第一部分　shell

第1章　文件安全与权限

为了防止未授权用户访问你的文件，可以在文件和目录上设置权限位。还可以设定文件在创建时所具有的缺省权限：这些只是整个系统安全问题中的一小部分。在这里我们并不想对系统安全问题的方方面面进行全面的探讨，只是介绍一下有关文件和目录的安全问题。

本章包含以下内容：

- 文件和目录的权限。
- setuid。
- chown和chgrp。
- umask。
- 符号链接。

创建文件的用户和他(她)所属于的组拥有该文件。文件的属主可以设定谁具有读、写、执行该文件的权限。当然，根用户或系统管理员可以改变任何普通用户的设置。一个文件一经创建，就具有三种访问方式：

1) 读，可以显示该文件的内容。
2) 写，可以编辑或删除它。
3) 执行，如果该文件是一个shell脚本或程序。

按照所针对的用户，文件的权限可分为三类：

1) 文件属主，创建该文件的用户。
2) 同组用户，拥有该文件的用户组中的任何用户。
3) 其他用户，即不属于拥有该文件的用户组的某一用户。

1.1　文件

当你创建一个文件的时候，系统保存了有关该文件的全部信息，包括：

- 文件的位置。
- 文件类型。
- 文件长度。
- 哪位用户拥有该文件，哪些用户可以访问该文件。
- i节点。
- 文件的修改时间。
- 文件的权限位。

让我们使用ls -l命令，来看一个典型的文件：

```
$ ls -l
total 4232
-rwxr-xr-x   1 root    root       3756  Oct 14 04:44 dmesg
-r-xr-xr-x   1 root    root      12708 Oct  3 05:40 ps
-rwxr-xr-x   1 root    root       5388  Aug  5  1998 pwd
....
```

下面让我们来分析一下该命令所得结果的前面两行，看看都包含了哪些信息：

total 4232：这一行告诉我们该目录中所有文件所占的空间。

-rwxr-xr-x：这是该文件的权限位。如果除去最前面的横杠，这里一共是 9个字符，他们分别对应9个权限位。通过这些权限位，可以设定用户对文件的访问权限。这 9个字符可以分为三组：

rwx：文件属主权限　　这是前面三位

r-x：同组用户权限　　这是中间三位

r-x：其他用户权限　　这是最后三位

后面我们还将对这些权限位作更详细的介绍。出现在 r、w、x位置上的横杠表示相应的访问权限被禁止。

1　该文件硬链接的数目。

root　文件的属主。

root　文件的属主root所在的缺省组(也叫做root)。

3578　用字节来表示的文件长度，记住，不是K字节！

Oct 14 04:44　文件的更新时间。

dmesg　文件名。

1.2 文件类型

还记得前面一节所提到的文件权限位前面的那个字符吗？我们现在就解释一下这个横杠所代表的意思，文件类型有七种，它可以从 ls -l命令所列出的结果的第一位看出，这七种类型是：

d　目录。

l　符号链接(指向另一个文件)。

s　套接字文件。 socket

b　块设备文件。

c　字符设备文件。

p　命名管道文件。 pipe

-　普通文件，或者更准确地说，不属于以上几种类型的文件。

1.3 权限

让我们用touch命令创建一个文件：

```
$ touch myfile
```

现在对该目录使用ls -l命令：

```
$ ls -l
-rw-r--r--   1 dave      admin           0 Feb 19 22:05 myfile
```

表示普通文件

我们已经创建了一个空文件，正如我们所希望的那样，第一个横杠告诉我们该文件是一个普通文件。你将会发现所创建的文件绝大多数都是普通文件或符号链接文件 (后面将会出现更多的符号链接文件)。

文件属主权限	组用户权限	其他用户权限
rw-	r--	r—

接下来的三个权限位是文件属主所具有的权限；再接下来的三位是与你同组用户所具有的权限，这里是admin组；最后三位是其他用户所具有的权限。在该命令的结果中，我所属于的缺省组也显示了出来。下面是对该文件权限的精确描述：

