已挂载有 sda、sdb 和 sdc 三块硬盘，其中 sda 磁盘上有两个分区，即 sda1 和 sda2。

要获取某个磁盘更为详细的信息，可以执行 fdisk 命令，这个命令支持对指定或输出系统所有磁盘的信息。例如：

```
[root@system ~]# fdisk -l /dev/sda

Disk /dev/sda: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00024798

   Device  Boot     Start     End     Blocks     Id   System
/dev/sda1   *       1       26     204800    83   Linux
/dev/sda2          26       1306    10279936  8e   Linux LVM
```

使用带有-l 选项的 fdisk 命令查看磁盘/dev/sda 的信息，可以获取关于/dev/sda 磁盘两个部分的信息：前半部分的内容包括磁盘的总容量、头部容量、I/O 速度和它在系统上的唯一标识码；后半部分的内容是磁盘分区的相关信息，包括磁盘的分区数、分区的起止位置、分区的总容量和分区的类型等。

对于系统当前的磁盘中的各个分区，在系统运行的过程中会因数据不断增加而消耗磁盘资源，此时要想了解分区的使用情况，可以通过 df 命令来获取相关的信息（选项不同，输出的单位不同）。例如：

```
[root@system ~]# df -h
Filesystem          Size   Used  Avail  Use%  Mounted on
/dev/mapper/vg_cent6-LogVol01
                    8.7G   1.2G  7.1G   15%   /
tmpfs               244M    0    244M   0%   /dev/shm
/dev/sda1           194M   25M  159M  14%  /boot
```

输出的信息包括文件系统、文件系统的容量、已使用的容量、空闲容量、使用率和挂载点这几个部分。

其中，根分区总容量为 8.7GB（加上自身损耗那就是 9GB），其使用 1.2GB（使用率为 15%）。如果存在多个分区，就可以看到其他分区的相关信息。

要知道系统中已经挂载的分区及相关的信息（如分区名称、挂载点、分区的文件系统类型等），可以查看/etc/fstab 文件中的配置内容。例如：

```
# /etc/fstab
# Created by anaconda on Tue May  6 21:48:15 2014
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
/dev/mapper/vg_cent6-LogVol01  /                ext4  defaults         1 1
UUID=3b4ff1ef-46c4-4b74-ac8e-8e6dc2990e4f /boot     ext4   defaults
1 2
/dev/mapper/vg_cent6-LogVol00 swap           swap  defaults          0 0
tmpfs                     /dev/shm          tmpfs  defaults          0 0
devpts                    /dev/pts          devptsgid=5,mode=620    0 0
sysfs                     /sys              sysfs  defaults          0 0
proc                      /proc          proc  defaults         0 0
```

在/etc/fstab 文件中记录的是系统当前所挂载且开机自动挂载的磁盘分区、挂载点、文件系统类型等信息。

下面对该文件中设置自动挂载的配置格式进行简单介绍。

第一段参数：设备的名称（或被挂载的设备），所指的设备可以是逻辑卷、逻辑卷和特殊的文件系统。

第二段参数：挂载点，就是某个设备要挂载到的地方（或目录）。

第三段参数：文件系统的类型，是在格式化分区时指定的。

第四段参数：指定挂载后谁有权限使用，可以指定某个用户或组员使用或拥有读写权限等参数，要是使用默认的方式就是开放使用权。

第五段参数：启动时是否备份，其中 0 表示不做备份，1 表示每天进行备份，2 表示不定期备份。

第六段参数：启动是否检查扇区，其中 0 表示不检查，1 表示启动时就检查，2 表示在 1 级别后进行检查。

要获取系统中存在的物理卷、卷组、逻辑卷及相关信息（如这些卷的名称、路径及相关的信息等），可以通过相应的命令来对系统中这些卷的信息进行扫描并显示出来。

以下使用 pvscan 命令扫描系统中存在的物理卷：

```
[root@system ~]# pvscan
  PV /dev/sdb1   VG vgsdb    lvm2 [50.00 GiB / 0    free]
  PV /dev/sda2   VG vg_sam   lvm2 [7.80 GiB / 0    free]
  Total: 2 [57.80 GiB] / in use: 2 [57.80 GiB] / in no VG: 0 [0   ]
```

通过输出的信息可获知到物理机所在的分区、名称、容量及物理卷的总容量等。

以下使用 vgscan 命令扫描系统中存在的卷组：

```
[root@system ~]# vgscan
  Reading all physical volumes.  This may take a while...
  Found volume group "vgsdb" using metadata type lvm2
  Found volume group "vg_sam" using metadata type lvm2
```

通过输出的信息可获知，系统中的卷组名称及分区的文件系统类型。

以下使用 lvscan 命令扫描系统中存在的逻辑卷：

```
[root@system ~]# lvscan
  ACTIVE            '/dev/vgsdb/lvsdb' [50.00 GiB] inherit
  ACTIVE            '/dev/vg_sam/LogVol01' [6.80 GiB] inherit
  ACTIVE            '/dev/vg_sam/LogVol00' [1.00 GiB] inherit
```

通过输出的信息可获知系统中的逻辑卷名称及各自的容量。

7.2.2　创建磁盘分区

在磁盘上创建分区可使用 fdisk 命令进入交互模式后划分，如需要将新添加的磁盘/dev/sdc 划分出一个大小为 5GB（大概为 652 cylinders）的新分区/dev/sdc1，该分区为主分区且分区号为 1。可以执行下述命令：

```
[root@system ~]# fdisk /dev/sdc
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0x79a40d2d.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
        switch off the mode (command 'c') and change display units to
        sectors (command 'u').

Command (m for help): n         # n（new）表示创建新分区
Command action
   e   extended
   p   primary partition (1-4)
p                       # 指定分区为主分区
Partition number (1-4): 1       # 分区号为 1
First cylinder (1-1305, default 1):     # 分区的起点，默认为从第一个柱面开始
Using default value 1
#分区的终止位置
Last cylinder, +cylinders or +size{K,M,G} (1-1305, default 1305): 652

Command (m for help): p     # 显示新分区信息

Disk /dev/sdc: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x79a40d2d
```

```
   Device   Boot    Start     End     Blocks      Id  System
/dev/sdc1            1       652     5237158+    83  Linux

Command (m for help): w     # 保存分区信息
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

至此，已经完成在磁盘/dev/sdc 上创建新的分区/dev/sdc1，如果是把整个磁盘划分成一个分区就不需要指定分区的终止柱面，使用默认的参数即可（也就是在设定分区的终止柱面时直接按“回车”键）。

下面是在交互模式下通过 m 命令获取的命令及相关信息。

- a: toggle a bootable flag（设置可引导的标记）。
- b: edit bsd disklabel（修改 bsd 磁盘的标签）。
- c: toggle the dos compatibility flag（设置 dos 的兼容性）。
- d: delete a partition（删除一个分区）。
- l: list known partition types（列出被支持的所有分区类型）。
- m: print this menu（显示帮助列表信息）。
- n: add a new partition（添加一个新的分区）。
- o: create a new empty DOS partition table（创建一个新的空 DOS 分区表）。
- p: print the partition table（显示当前的分区列表）。
- q: quit without saving changes（退出操作但不保存修改）。
- s: create a new empty Sun disklabel（创建一个新的空 Sun 磁盘标签）。
- t: change a partition's system id（更改分区的系统 id 号）。
- u: change display/entry units（更改显示记录）。
- v: verify the partition table（对分区表进行核实）。
- w: write table to disk and exit（保存新的分区表参数后退出）。
- x: extra functionality (experts only)（特殊功能，建议初学者不要用）。

7.2.3 分区卸载报错解决方案

对于磁盘分区，在需要维护时需要把它们卸载下来，不过有时候在卸载磁盘分区时会出现错误，其中的错误之一就是无法卸载，这种情况一般是提示设备忙。

为了模拟过程，先执行命令把分区挂载，并在切换到挂载点后在其他终端窗口上执行 umount 命令来卸载分区：

