

鸟哥的 Linux 私房菜

为取得较佳浏览结果, 请爱用 [firefox](#) 浏览本网页



第七章、Linux 档案与目录管理

最近更新日期: 2008/09/29

在第六章我们认识了 Linux 系统下的档案权限概念以及目录的配置说明。在这个章节当中, 我们就直接来进一步的操作与管理档案与目录吧! 包括在不同的目录间变换、建立与删除目录、建立与删除档案, 还有寻找档案、查阅档案内容等等, 都会在这个章节作个简单的介绍啊!

1. 目录与路径
 - 1.1 相对路径与绝对路径
 - 1.2 目录的相关操作: `cd`, `pwd`, `mkdir`, `rmdir`
 - 1.3 关于执行文件路径的变量: `$PATH`
2. 档案与目录管理
 - 2.1 档案与目录的检视: `ls`
 - 2.2 复制、删除与移动: `cp`, `rm`, `mv`
 - 2.3 取得路径的文件名与目录名称
3. 档案内容查阅:
 - 3.1 直接检视档案内容: `cat`, `tac`, `nl`
 - 3.2 可翻页检视: `more`, `less`
 - 3.3 资料撷取: `head`, `tail`
 - 3.4 非纯文本档: `od`
 - 3.5 修改档案时间与建置新档: `touch`
4. 档案与目录的默认权限与隐藏权限
 - 4.1 档案预设权限: `umask`
 - 4.2 档案隐藏属性: `chattr`, `lsattr`
 - 4.4 档案特殊权限: `SUID`, `SGID`, `SBIT`, 权限设定
 - 4.3 观察文件类型: `file`
5. 指令与档案的搜寻:
 - 5.1 脚本文件名的搜寻: `which`
 - 5.2 档案档名的搜寻: `whereis`, `locate`, `find`
6. 极重要! 权限与指令间的关系:
7. 重点回顾
8. 本章习题
9. 参考数据与延伸阅读
10. 针对本文的建议: <http://phorum.vbird.org/viewtopic.php?t=23879>



目录与路径:

由第六章 [Linux 的档案权限与目录配置](#) 中透过 FHS 了解了 Linux 的『树状目录』概念之后，接下来就得要实际的来搞定一些基本的路径问题了！这些目录的问题当中，最重要的莫过于第六章也谈过的『绝对路径』与『相对路径』的意义啦！绝对/相对路径的写法并不相同，要特别注意。此外，当妳下达指令时，该指令是透过什么功能来取得的？这与 PATH 这个变数有关呢！底下就让我们来谈谈啰！

相对路径与绝对路径：

在开始目录的切换之前，你必须要先了解一下所谓的『路径(PATH)』，有趣的是：什么是『相对路径』与『绝对路径』？虽然前一章已经稍微针对这个议题提过一次，不过，这里不厌其烦的再次的强调一下！

- 绝对路径：路径的写法『一定由根目录 / 写起』，例如： /usr/share/doc 这个目录。
- 相对路径：路径的写法『不是由 / 写起』，例如由 /usr/share/doc 要到 /usr/share/man 底下时，可以写成：『cd ../man』这就是相对路径的写法啦！相对路径意指『相对于目前工作目录的路径！』
- 相对路径的用途

那么相对路径与绝对路径有什么了不起呀？喝！那可真的是了不起了！假设你写了一个套件，这个套件共需要三个目录，分别是 etc, bin, man 这三个目录，然而由于不同的人喜欢安装在不同的目录之下，假设甲安装的目录是 /usr/local/packages/etc, /usr/local/packages/bin 及 /usr/local/packages/man，不过乙却喜欢安装在 /home/packages/etc, /home/packages/bin, /home/packages/man 这三个目录中，请问如果需要用到绝对路径的话，那么是否很麻烦呢？是的！如此一来每个目录下的东西就很难对应的起来！这个时候相对路径的写法就显的特别的重要了！