表1-1 ls -l命令输出的含义

(第一个字符)-	普通文件
(接下来的三个字符)rw-	文件属主的权限
(再接下来的三个字符)r--	同组用户的权限
(最后三个字符)r--	其他用户的权限

因此，这三组字符(除了第一个字符)分别定义了：

1) 文件属主所拥有的权限。

2) 文件属主缺省组(一个用户可以属于很多的组)所拥有的权限。

3) 系统中其他用户的权限。

在每一组字符中含有三个权限位：

r 读权限

w 写/更改权限

x 执行该脚本或程序的权限

这里我们采用另外一种方式来表示刚才所列出 myfile的文件权限：

-	rw-	r--	r--
文件类型为普通文件	文件属主可以读、写	同组用户可以读	其他用户可以读

你可能已经注意到了，myfile在创建的时候并未给属主赋予执行权限，在用户创建文件时，系统不会自动地设置执行权限位。这是出于加强系统安全的考虑。必须手工修改这一权限位：后面讲到umask命令时，你就会明白为什么没有获得执行权限。然而，你可以针对目录设置执行权限位，但这与文件执行权限位的意义有所不同，这一点我们将在后面讨论。

上面这段关于权限位的内容可能不太好理解，让我们来看几个例子 (见表1-2)。

更令人迷惑的是，对于文件属主来说，在只有读权限位被置位的情况下，仍然可以通过文件重定向的方法向该文件写入。过一会儿我们就会看到，能否删除一个文件还依赖于该文件所在目录权限位的设置。

表1-2 文件权限及含义

权 限	所代表的含义
r-- --- ---	文件属主可读，但不能写或执行
r-- r-- ---	文件属主和同组用户(一般来说，是文件属主所在的缺省组)可读
r-- r-- r--	任何用户都可读，但不能写或执行
rwx r-- r--	文件属主可读、写、执行，同组用户和其他用户只可读
rwx r-x ---	文件属主可读、写、执行，同组用户可读、执行

(续)

权　限	所代表的含义
rwx r-x r-x	文件属主可读、写、执行，同组用户和其他用户可读、执行
rw- rw- ---	文件属主和同组用户可读、写
rw- rw- r--	文件属主和同组用户可读、写，其他用户可读
rw- rw- ---	文件属主和同组用户及其他用户读可以读、写，慎用这种权限设置，因为任何用户都可以写入该文件

1.4 改变权限位

对于属于你的文件，可以按照自己的需要改变其权限位的设置。在改变文件权限位设置之前，要仔细地想一想有哪些用户需要访问你的文件 (包括你的目录)。可以使用chmod命令来改变文件权限位的设置。这一命令有比较短的绝对模式和长一些的符号模式。我们先来看一看符号模式。

1.4.1 符号模式

chmod命令的一般格式为：

```
chmod [who] operator [permission] filename
```

who的含义是：

u　文件属主权限。

g　同组用户权限。

o　其他用户权限。

a　所有用户(文件属主、同组用户及其他用户)。

operator的含义：

+　增加权限。

-　取消权限。

=　设定权限。

permission的含义：

r　读权限。

w　写权限。

x　执行权限。

s　文件属主和组set-ID。

t　粘性位*。

l　给文件加锁，使其他用户无法访问。

u,g,o　针对文件属主、同组用户及其他用户的操作。

*在列文件或目录时，有时会遇到“ t ”位。“ t ”代表了粘性位。如果在一个目录上出现“ t ”位，这就意味着该目录中的文件只有其属主才可以删除，即使某个同组用户具有和属主同等的权限。不过有的系统在这一规则上并不十分严格。

如果在文件列表时看到“ t ”，那么这就意味着该脚本或程序在执行时会被放在交换区 (虚存)。不过由于当今的内存价格如此之低，大可不必理会文件的“ t ”的使用。

交换区--虚存

1.4.2 chmod命令举例

现在让我们来看一些使用 chmod命令的例子。假定 myfile文件最初具有这样的权限： rwx rwx rwx ：

命 令	结 果	含 义
chmod a-x myfile	rw- rw- rw-	收回所有用户的执行权限
chmod og-w myfile	rw- r-- r--	收回同组用户和其他用户的写权限
chmod g+w myfile	rw- rw- r--	赋予同组用户写权限
chmod u+x myfile	rwx rw- r--	赋予文件属主执行权限
chmod go+x myfile	rwx rwx r-x	赋予同组用户和其他用户执行权限