```
[root@localhost ]# mount /dev/sdb1 /sdb1/
[root@localhost ~]# cd /sdb1/
[root@localhost ~]# umount /sdb1
umount: /sdb1：目标忙
```

通过输出的信息可知道被卸载的目标设备处于忙碌状态，简单地说就是该设备在被使用中无

法停止，也就无法被卸载。这种情况就好比是在 Windows 下要卸载 U 盘时提示设备在使用。

如果要卸载该设备，那么首先要找到使用该设备的进程，并将它退出。要查找是谁在使用某个设备时可以使用 lsof 命令，比如查看是谁在使用磁盘分区 sdb1：

```
[root@localhost ~]# lsof /sdb1
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
bash   3322 root  cwd    DIR   8,17      20   64 /sdb1
lsof    4242 root  cwd    DIR   8,17      20   64 /sdb1
lsof    4243 root  cwd    DIR   8,17      20   64 /sdb1
```

此时就可以知道是谁在使用 sdb1 这个分区（或目录）了，将对应的进程杀死或退出后就可以卸载 sdb1。例如，使用 kill 命令杀死正在使用 sdb1 分区的进程：

```
[root@localhost  sdb1]# kill -9  3322
Connection closing ... Socket close.
Connection closed by foregin host.
Disconnected from remote host (centos7-1) at 10:21:23.
Type ‘help’ to learn how to use Xshell prompt.
```

如果知道执行什么命令占用了分区资源，可直接停止退出。例如，在本例中出现不能卸载设备的原因是由于切换到了 sdb1 目录下，因此找到执行切换到 sdb1 目录下的终端后，执行退出目录的命令就可以卸载了。

```
[root@localhost sdb1]# cd
[root@localhost ~]# umount /dev/sdb1
```

7.2.4　利用/etc/fstab 文件挂载分区

在挂载磁盘分区时会涉及/etc/fstab 文件，该文件在系统启动时读取并根据其中的配置来挂载指定的分区。如果出现配置错误，那么系统就无法启动。本节主要介绍该文件配置出错时的修复、基本应用和分区挂载的其他方式。

1. /etc/fstab 文件配置错误修复

通常，在配置磁盘空间时都采取开机自动挂载的方式，这样可以避免系统重启后可能因磁盘没法及时挂载而导致数据无法同步的问题，毕竟无法做到及时对每台服务器重启后手动挂载磁盘。但挂载也可能出现错误，比如在修改/etc/fstab 文件后重启系统的过程中因/etc/fstab 文件修改格式等问题导致系统报错，这时系统就无法继续启动而是等待修复，因此在启动到某个阶段时就会提示输入认证信息并进入单用户模式进行维护。

如图 7-2 所示的界面，这是由于/etc/fstab 文件修改出错导致启动失败的错误提示，此时输入 root 用户密码并进入单用户模式就可以进行修复工作。

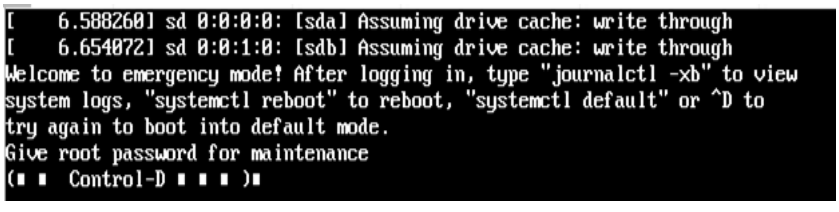


图 7-2　进入单用户模式

造成开机失败的原因是因为在配置磁盘开机自动挂载时命令行出错，只需在进入单用户模式后使用编辑器（如 vi/vim）打开/etc/fstab 文件，并把刚刚添加的自动挂载磁盘分区的命令行删除即可解决问题。如图 7-3 所示是删除后的/etc/fstab 文件配置信息。

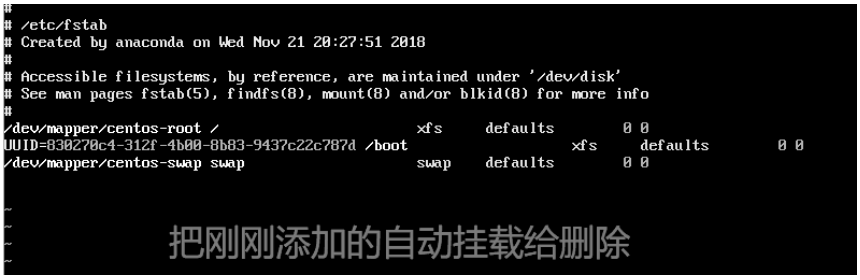


图 7-3　删除后的/etc/fstab 文件配置信息

完成后保存退出，执行 reboot 命令重启系统，这样就处理了自动挂载出现的问题。

2. 挂载分区的方式补充

磁盘分区挂载的方式分为手动挂载和自动挂载，手动挂载更多的是用于日常维护和测试。

通常，使用 mount 命令来将指定的磁盘分区挂载到指定的挂载点上，这是一个比较常见的挂载命令。当然可以直接执行带有选项-a 的 mount 命令挂载磁盘分区，但该命令在/etc/fstab 文件中指定的分区没被挂载时执行才有效。下面是在/etc/fstab 文件中配置的分区没有自动挂载时执行带-a 选项的 mount 命令的执行结果：

```
[root@localhost ~]# mount -a
[root@localhost ~]# df -h
文件系统                容量  已用  可用 已用% 挂载点
/dev/mapper/centos-root  17G  3.8G   14G   23% /
devtmpfs                 895M     0  895M    0% /dev
tmpfs                    911M     0  911M    0% /dev/shm
tmpfs                    911M   11M  901M    2% /run
tmpfs                    911M     0  911M    0% /sys/fs/cgroup
/dev/sda1               1014M  170M  845M   17% /boot
tmpfs                    183M   32K  183M    1% /run/user/1000
/dev/sr0                 4.2G  4.2G     0  100% /run/media/gavin/CentOS 7 x86_64
/dev/sdb1               1014M   33M  982M    4% /sdb1
tmpfs                    183M     0  183M    0% /run/user/0
```

通过 df 命令的输出可以确定新建的磁盘分区已经被挂载。

对于设置磁盘分区在系统启动时的自动挂载，还可以使用 UUID 来代替该磁盘分区的名称。UUID 是系统分配给每个分区的一个特殊标号或标识码，这个符号所指向的是分区的具体类型或路径，如逻辑卷所在的绝对路径等。

UUID（Universally Unique Identifier）称为通用唯一识别码，因此在使用上相当于磁盘分区的名称，也就是说要在/etc/fstab 文件上设置磁盘分区开机自动挂载可使用它来代替分区的名称，不过在使用前需要获取 UUID。对于分区所对应的 UUID，可以使用 blkid 命令来获取：

```
[root@localhost ~]# blkid
/dev/sda1: UUID="bac2ebef-26d2-4a61-9c27-bad2d7290120" TYPE="xfs"
```