当创建myfile文件时，它具有这样的权限：

-rw-r--r-- 1 dave admin 0 Feb 19 22:05 myfile

如果这是我写的一个脚本，我希望能够具有执行权限，并取消其他用户 (所有其他用户)的写权限，可以用： 不能这么用，会把o-w识别为目录或者文件

```
$ chmod u+x o-w myfile
```

这样，该文件的权限变为：

-rwx r-- --- 1 dave admin 0 Feb 19 22:05 myfile

现在已经使文件属主对 myfile文件具有读、写执行的权限，而 admin组的用户对该文件具有读权限。

如果希望某个脚本文件对你自己来说可执行，而且你对该文件的缺省权限很放心，那么只要使它对你来说具有执行权限即可。

```
$ chmod u+x dt
```

1.4.3 绝对模式

chmod命令绝对模式的一般形式为：

```
chmod [mode] file
```

其中mode是一个八进制数。

在绝对模式中，权限部分有着不同的含义。每一个权限位用一个八进制数来代表，如表1-3所示。

表1-3 八进制目录/文件权限表示

八 进 制 数	含 义	八 进 制 数	含 义
0400	文件属主可读	0010	同组用户可执行
0200	文件属主可写	0004	其他用户可读
0100	文件属主可执行	0002	其他用户可写
0040	同组用户可读	0001	其他用户可执行
0020	同组用户可写		

在设定权限的时候，只需按照表 1-3查出与文件属主、同组用户和其他用户所具有的权限相对应的数字，并把它们加起来，就是相应的权限表示。

从表1-3中可以看出，文件属主、同组用户和其他用户分别所能够具有的最大权限值就是7。

再来看看前面举的例子：

-rw-r--r-- 1 dave admin 0 Feb 19 22:05 myfile

相应的权限表示应为644，它的意思就是：

0400+0200(文件属主可读、写) =0600

0040(同组用户可读) =0040

0004(其他用户可读) =0004

0644

有一个计算八进制权限表示的更好办法，如表 1-4所示：

表1-4 计算权限值

文件属主	同组用户	其他用户
r w x	r w x	r w x
4 + 2 + 1	4 + 2 + 1	4 + 2 + 1

使用表1-4，可以更容易地计算出相应的权限值，只要分别针对文件属主、同组用户和其他用户把相应权限下面的数字加在一起就可以了。

myfile文件具有这样的权限：

r w - r - - r - -

4 + 2 4 4

把相应权限位所对应的值加在一起，就是 644。

1.4.4 chmod命令的其他例子

以下是一些chmod命令绝对模式的例子：

命 令	结 果	含 义
chmod 666	rw- rw- rw-	赋予所有用户读和写的权限
chmod 644	rw- r-- r--	赋予所有文件属主读和写的权限，所有其他用户读权限
chmod 744	rwx r-- r--	赋予文件属主读、写和执行的权限，所有其他用户读的权限
chmod 664	rw- rw- r--	赋予文件属主和同组用户读和写的权限，其他用户读权限
chmod 700	rwx --- ---	赋予文件属主读、写和执行的权限
chmod 444	r-- r-- r--	赋予所有用户读权限

下面举一个例子，假定有一个名为 yoa的文件，具有如下权限：

-rw-rw-r-- 1 dave admin 455 Feb 19 22:05 yoa

我现在希望使自己对该文件可读、写和执行， admin组用户对该文件只读，可以键入：

```
$ chmod 740 yoa
-rwxr-- ---  1 dave     admin          455 Feb 19 22:05 yoa
```

如果希望自己对该文件可读、写和执行，对其他所有用户只读，我可以用：

```
$ chmod 744 myfile
-rwxr-- r--  1 dave     admin          0 Feb 19 22:05 myfile
```

如果希望一次设置目录下所有文件的权限，可以用：

```
chmod 644*
```

这将使文件属主和同组用户都具有读和写的权限，其他用户只具有读权限。

还可以通过使用-R选项连同子目录下的文件一起设置：

```
chmod -R 664 /usr/local/home/dave/*
```

这样就可以一次将/usr/local/home/dave目录下的所有文件连同各个子目录下的文件的权限全部设置为文件属主和同组用户可读和写，其他用户只读。使用 -R选项一定要谨慎，只有在需要改变目录树下全部文件权限时才可以使用。