```
    /dev/sda2: UUID="QCYxLM-ukM3-wQq1-rTdS-1RJ3-YO0e-R7qdtT" TYPE="LVM2_member"
    /dev/sdb1: UUID="6ff67883-8e92-4d57-8743-1293611b9a0e" TYPE="xfs"
    /dev/sr0: UUID="2018-05-03-20-55-23-00" LABEL="CentOS 7 x86_64"
TYPE="iso9660" PTTYPE="dos"
    /dev/mapper/centos-root: UUID="92493bdd-6684-4145-beb1-eccf09bad290"
TYPE="xfs"
```

通过命令的输出可以获取系统当前所有磁盘分区与其所对应的 UUID 及文件系统类型等信息。比如，要将磁盘分区/dev/sdb1 通过/etc/fstab 文件来设置开机自动挂载，配置时所需的命令行中要指定分区名称、挂载点、文件系统类型等，使用 UUID 时就不需要使用分区名称。

```
    [root@localhost ~]# echo "6ff67883-8e92-4d57-8743-1293611b9a0e  /sdb1 xfs
defaults  0  0" >> /etc/fstab
```

echo 命令把“6ff67883-8e92-4d57-8743-1293611b9a0e /sdb1 xfs defaults 0 0”写入/etc/fstab 文件中，设置分区开机自动挂载。

需要注意的是，“>>”符号（重定向符）可以把信息写入指定文件的末尾处，并保持原有内容不变，而执行“>”符号时意味着要先清空文件的内容再写入。请注意这两个符号的区别。

7.3　gdisk 磁盘分区工具

gdisk 是一个用于创建分区的工具，主要用来划分容量大于 4TB 的硬盘。大于 4TB 的磁盘使用 fdisk 命令来创建分区时明显存在不足。

分区表有两种类型：GPT（GUID Partition Table，全局唯一标识分区表）和 MBR。其中，MBR 不支持 4TB 以上。

GPT 分区使用 128 位 GUID 来唯一标识每个磁盘和分区，与 MBR 存在单一故障点不同。GPT 提供分区表信息的冗余，一个在磁盘头部，一个在磁盘尾部；它通过 CRC 校验和来检测 GPT 头和分区表中的错误与损坏；默认一个硬盘支持 128 个分区。

示例：对 sdb 做 GPT 分区，创建一个 sdb1。

执行 gdisk 命令进入创建磁盘分区的交互界面，比如创建/dev/sdb 磁盘分区。

```
[root@localhost ~]# gdisk /dev/sdb
…
Command (? for help): ?    #查看帮助
b   back up GPT data to a file
c   change a partition's name
d   delete a partition    #删除分区
i   show detailed information on a partition
l   list known partition types
n   add a new partition       #添加一个分区
o   create a new empty GUID partition table (GPT)
p   print the partition table      #打印分区表
q   quit without saving changes     #退出不保存
r   recovery and transformation options (experts only)
s   sort partitions
t   change a partition's type code
v   verify disk
```

```
w   write table to disk and exit     #  # 写入分区表并退出
x   extra functionality (experts only)
?   print this menu

Command (? for help): n   #新建分区表
Partition number (1-128, default 1):     #直接按回车键
#直接按回车键，从头开始划分空间
First sector (34-41943006, default = 2048) or {+-}size{KMGTP}:
#给 1GB 空间
Last sector (2048-41943006, default = 41943006) or {+-}size{KMGTP}: +1G
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300):     #分区类型直接按回车键
注：8300 Linux filesystem  ；8e00 Linux LVM     想查看，可以按 L 键显示
Changed type of partition to 'Linux filesystem'

Command (? for help): p  #查看
…
Number  Start (sector)    End (sector)  Size       Code  Name
   1          2048        2099199   1024.0 MiB  8300  Linux filesystem
Command (? for help): w   #保存
Do you want to proceed? (Y/N): y    #确定写入
OK; writing new GUID partition table (GPT) to /dev/sdb.
The operation has completed successfully.
```

通过上述操作，创建了名为/dev/sdb1 的磁盘分区。

分区创建后是不能直接挂载的，需要先对新建的分区进行格式化，否则挂载时会出错。要对分区进行格式化，可以使用 mkfs.xfs4 命令并指定要格式化的分区名称，如对/dev/sdb1 分区进行格式化。执行下述命令：