1.4.5 可以选择使用符号模式或绝对模式

上面的例子中既有绝对模式的，也有符号模式的，我们可以从中看出，如果使用该命令的符号模式，可以设置或取消个别权限位，而在绝对模式中则不然。我个人倾向于使用符号模式，因为它比绝对模式方便快捷。

1.5 目录

还记得在前面介绍chmod命令时讲过，目录的权限位和文件有所不同。现在我们来看看其中的区别。目录的读权限位意味着可以列出其中的内容。写权限位意味着可以在该目录中创建文件，如果不希望其他用户在你的目录中创建文件，可以取消相应的写权限位。执行权限位则意味着搜索和访问该目录（见表 1-5、表1-6）。

表1-5 目录权限

r	w	x
可以列出该目录中的文件	可以在该目录中创建或删除文件	可以搜索或进入该目录

表1-6 目录权限举例

权 限	文件属主	同组用户	其他用户
drwx rwx r-x(775)	读、写、执行	读、写、执行	读、执行
drwx r-x r--(754)	读、写、执行	读、执行	读
drwx r-x r-x(755)	读、写、执行	读、执行	读、执行

如果把同组用户或其他用户针对某一目录的权限设置为 --x，那么他们将无法列出该目录中的文件。如果该目录中有一个执行位置位的脚本或程序，只要用户知道它的路径和文件名，仍然可以执行它。用户不能够进入该目录并不妨碍他的执行。

目录的权限将会覆盖该目录中文件的权限。例如，如果目录 docs具有如下的权限：

drwx r-- r-- 1 louise admin 2390 Jul 23 09:44 docs

而其中的文件pay的权限为：

-rwx rwx rwx 1 louise admin 5567 Oct 3 05:40 pay

那么admin组的用户将无法编辑该文件，因为它所属的目录不具有这样的权限。

该文件对任何用户都可读，但由于它所在的目录并未给 admin组的用户赋予执行权限，所以该组的用户都将无法访问该目录，他们将会得到“ 访问受限 ”的错误消息。

1.6 suid/guid

我们在前面曾经提到过suid和guid。这种权限位近年来成为一个棘手的问题。很多系统供

应商不允许实现这一位，或者即使它被置位，也完全忽略它的存在，因为它会带来安全性风险。那么人们为何如此大惊小怪呢？

suid意味着如果某个用户对属于自己的shell脚本设置了这种权限，那么其他用户在执行这一脚本时也会具有其属主的相应权限。于是，如果根用户的某一个脚本设置了这样的权限，那么其他普通用户在执行它的期间也同样具有根用户的权限。同样的原则也适用于 guid，执行相应脚本的用户将具有该文件所属用户组中用户的权限。

1.6.1 为什么要使用suid/guid

为什么要使用这种类型的脚本？这里有一个很好的例子。我管理着几个大型的数据库系统，而对它们进行备份需要有系统管理权限。我写了几个脚本，并设置了它们的 guid，这样我指定的一些用户只要执行这些脚本就能够完成相应的工作，而无须以数据库管理员的身份登录，以免不小心破坏了数据库服务器。通过执行这些脚本，他们可以完成数据库备份及其他管理任务，但是在这些脚本运行结束之后，他们就又回复到他们作为普通用户的权限。

有相当一些UNIX命令也设置了suid和guid。如果想找出这些命令，可以进入 /bin或/sbin目录，执行下面的命令：

```
$ ls -l | grep '^...s'
```

上面的命令是用来查找suid文件的；

```
$ ls -l | grep '^...s..s'
```

上面的命令是用来查找suid和guid的。

现在我们明白了什么是suid，可是如何设置它呢？下面就来介绍这个问题。如果希望设置suid，那么就将相应的权限位之前的那一位设置为4；如果希望设置guid，那么就将相应的权限位之前的那一位设置为2；如果希望两者都置位，那么将相应的权限位之前的那一位设置为4+2。

一旦设置了这一位，一个s将出现在x的位置上。记住：在设置suid或guid的同时，相应的执行权限位必须要被设置。例如，如果希望设置 guid，那么必须要让该用户组具有执行权限。

如果想要对文件 login设置suid，它当前所具有的权限为 rwx rw- r-- (741)，需要在使用chmod命令时在该权限数字的前面加上一个 4，即chmod 4741，这将使该文件的权限变为 rws rw- r--。

```
$ chmod 4741 logit
```

1.6.2 设置suid/guid的例子

下面给出几个例子：

表1-7 设置suid/guid

命 令	结 果	含 义
chmod 4755	rws r-x r-x	文件被设置了suid，文件属主具有读、写和执行的权限，所有其他用户具有读和执行的权限
chmod 6711	rws --s --s	文件被设置了suid和guid，文件属主具有读、写和执行的权限，所有其他用户具有执行的权限
chmod 4764	rws rw- r--	文件被设置了suid，文件属主具有读、写和执行的权限，同组用户具有读和执行的权限，其他用户具有读权限

还可以使用符号方式来设置suid/guid。如果某个文件具有这样的权限：rwx r-x r-x，那么可以这样设置其suid：

```
chmod u+s <filename>
```

于是该文件的权限将变为：rws r-x r-x

在查找设置了suid的文件时，没准会看到具有这样权限的文件：rwS r-x r-x，其中S为大写。它表示相应的执行权限位并未被设置，这是一种没有什么用处的suid设置，可以忽略它的存在。

注意，chmod命令不进行必要的完整性检查，可以给某一个没用的文件赋予任何权限，但chmod 命令并不会对所设置的权限组合做什么检查。因此，不要看到一个文件具有执行权限，就认为它一定是一个程序或脚本。

1.7 chown和chgrp

当你创建一个文件时，你就是该文件的属主。一旦你拥有某个文件，就可以改变它的所有权，把它的所有权交给另外一个 /etc/passwd文件中存在的合法用户。可以使用用户名或用户ID号来完成这一操作。在改变一个文件的所有权时，相应的 suid也将被清除，这是出于安全性的考虑。只有文件的属主和系统管理员可以改变文件的所有权。一旦将文件的所有权交给另外一个用户，就无法再重新收回它的所有权。如果真的需要这样做，那么就只有求助于系统管理员了。

chown命令的一般形式为：

```
chmod -R -h owner file
```

-R选项意味着对所有子目录下的文件也都进行同样的操作。 -h选项意味着在改变符号链接文件的属主时不影响该链接所指向的目标文件。

1.7.1 chown举例

这里给出几个例子：

```
$ ls -l
$ -rwxrwxrwx    1     louise    admin 345 Sep 20 14:33 project
$ chown pauline project
$ ls -l
$ -rwxrwxrwx    1     pauline   admin     Sep 20 14:33 project
```

文件project的所有权现在由用户louise交给了用户pauline。

1.7.2 chgrp举例

chgrp命令和chown命令的格式差不多，下面给出一个例子。

```
-rwxrwxrwx   1   pauline  admin    345 Sep 20 14:33 project
$ chgrp sybadmin project
$ ls -l
$ -rwxrwxrwx 1 pauline   sybadmin 345 Sep 20 14:33 project
```

用户pauline现在把该文件所属的组由admin变为sybadmin（系统中的另外一个用户组）。

1.7.3 找出你所属于的用户组

如果你希望知道自己属于哪些用户组，可以用如下的命令：

```
$ group
$ admin sysadmin appsgen general
```

或者可以使用id命令：

```
$ id
$ uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm)
```

1.7.4 找出其他用户所属于的组

为了找出其他用户所属于的组，可以用如下的命令：

```
$ group matty
$ sybadmin appsgen post
```

上面的命令告诉我们用户matty属于sybadmin、appsgen和post用户组。

1.8 umask

当最初登录到系统中时， umask命令确定了你创建文件的缺省模式。这一命令实际上和chmod命令正好相反。你的系统管理员必须要为你设置一个合理的 umask值，以确保你创建的文件具有所希望的缺省权限，防止其他非同组用户对你的文件具有写权限。