```
[root@localhost ~]# mkfs.xfs4 /dev/sdb1
```

结果显示如图 7-4 所示。

```
[root@localhost ~]# mkfs.ext4  /dev/sdb1
mke2fs 1.42.9 (28-Dec-2013)
文件系统标签=
OS type: Linux
块大小=4096 (log=2)
分块大小=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
65536 inodes, 262144 blocks
13107 blocks (5.00%) reserved for the super user
第一个数据块=0
Maximum filesystem blocks=268435456
8 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Allocating group tables: 完成
```

图 7-4　分区格式化的结果

完成对分区的格式化之后，就可以在挂载后使用。

第 8 章

RAID 磁盘阵列的搭建

RAID（Redundant Array of Inexpensive Disks，廉价磁盘冗余阵列）提出的理由是提高磁盘存储容量和改善磁盘的存储性能，目前在服务器中已经得到普及性应用。本章主要学习 RAID 的搭建，包括软 RAID（通过操作系统软件来实现）和硬 RAID（使用硬件阵列卡）及其 RAID 的配置。

8.1 RAID 概述

RAID 于 1987 年由美国 Berkeley 大学的两名工程师提出，最初目的是把多个容量较小的廉价硬盘合并成为一个大容量的“逻辑盘”或磁盘阵列，实现提高硬盘容量和性能的功能。

随着 RAID 的普及，其各方面的技术都得到了发展，再加上大数据信息化时代的推动，磁盘阵列已经是服务器的标配。通过磁盘阵列能够保证单盘故障有替换盘，最大限度地保证数据的安全，同时它还能利用同位检查（Parity Check）技术在数组中任意一个硬盘故障时读出数据，保证数据的完整性。

RAID 的实现有软件和硬件两种方式，软件实现需要操作系统的支持，硬件实现就是用专用的 RAID 卡。

1. 软件实现的 RAID

目前，有的网络操作系统可以使用标准的 SCSI 适配卡支持和管理驱动器，也能够支持 RAID0、RAID1 和 RAID5 等类型的阵列。

软件实现的 RAID 不能保护系统盘，因为有些操作系统会把 RAID 的配置信息存在系统信息中，一旦系统发生故障就意味着 RAID 的信息丢失，还有些操作系统并不支持热插拔或热交换，因此在硬盘出现故障时无法在线更换，这都是软件 RAID 的弊端。

2. 硬件实现的 RAID

硬件实现的 RAID 采用集成的阵列卡或专用的阵列卡来控制硬盘驱动器，它的控制器直接接入主系统的存取接口，是一个独立的直接存取储存体，也就是说操作系统是不会看到或直接管理硬盘的。

硬件实现的 RAID 在安装操作系统之前就已完成配置，安装系统时只是看到一块大容量的磁盘。硬件 RAID 技术的使用可以极大地节省系统的资源，从而使服务器的性能获得很大的提高。更重要的是，这种方式实现的 RAID 能够支持硬盘在线更换、热插拔或热交换，并且还可以通过后台系统来实现对它们的监控和管理。

8.2 常见的 RAID 类型

目前，RAID 已由最初的 RAID0~RAID5 扩展到了 RAID0+1 和 RAID0+5 等不同的组合方式，并且可以根据实际需要来设置，包括扩大储存空间、提供数据冗余和提高磁盘系统的 I/O 能力等。这几类阵列组合是目前比较常见的，而且都有自己的优缺点，在实际的生产环境下须根据实际需要综合衡量出一种“最合适”的 RAID。RAID 的基本类型和特点如表 8-1 所示。

表8-1 几种常见的RAID类型

RAID 类型		最低磁盘个数	空间利用率	各自的优缺点
级 别	说 明			
RAID0	条带卷	2+	100%	读写速度快，不容错
RAID1	镜像卷	2	50%	读写速度一般，容错
RAID5	带奇偶校验的条带卷	3+	（n−1）/n	读写速度快，容错，允许坏一块盘
RAID10	RAID1 的安全+RAID0 的高速	4	50%	读写速度快，容错

8.2.1 RAID0

RAID0 可用于两个或更多硬盘或 SSD（至少需要两块盘），目的是提高访问的性能，有效平衡 I/O 问题。

在 RAID0 中储存数据时，数据以特定大小（通常是 6KB）的块存储，并由 RAID 阵列控制器平均分配到每块硬盘上，其分配的原理是：第一块数据块写入硬盘 1，第二块数据块写入硬盘 2，第三块数据块写入硬盘 1，第四块数据块写入硬盘 2，以此类推，直到数据写完。对于这类阵列，可以把它们理解成“与”的关系。

由于数据分布均匀，因此在取数据时会在硬盘 1~硬盘 2 之间进行，并把提取到的数据拼接在一起形成一个完整的数据。RAID0 阵列的储存空间可用率能够达到 100%，而且由于是多块硬盘同时读取数据，因此其访问数据的速度会成倍增加，理论上访问能力与磁盘数量成正比。

当然，RAID0 也有缺点，没有采用冗余技术，无法对硬盘在线维护。数据均匀放在每块盘上，如果其中一个磁盘出现故障，就意味着得到的数据不是完整的。另外，磁盘越多，发生故障的可能性越高，所以建议将 RAID0 上的重要数据备份到其他地方。

8.2.2 RAID1

RAID1 中的硬盘是一种镜像关系，它会将同一份数据写入两个单独的硬盘，并在一块硬盘发生故障时使用另外一块硬盘上的数据。

这里需要明白的是，镜像不是备份，简单来讲，硬盘 HDD1 和 HDD2 上的数据是一样的，在 HDD1 上的数据出问题后，HDD2 的数据也会出现同样的问题，也就是说数据是一致的，只有硬盘出现故障时数据才会存在差异。备份是保持源数据不变，除非是数据被更新。

RAID1 的读性能通常与单独的硬盘差不多，在需要数据时会同时读镜像盘，先读取到哪块盘的数据就用哪块的，但在写数据时是同时写入的。因此，使用 RAID1 来获得额外的读写性能是不太可能的。另外，对于 RAID1 而言，在一定的程度上是为保证系统的正常工作，但数据的完整性会受到影响；从空间可用率的角度上来看，其存储空间使用率只有 50%。

8.2.3　RAID10 和 RAID01

RAID10（或称 RAID1+0，组合架构如图 8-1 所示）实际上就是 RAID1 和 RAID0 这两种磁盘阵列的组合，这种组合先做 RAID1，并在此基础上做 RAID0。同理，RAID0+1（或称 RAID01）先做 RAID0，并在此基础上做 RAID1。无论是 RAID10 还是 RAID01，都至少需要 4 块硬盘。

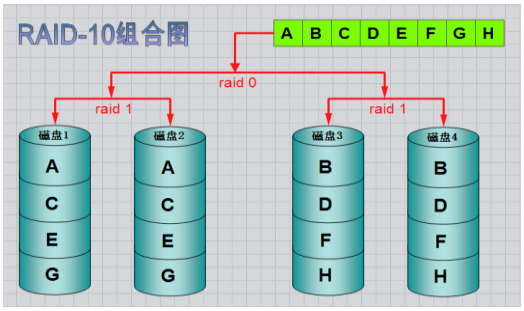


图 8-1　RAID10 磁盘阵列

8.2.4　RAID3

RAID3 的实现至少需要三块盘，实现的原理是：如果有 N（N 大于等于 3）块硬盘，其中拿出一块作校验盘，剩余（N−1）块相当于作 RAID0 同时读写，当（N−1）中的某块盘出现故障时，可以通过校验码还原出坏掉盘的原始数据。

这种校验方式是奇偶检验，在校验后发现中间有缺失数据时就会通过其他盘的数据和校验数据推算出来。由于这是在 RAID0 上实现的，因此每一次读写数据时全部硬盘上的数据都会被更新，如果校验盘损坏就意味着数据找不回来了。

8.2.5　RAID5 或 RAID6

RAID5 算是多种阵列技术的集合，数据的保存与 RAID0 一样，是分成块并执行写入处理；把 RAID3 的“校验盘”也分成块分散到所有盘里，产生并写入称为“奇偶校验”的冗余代码。因此，即使其中的一个硬盘出现故障，也可以根据剩余的数据和奇偶校验来计算出丢失的数据，这样就可以把数据还原。

由于是多个驱动器同时处理数据的读取，因此 RAID5 模式下硬盘读取数据的速度很快，速度与驱动器的数量成比例增加，但数据的写入及奇偶校验会慢一些。RAID5 已经提供一定程度的可靠性，也牺牲了一定的读取速度。另外，RAID5 只允许一块硬盘故障，如果是两块或两块以上的

硬盘同时发生故障，RAID5 就无能为力了，于是出现了 RAID6。

与 RAID5 对比，RAID6 最大的区别就是在 RAID5 的基础上除了具有 P 校验位以外，还加入了第二个校验位 Q 位。在这种情况下，当一块磁盘出现数据错误或丢失的时，恢复方法同 RAID5（不需要使用 Q 校验位），当两块磁盘上的数据都出现错误或丢失时，可利用给出的 P、Q 校验数据进行恢复。

基于 RAID5 和 RAID6 的基础上也出现了 RAID5+0 和 RAID6+0 的组合阵列。

RAID5+0（或称 RAID50）是在 RAID5 的基础上再建立 RAID0。同理，RAID6+0（也称 RAID60）是在 RAID6 的基础上再建立 RAID0。

当服务器的硬盘数量达到 10 块或更多时，可以通过 RAID50 或 RAID60 来解决，至于使用哪一种，可以综合硬盘数量、设备运行的业务等各方面的信息来决定。

8.3 案例：华为 2U 机架式服务器 RAID5 配置

在实际的工作环境下很少涉及软 RAID 的配置，再加上虚拟化的普及和 LVM 的应用和数据的远程备份，能够使用软 RAID 的环境很少，因此这里主要就硬 RAID 进行介绍。硬 RAID 一直在使用，毕竟每台服务器上都需要使用。

在服务器上使用硬 RAID 须阵列卡的支持。阵列卡有集成型和独立型，在一些对设备要求不高且考虑到成本等问题的环境中，通常采用集成型的阵列卡，有些阵列卡需要安装驱动，否则无法配置 RAID。以下我们以华为的服务器作为例子来讲解 RAID5 配置的基本过程。

示例：这是一台 2U 的机架式服务器（或称网络服务器），运行的是公司的业务系统，数据库和应用系统都在其上运行，所配置的是集成阵列卡和三块硬盘，在现有的资源条件下尽最大可能保证数据的安全，因此采取 RAID5 的方式。

具体 RAID5 的配置过程如下：

（1）启动服务器，在启动的过程中出现如图 8-2 所示的 Press<Ctrl><R> to Run MegaRAID Configuration Utitity 界面时按 Ctrl+R 组合键来配置 RAID。

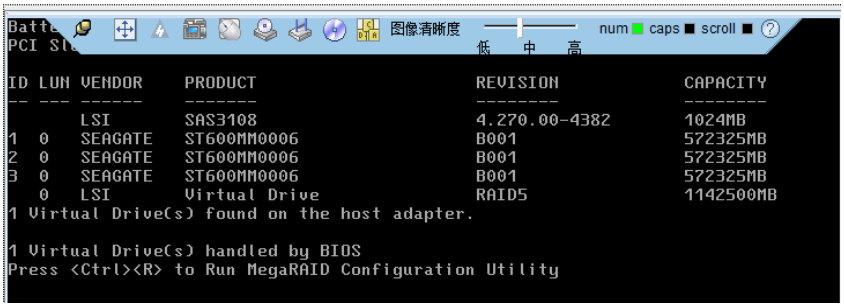


图 8-2 Press<Ctrl><R> to Run MegaRAID Configuration Utitity 界面

（2）在如图 8-3 所示的 Virtual Drives Management 界面中，将看到被阵列卡识别的硬盘及相关的概要信息。由于接下来要配置 RAID，因此将光标移动到 SAS 3108（Bus 0x01, Dev 0x00）后。

图 8-3　Virtual Drives Management 界面

（3）根据界面提示，按 F2 键打开如图 8-4 所示的虚拟磁盘配置菜单。

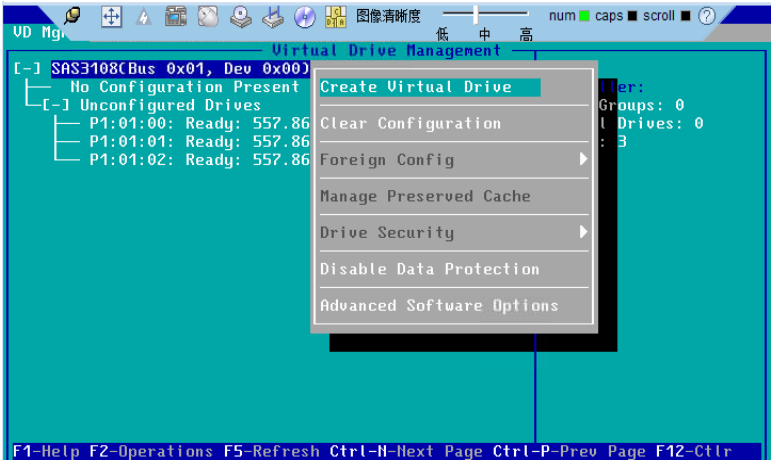


图 8-4　打开虚拟磁盘配置菜单

（4）由于要配置 RAID，因此需要选择 Create Virtual Drive 选项并按回车键，之后打开创建 RAID 的界面，在此界面中可以看到 RAID 级别的选择、可用的硬盘和相关的描述信息等。接下来在 RAID Level 处选择 RAID 的级别，将光标移动到此处后按回车键，并在下拉菜单中选择 RAID-5。在其右侧以按空格符的方式将所有的硬盘都选中，并单击 OK 按钮确认，如图 8-5 所示。

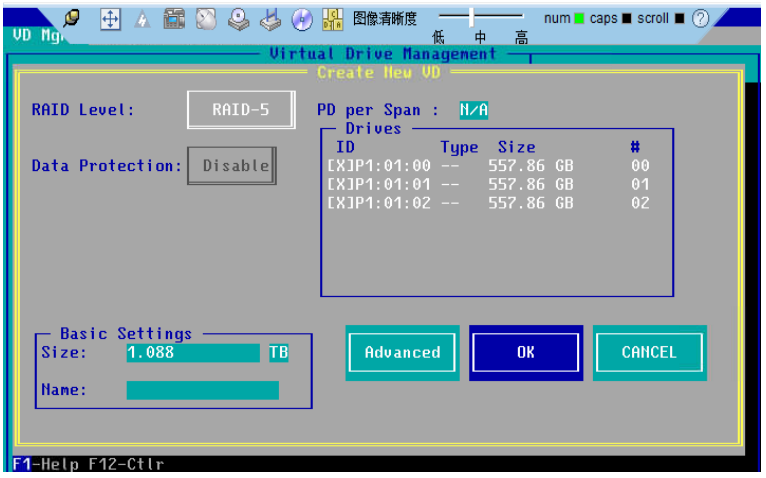


图 8-5　硬盘选择

（5）在接下来的界面中单击 OK 按钮，返回 Virtual Drives Management 界面，会看到 RAID5，说明 RAID5 已经配置完成，如图 8-6 所示。

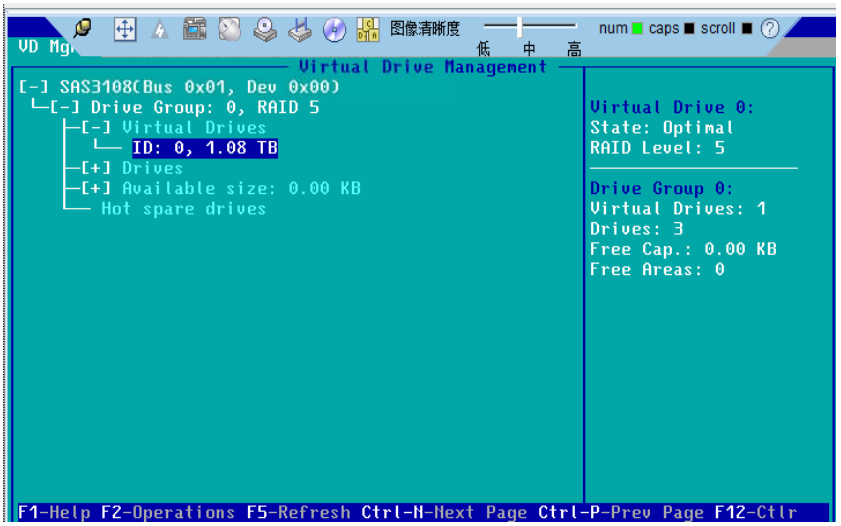


图 8-6　在 Virtual Drives Management 界面配置完成 RAID5

如果要删除创建的 RAID，可在 Virtual Drives Management 界面选择第一项并按回车键，在弹出的菜单中选择如图 8-7 所示的 Clear Configuration 选项，并在弹出的警告信息界面中单击 YES 按钮就可以删除 RAID。