在已经登录之后，可以按照个人的偏好使用 umask命令来改变文件创建的缺省权限。相应的改变直到退出该shell或使用另外的umask命令之前一直有效。

一般来说，umask命令是在/etc/profile文件中设置的，每个用户在登录时都会引用这个文件，所以如果希望改变所有用户的 umask，可以在该文件中加入相应的条目。如果希望永久性地设置自己的umask值，那么就把它放在自己$HOME目录下的.profile或.bash_profile文件中。

1.8.1 如何计算umask值

umask命令允许你设定文件创建时的缺省模式，对应每一类用户 (文件属主、同组用户、其他用户)存在一个相应的umask值中的数字。对于文件来说，这一数字的最大值分别是 6。系统不允许你在创建一个文本文件时就赋予它执行权限，必须在创建后用 chmod命令增加这一权限。目录则允许设置执行权限，这样针对目录来说， umask中各个数字最大可以到7。

该命令的一般形式为：

```
umask nnn
```

其中nnn为umask置000-777。

让我们来看一些例子。

计算出你的umask值：

可以有几种计算umask值的方法，通过设置umask值，可以为新创建的文件和目录设置缺省权限。表1-8列出了与权限位相对应的umask值。

在计算umask值时，可以针对各类用户分别在这张表中按照所需要的文件 /目录创建缺省权限查找对应的umask值。

例如，umask值002 所对应的文件和目录创建缺省权限分别为 664和775。

还有另外一种计算 umask值的方法。我们只要记住 umask是从权限中“ 拿走 ”相应的位即可。

表1-8 umask值与权限

umask	文 件	目 录
0	6	7
1	6	6
2	4	5
3	4	4
4	2	3
5	2	2
6	0	1
7	0	0

例如，对于umask值002，相应的文件和目录缺省创建权限是什么呢？

第一步，我们首先写下具有全部权限的模式，即 777(所有用户都具有读、写和执行权限)。

第二步，在下面一行按照umask值写下相应的位，在本例中是 002。

第三步，在接下来的一行中记下上面两行中没有匹配的位。这就是目录的缺省创建权限。稍加练习就能够记住这种方法。

第四步，对于文件来说，在创建时不能具有文件权限，只要拿掉相应的执行权限比特即可。

这就是上面的例子，其中umask值为002：

1) 文件的最大权限 rwx rwx rwx (777)

2) umask值为002 - - - - - - -w-

3) 目录权限 rwx rwx r-x (775) 这就是目录创建缺省权限

4) 文件权限 rw- rw- r-- (664) 这就是文件创建缺省权限

下面是另外一个例子，假设这次umask值为022：

1) 文件的最大权限 rwx rwx rwx (777)

2) umask值为022 - - - -w- -w-

3) 目录权限 rwx r-x r-x (755) 这就是目录创建缺省权限

4) 文件权限 rw- r-- r-- (644) 这就是文件创建缺省权限

1.8.2 常用的umask值

表1-9列出了一些umask值及它们所对应的目录和文件权限。

表1-9 常用的umask值及对应的文件和目录权限

umask值	目 录	文 件
022	755	644
027	750	640
002	775	664
006	771	660
007	770	660

如果想知道当前的umask 值，可以使用umask命令：

```
$ umask
$ 022
```

```
$ touch file1
$ ls -l file1
$ -rw-r--r--   1 dave      admin              0 Feb 18 42:05 file1
```

如果想要改变umask值，只要使用umask命令设置一个新的值即可：

```
$ umask 002
```

确认一下系统是否已经接受了新的umask值：

```
$ umask
$ 002
$ touch file2
$ ls -l file2
$ -rw-rw-r--   1 dave      admin              0 Feb 18 45:07 file2
```

在使用umask命令之前一定要弄清楚到底希望具有什么样的文件 /目录创建缺省权限。否则可能会得到一些非常奇怪的结果；例如，如果将 umask值设置为600，那么所创建的文件/目录的缺省权限就是066！