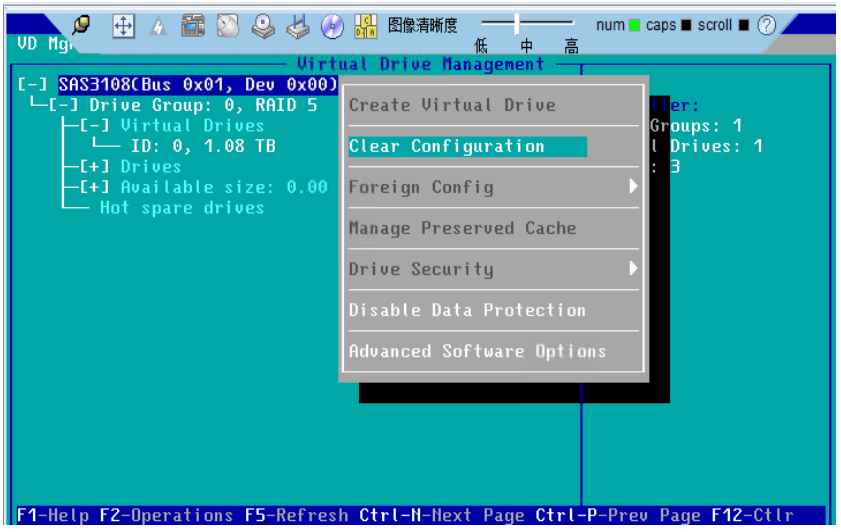


图 8-7　选择“Clear Configuration”选项

运维前线

删除并创建 RAID 时，如果重建的是一样级别的 RAID，那么原先的数据是不会被清除的。例如，原来是 RAID5，被删除后再重建 RAID5，那么原来的数据不会被清除，只有重建的不是 RAID5 时原来的数据才会被清除。

第 9 章

LVM 存储空间的管理

磁盘资源是数据存储的主要载体，对于这种在数据存储不确定环境下对有限磁盘资源的合理规划是很不容易做到的，而 LVM（Logicd Volume Manager，逻辑卷管理）的出现在很大程度上解决了这些问题，本章将介绍 LVM 的基本概念、LVM 的创建和常用命令及维护方法。

9.1　LVM 概述

9.1.1　LVM 的原理

每个磁盘可分成多个分区，每个分区为一个物理卷（physical volume，PV），这些物理卷是使用大小固定的物理区段来定义的。若干个物理卷可组成一个卷组（volume group，VG），形成一个共享池，而在卷组上可以创建一个或多个逻辑卷（logical volumes，LV），并在这些逻辑卷上创建文件系统。如图 9-1 所示是这三者之间的关系。

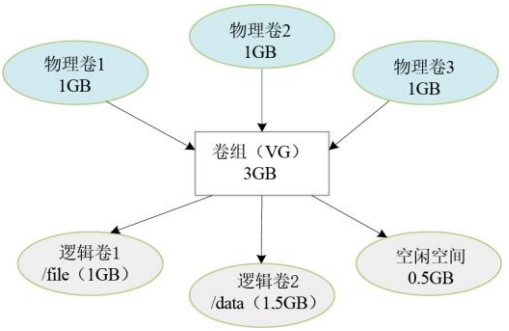


图 9-1　PV、VG 和 LV 的关系

一直以来磁盘空间使用完后都是添加磁盘块数或者把数据迁移到空间更大的地方，可选的余地很少。随着技术的发展，数据量爆发式增加，对磁盘存储空间的要求也在不断增加，再加上应用场合的多样化，因此往往出现初期不需要过大的磁盘空间，而后期需要不停扩容空间的情况，再加上资金方面等问题，就需要有一种灵活多变的存储空间管理方式，LVM 随即出现了。

LVM 是在磁盘分区和文件系统之间添加的一个逻辑层，这个层为文件系统屏蔽下层磁盘分区布局提供一个抽象的盘卷，并允许在盘卷上建立文件系统。管理员利用 LVM 可以在磁盘不用重新分区的情况下动态调整文件系统的大小，并且可以跨越磁盘。当服务器添加了新的磁盘后，管理员

不必将原有的文件移动到新的磁盘上，而是通过 LVM 直接扩展文件系统跨越磁盘。

LVM 将底层的物理硬盘封装起来，以逻辑卷的方式呈现给上层应用，当我们对底层的物理硬盘进行操作时，不再是针对分区进行操作，而是通过逻辑卷来对其进行底层的磁盘管理操作。

9.1.2 LVM 常用术语

- 物理存储介质（physical media）: LVM 的存储介质既可以是磁盘分区，也可以是整个磁盘、RAID 阵列或 SAN 磁盘，不管是哪种设备都必须先被初始化为 LVM 物理卷才能使用。
- 物理卷（physical volume，PV）：物理卷是 LVM 的基本存储逻辑块，但和基本的物理存储介质（如分区、磁盘等）比较，它包含有与 LVM 相关的管理参数。例如，创建物理卷可以用硬盘分区，也可以用硬盘本身。
- 卷组（volume group，VG）：一个 LVM 卷组由一个或多个物理卷组成。
- 逻辑卷（logical volume，LV）：LV 建立在 VG 之上，可以在 LV 之上建立文件系统。
- PE（physical extents）：PV 物理卷中可以分配的最小存储单元，大小可以自己指定，默认值为 4MB。
- LE（logical extent）：LV 逻辑卷中可以分配的最小存储单元，在同一个卷组中 LE 的大小和 PE 是相同的，并且是一一对应的关系。

最小存储单位如表 9-1 所示。

表9-1 最小存储单位

名 称	最小存储单位
硬盘	扇区（512 字节）
文件系统	block（1K 或 4K）# mkfs.ext4 -b 2048 /dev/sdb1，最大支持到 4096
RAID	chunk（512K）#mdadm -C -v /dev/md5 -l 5 -n 3 -c 512 -x 1 /dev/sde{1,2,3,5}
LVM	PE（4M）# vgcreate -s 4M vg1 /dev/sdb{1,2}

9.1.3 LVM 的优点

LVM 主要是基于虚拟化环境下的应用，与传统的磁盘空间管理方式相比发生了很大的变化，比如其可以在线更改磁盘空间的大小。对于中小企业来说，这将为起步阶段的服务器应用节省不少的成本。

相对于传统的磁盘管理方式，LVM 具有以下优点：

- 能够随时更改磁盘空间的大小，即使后期增加磁盘空间，这些磁盘所组成的磁盘组看起来也像是一个大的磁盘。
- 使用逻辑卷，可以跨多个硬盘空间的分区。
- 在使用逻辑卷时，可以在空间不足时动态调整大小。
- 在调整逻辑卷大小时，不需要考虑逻辑卷在硬盘上的位置，不用担心磁盘空间的连续性。
- 可以在线创建、删除及调整 LV、VG 的空间大小，而不影响系统的运行。
- 允许创建磁盘快照，可以用来保存文件系统的备份。

运维前线

RAID 和 LVM 可以一起用，LVM 是软件的卷管理方式，RAID 是磁盘管理的方法。对于重要的数据，使用 RAID 来保护物理磁盘不会因为故障而中断业务，再用 LVM 来实现对卷的可伸缩性管理，可更好地利用磁盘资源。

9.2　创建 LVM 的基本步骤

9.2.1　LVM 的创建

创建 LVM 的具体方法如下：

（1）物理磁盘被格式化为 PV，空间再被划分为一个或多个 PE。
（2）不同的 PV 加入同一个 VG 中，不同 PV 的 PE 全部进入 VG 池内。
（3）在 VG 中创建 LV，组成 LV 的 PE 可能来自不同的物理磁盘。
（4）LV 可以直接格式化后挂载使用。
（5）LV 的扩充缩减实际上就是增加或减少 LV 的 PE 数量，不会丢失原始数据。

下面我们来看看具体的操作步骤。
在虚拟机上添加一块硬盘和一个 sdb 磁盘，并使用相同的方式添加 sdc 磁盘，如图 9-2 所示。

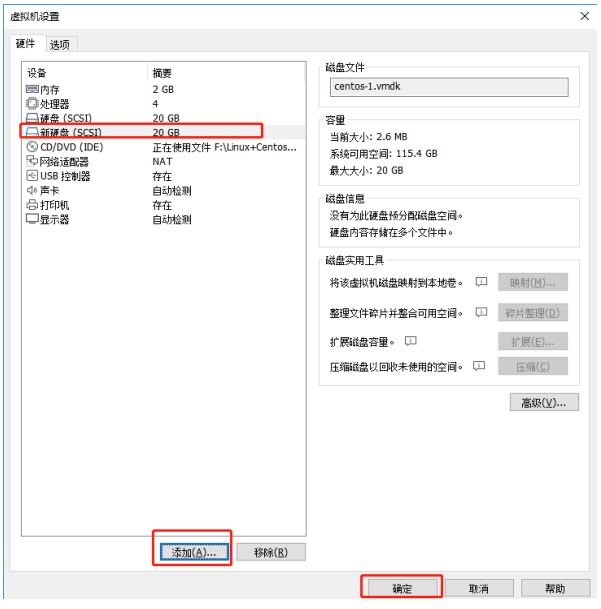


图 9-2　在虚拟机上添加一块硬盘

创建磁盘分区，使用 fdisk 命令进入交互模式后进行磁盘分区的创建，即使用 fdisk 命令进入 /dev/dbc 磁盘创建分区。

设定分区类型代码：fdisk /dev/sdb ===> t ===> 选择分区号 ====> 8e ====> w。

```
[root@localhost ~]#fdisk /dev/sdc
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n     # 创建磁盘分区
Command action
   e   extended
   p   primary partition (1-4)
p                           # 创建主分区
Partition number (1-4): 1
First cylinder (1-652, default 1):    # 保持默认，默认第一个分区编号
Using default value 1
# 保持默认，即默认第一个扇区开始位置，即整块盘都用于创建磁盘分区
Last cylinder or +size or +sizeM or +sizeK (1-652, default 652):
Using default value 652

Command (m for help): t     # 转换 System 类型
Selected partition 1
Hex code (type L to list codes): 8e     # 指定转换的代码
Changed system type of partition 1 to 8e (Linux LVM)

Command (m for help): p     # 显示分区信息

Disk /dev/sdc: 5368 MB, 5368709120 bytes
255 heads, 63 sectors/track, 652 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device   Boot      Start       End      Blocks     Id     System
/dev/sdc1               1         652     5237158+    8e   Linux LVM

Command (m for help): w     # 保存退出
The partition table has been altered!
```

上面是在/dev/sdc 分区上创建的。关于在交互模式下获取帮助，可以使用 h 命令。如果要查看 system 类型，可以执行 l 命令。

同样，可以在添加/dev/sdb 磁盘后创建 4 个分区，并在创建完成后使用以下命令查看磁盘分区的数量。

```
[root@localhost ~]# ls /dev/sdb*
/dev/sdb  /dev/sdb1  /dev/sdb2  /dev/sdb3  /dev/sdb4
```

从输出的信息看，/dev/sdb 共有 4 个分区。

物理卷是基于分区建立的，下面通过 pvcreate 命令为/dev/sdb 磁盘创建 4 个物理卷分区。

```
[root@localhost ~]#pvcreate /dev/sdb{1,2,3,4}
  Physical volume "/dev/sdb1" successfully created.
  Physical volume "/dev/sdb2" successfully created.
  Physical volume "/dev/sdb3" successfully created.
  Physical volume "/dev/sdb4" successfully created.
```

完成物理卷的创建后，可以使用 pvdisplay 命令查看物理卷的相关信息：

```
[root@localhost ~]#pvdisplay /dev/sdb1
  "/dev/sdb1" is a new physical volume of "1.00 GiB"
  --- NEW Physical volume ---
  PV Name           /dev/sdb1
  VG Name
  PV Size               1.00 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID           SHKFwf-WsLr-kkox-wlee-dAXc-5eL0-hyhaTV
```

卷组是建立在物理卷上的，因此完成物理卷的创建后就可以创建卷组了。使用 vgcreate 命令创建名称为 vg01 的卷组。

```
[root@localhost ~]# vgcreate vg01 /dev/sdb1
  Volume group "vg01" successfully created
```

查看卷组的相关信息。

```
[root@localhost ~]# vgs
  VG   #PV #LV #SN Attr   VSize    VFree
  vg01   1   0   0 wz--n- 1020.00m 1020.00m
[root@localhost ~]# vgdisplay vg01
  --- Volume group ---
  VG Name           vg01
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No      1
  VG Access             read/write
  VG Status             resizable
  MAX LV            0
  Cur LV            0
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               1020.00 MiB
  PE Size               4.00 MiB
  Total PE              255
  Alloc PE / Size       0 / 0
```

逻辑卷是建立在卷组之上的，创建命令为 lvcreate。创建逻辑卷时可以使用-L 选项来指定空间大小。例如，创建名称为 lv01 和 lv02 的逻辑卷，大小为 16MB。

```
[root@localhost ~]# lvcreate -n lv01 -L 16M vg01
  Logical volume "lv01" created.
[root@localhost ~]# lvcreate -n lv02 -l 4 vg01
  Logical volume "lv02" created.
```

查看逻辑卷的信息：

```
[root@localhost ~]# lvs
  LV   VG   Attr   LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  lv01 vg01 -wi-a----- 16.00m
  lv02 vg01 -wi-a----- 16.00m
```

查看/dev/sdb1 分区的信息：

```
[root@localhost ~]#pvdisplay /dev/sdb1
  --- Physical volume ---
  PV Name               /dev/sdb1
  VG Name           vg01
  PV Size               1.00 GiB / not usable 4.00 MiB
  Allocatable           yes
  PE Size               4.00 MiB
  Total PE              255
  Free PE               247
  Allocated PE          8
```

9.2.2 LVM 管理常用命令

在对 LVM 的管理和使用上，有不少的命令（工具），如表 9-2 所示。获取 LVM 相关信息常用的一些命令如表 9-3 所示。

表9-2 使用和管理LVM常用的命令

功 能	PV 管理命令	VG 管理命令	LV 管理命令
scan（扫描）	Pvscan	vgscan	lvscan
create（创建）	Pvcreate	vgcreate	lvcreate
Display（显示）	Pvdisplay	vgdisplay	lvdisplay
remove（移除）	Pvremove	vgremove	lvremove
extend（扩展）		vgextend	lvextend
Reduce（减少）		vgreduce	lvreduce

表9-3 获取LVM信息常用的命令

查看卷名	简单对应卷信息的查看	扫描相关的所有对应卷	详细对应卷信息的查看
物理卷	pvs	pvscan	pvdisplay
卷组	vgs	vgscan	vgdisplay
逻辑卷	lvs	lvscan	lvdisplay

9.2.3 逻辑卷的挂载

挂载逻辑卷到系统目录下，其原理相当于挂载磁盘分区。要挂载逻辑卷，首先要格式化逻辑卷，而在格式化逻辑卷前需要获取逻辑卷名称：

```
[root@test ~]# lvdisplay
  --- Logical volume ---
  LV Name                /dev/vgsdb1/lvsdb
  VG Name              vgsdb1
  LV UUID               WYoOGZ-KKgC-cnyi-cEsX-8KAV-t2wY-uJVi2t
```

```
  LV Write Access           read/write
  LV Status              available
  # open              0
  LV Size               1.00 GB
  Current LE            256
  Segments              1
  Allocation            inherit
  Read ahead sectors        0
  Block device          253:0
```

如果系统中只有少量的逻辑卷，那么使用该命令时很快就能找到逻辑卷名称。若系统中有大量的逻辑卷，则可以使用 lvscan 命令来扫描系统的逻辑卷：