1.9 符号链接

存在两种不同类型的链接，软链接和硬链接，这里我们只讨论软链接。软链接实际上就是一个指向文件的指针。你将会发现这种软链接使用起来非常方便。

1.9.1 使用软链接来保存文件的多个映像

下面我们就解释一下符号链接是怎么回事。比方说在 /usr/local/admin/sales目录下有一个含有销售信息的文件，销售部门的每一个人都想看这份文件。你可以在每一位用户的 $HOME目录下建立一个指向该文件的链接，而不是在每个目录下拷贝一份。这样当需要更改这一文件时，只需改变一个源文件即可。每个销售 $HOME目录中的链接可以起任何名字，不必和源文件一致。

如果有很多子目录，而进入这些目录很费时间，在这种情况下链接也非常有用。可以针对$HOME目录下的一个很深的子目录创建一个链接。还有，比如在安装一个应用程序时，它的日志被保存到/usr/opt/app/log目录下，如果想把它保存在另外一个你认为更方便目录下，可以建立一个指向该目录的链接。

该命令的一般形式为：

```
ln [-s] source_path target_path
```

其中的路径可以是目录也可以是文件。让我们来看几个例子。

1.9.2 符号链接举例

假如系统中有40个销售和管理用户，销售用户使用一个销售应用程序，而管理用户使用一个管理应用程序。我作为系统管理员该怎么做呢？首先删除它们各自 $HOME目录下的所有.profile文件。然后在/usr/local/menus/目录下创建两个profile文件，一个是sales.profile，一个是admin.profile，它们分别为销售和管理人员提供了所需的环境，并引导他们进入相应的应用程序。现在我在所有销售人员的 $HOME目录下分别创建一个指向sales.profile的链接，在所有管理人员的$HOME目录下分别创建一个指向 admin.profile文件的链接。注意，不必在上面命令格式中的target_path端创建相应文件，如果不存在这样一个文件， ln命令会自动创建该文

件。下面就是我对销售人员matty所做的操作。

```
$ cd /home/sales/matty
$ rm.profile
$ ln -s /usr/local/menus/sales.profile. profile
$ ls -al.profile
$ lrwx rwx  rwx   1  sales    admin 5567 Oct  3 05:40.profile->
/usr/local/menus/sales.profile
```

(你所看到的可能会与此稍有差别)。

这就是我所要做的全部工作；对于管理人员也是如此。而且如果需要作任何修改的话，只要改变销售和管理人员的profile文件即可，而不必对40个用户逐一进行修改。

下面是另外一个例子。我所管理的系统中有一个网络监视器，它将日志写在 /usr/opt/monitor/regstar目录下，但其他所有的日志都保存在 /var/adm/logs目录下，这样只需在该目录下建立一个指向原有文件的链接就可以在一个地方看所有的日志了，而不必花费很多时间分别进入各个相应的目录。下面就是所用的链接命令：

```
$ ln -s /usr/opt/monitor/regstar/reg.log /var/adm/logs/monitor.log
```

如果链接太多的话，可以删掉一些，不过切记不要删除源文件。

不管是否在同一个文件系统中，都可以创建链接。在创建链接的时候，不要忘记在原有目录设置执行权限。链接一旦创建，链接目录将具有权限 777或rwx rwx rwx，但是实际的原有文件的权限并未改变。

在新安装的系统上，通常要进行这样的操作，在/var目录中创建一个指向/tmp目录的链接，因为有些应用程序认为存在 /var/tmp目录(然而它实际上并不存在)，有些应用程序在该目录中保存一些临时文件。为了使所有的临时文件都放在一个地方，可以使用 ln命令在/var目录下建立一个指向/tmp目录的链接。

```
$ pwd /var
$ ln -s /tmp  /var/tmp
```

现在如果我在/var目录中列文件，就能够看到刚才建立的链接：

```
$ ls -l
$ lrwx rwx  rwx   1  root    root      5567 Sep  9  10:40 tmp->
/tmp
```

1.10 小结

本章介绍了一些有关文件安全的基本概念。如果这些命令能够使用得当而且使用得比较谨慎，应该不会有什么问题。手指轻轻一敲就有可能输入 chmod -R这样的命令，它将改变整个文件系统的权限，如果没有做备份的话，没有几年的时间恐怕是无法恢复了，所以在输入这些命令时，千万不要在手指上贴膏药！

是否使用设置了suid的脚本完全取决于你自己。如果使用的话，一定要确保能够监控它的使用，而且不要以根用户身份设置 suid。