```
[root@test ~]# lvscan
  ACTIVE          '/dev/vgsdb1/lvsdb' [1.00 GB] inherit
```

从输出的信息中可知，逻辑卷的完整路径为'/dev/vgsdb1/lvsdb'。接着格式化逻辑卷，也就是在该逻辑卷上创建文件系统（文件系统为 ext4）。

```
[root@test ~]# mkfs -t ext4 /dev/vgsdb1/lvsdb
mke2fs 1.35 (28-Feb-2004)
max_blocks 268435456, rsv_groups = 8192, rsv_gdb = 63
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
131072 inodes, 262144 blocks
13107 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=268435456
8 block groups
32768 blocks per group, 32768 fragments per group
16384 inodes per group
Superblock backups stored on blocks:
       32768, 98304, 163840, 229376

Writing inode tables: done
inode.i_blocks = 2528, i_size = 4243456
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 23 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

挂载逻辑卷到系统的/opt 目录下。

```
[root@test ~]# mount /dev/vgsdb1/lvsdb /opt
[root@test ~]# mount      # 检查系统已挂载的文件系统
/dev/sda4 on / type ext4 (rw)
none on /proc type proc (rw)
none on /sys type sysfs (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/sda1 on /boot type ext4 (rw)
none on /dev/shm type tmpfs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
```

```
none on /proc/fs/vmblock/mountPoint type vmblock (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
/dev/mapper/vgsdb1-lvsdb on /opt type ext4 (rw)
```

结果显示逻辑卷/dev/vgsdb1/lvsdb 是以读写的方式挂载到/opt 目录下。

9.3 LVM 的日常维护

一些存储空间不足的服务器会通过 LVM 的方式来增加存储空间，本节将从 LVM 方面来介绍存储空间的扩容和移除。

9.3.1 LV 存储空间扩容

在 LVM 中，LV 存储空间的扩容使用的是 lvextend 命令。在扩容之前，需要知道是给谁扩容、使用的是谁的存储空间以及扩容多少。

接下来演示如何对 LV 的存储空间进行扩容。由于是对 LV 扩容，所需要的存储空间直接来自卷组（VG），因此要先获取系统中的卷组（可用空间）。

```
[root@localhost ~]# vgs
  VG   #PV #LV #SN Attr   VSize    VFree
  vg01   1   2   0 wz--n- 1020.00m  988.00m
  vg02   1   0   0 wz--n- 1008.00m 1008.00m
```

从输出的信息中可以看到有两个卷组，并且它们都有可用的存储空间。可以通过 lvscan 命令来获取逻辑卷的路径，之后使用 lvextend 命令来给逻辑卷扩容，比如给/dev/vg01/lv01 增加 30MB 的存储空间。

```
[root@localhost ~]# lvextend -L +30m /dev/vg01/lv01
```

说明：在指定大小的时候，扩容 30MB 和扩容到 30MB 的写法不一样。

- 扩容 30MB：-L+30M。
- 扩容到 30MB：-L 30M。

```
[root@localhost ~]# lvextend -L +30m /dev/vg01/lv02
  Rounding size to boundary between physical extents: 32.00 MiB.
  Size of logical volume vg01/lv01 changed from 16.00 MiB (4 extents) to 48.00
MiB (12 extents).
  Logical volume vg01/lv01 successfully resized.
```

执行 lvextend 命令对/dev/vg01/lv01 扩容后，实际上空间是没有发生变化的，还需要使用 resize2fs 命令来使更改生效。

```
[root@localhost ~]# resize2fs /dev/vg01/lv01
resize2fs 1.42.9 (28-Dec-2013)
Filesystem at /dev/vg01/lv01 is mounted on /lv01; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 1
The filesystem on /dev/vg01/lv01 is now 49152 blocks long.
```

ext4 文件系统扩容使用的命令语法如下：

```
resize2fs  逻辑卷名称
```

xfs 文件系统扩容使用的命令语法如下：

```
xfs_growfs  挂载点
```

resize2fs 和 xfs_growfs 的区别是传递的参数不一样，xfs_growfs 采用的是挂载点，resize2fs 是逻辑卷名称，而且 resize2fs 命令不能对 xfs 类型文件系统使用。

9.3.2　VG 存储空间扩容

卷组扩容需要的存储空间来自物理卷，因此首先要有空闲的物理卷。在给卷组扩容存储空间时，可以使用 vgscan 命令来获取卷组的路径，并在有空闲物理卷时加入卷组。

例如，使用物理卷/dev/sdb3 给卷组/dev/sdb3 增加存储空间：

```
[root@localhost ~]# pvcreate /dev/sdb3
[root@localhost ~]# vgextend vg01 /dev/sdb3
  Volume group "vg01" successfully extended
```

9.3.3　LVM 删除操作

可以按照“umount 卸载→lvremove lv 移出卷组中所有逻辑卷→vgremove vg 移出卷组→pvremove 移出 pv”的流程删除 LVM。

首先要说明的是，在实际的生产环境下对卷的移除需要非常谨慎，以免导致服务器宕机，除非是一些确认不再使用或用于测试的卷才能移除。

移除卷的操作步骤如下。

（1）卸载挂载到系统下的逻辑卷，即/lv01/。

```
[root@localhost ~]# umount /lv01
[root@localhost ~]#  lvremove /dev/vg01/lv01
Do you really want to remove active logical volume vg01/lv01? [y/n]: y
  Logical volume "lv01" successfully removed
```

（2）使用 lvs 命令确认被移除的卷是否还存在。

```
[root@localhost ~]# lvs
  LV   VG   Attr  LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  lv02 vg01 -wi-a----- 16.00m       #已经看不到 lv01
```

（3）逻辑卷移除后，接着对逻辑卷所在的卷组进行移除。

```
[root@localhost ~]#  vgremove vg01
Do you really want to remove volume group "vg01" containing 1 logical volumes?
[y/n]: y
Do you really want to remove active logical volume vg01/lv02? [y/n]: y
  #如果卷组中还有 lv，移出时会提示是否也移出，这里直接移出
  Logical volume "lv02" successfully removed
  Volume group "vg01" successfully removed
```

（4）使用 vgs 命令确认卷组是否已经被移除。

```
[root@localhost ~]# vgs
  VG   #PV #LV #SN Attr   VSize    VFree
  vg02   1   0   0 wz--n- 1008.00m 1008.00m    #没有 vg01
```

（5）移除物理卷。

```
[root@localhost ~]# pvs
  PV         VG   Fmt  Attr PSize    PFree
  /dev/sdb1       lvm2 ---     1.00g   1.00g
  /dev/sdb2  vg02 lvm2 a--  1008.00m 1008.00m
  /dev/sdb3       lvm2 ---     1.00g   1.00g
  /dev/sdb4       lvm2 ---     1.00g   1.00g
```

（6）移除分区。

```
[root@localhost ~]# pvremove /dev/sdb1
  Labels on physical volume "/dev/sdb1" successfully wiped.
```

（7）查看物理卷，/dev/sdb1 卷已经不存在。

```
[root@localhost ~]# pvs
  PV         VG   Fmt  Attr PSize    PFree
  /dev/sdb2  vg02 lvm2 a--  1008.00m 1008.00m
  /dev/sdb3       lvm2 ---     1.00g   1.00g
  /dev/sdb4       lvm2 ---     1.00g   1.00g
```