此外，如果你跟鸟哥一样，喜欢将路径的名字写的很长，好让自己知道那个目录是在干什么的，例如： /cluster/raid/output/taiwan2006/smoke 这个目录，而另一个目录在 /cluster/raid/output/taiwan2006/cctm，那么我从第一个要到第二个目录去的话，怎么写比较方便？当然是『cd ../cctm』比较方便啰！对吧！

- 绝对路径的用途

但是对于档名的正确性来说，『绝对路径的正确度要比较好～』。一般来说，鸟哥会建议你，如果是在写程序(shell scripts)的条件下，务必使用绝对路径的写法。怎么说呢？因为绝对路径的写法虽然比较麻烦，但是可以肯定这个写法绝对不会有问题。如果使用相对路径在程序当中，则可能由于你执行的工

作环境不同，导致一些问题的发生。这个问题在[例行性命令 \(at, cron\)](#)当中尤其重要！这个现象我们在 [shell script](#) 时，会再次的提醒你喔！ ^_^

🐼 目录的相关操作：

我们之前稍微提到变换目录的指令是 `cd`，还有哪些可以进行目录操作的指令呢？例如建立目录啊、删除目录之类的～还有，得要先知道的，就是有哪些比较特殊的目录呢？举例来说，底下这些就是比较特殊的目录，得要用的记下来才行：

```
.      代表此层目录
..     代表上一层目录
-      代表前一个工作目录
~      代表『目前用户身份』所在的家目录
~account 代表 account 这个用户的家目录(account 是个账号名称)
```

需要特别注意的是：在所有目录底下都会存在的两个目录，分别是『`.`』与『`..`』分别代表此层与上层目录的意思。那么来思考一下底下这个例题：

例题：

请问在 Linux 底下，根目录下有没有上层目录(`..`)存在？

答：

若使用『`ls -al /`』去查询，可以看到根目录下确实存在 `.` 与 `..` 两个目录，再仔细的查阅，可发现这两个目录的属性与权限完全一致，这代表根目录的上一层(`..`)与根目录自己(`.`)是同一个目录。

底下我们就来谈一谈几个常见的处理目录的指令吧：

- `cd`：变换目录
 - `pwd`：显示当前目录
 - `mkdir`：建立一个新的目录
 - `rmdir`：删除一个空的目录
-

- `cd` (变换目录)

我们知道 `vbird` 这个用户的家目录是 `/home/vbird/`，而 `root` 家目录则是 `/root/`，假设我以 `root` 身份在 Linux 系统中，那么简单的说明一下这几个特殊的目录的意义是：

```
[root@www ~]# cd [相对路径或绝对路径]
# 最重要的就是目录的绝对路径与相对路径，还有一些特殊
# 目录的符号啰！
[root@www ~]# cd ~vbird
# 代表去到 vbird 这个用户的家目录，亦即 /home/vbird
[root@www vbird]# cd ~
# 表示回到自己的家目录，亦即是 /root 这个目录
[root@www ~]# cd
# 没有加上任何路径，也还是代表回到自己家目录的意思
# 喔！
[root@www ~]# cd ..
# 表示去到目前的上层目录，亦即是 /root 的上层目录的
# 意思；
[root@www /]# cd -
# 表示回到刚刚的那个目录，也就是 /root 啰~
[root@www ~]# cd /var/spool/mail
# 这个就是绝对路径的写法！直接指定要去的完整路径名
# 称！
[root@www mail]# cd ../mqueue
# 这个是相对路径的写法，我们由/var/spool/mail 去到
# /var/spool/mqueue 就这样写！
```

cd 是 Change Directory 的缩写，这是用来变换工作目录的指令。注意，目录名称与 cd 指令之间存在一个空格。一登入 Linux 系统后，root 会在 root 的家目录！那回到上一层目录可以用『 cd.. 』。利用相对路径的写法必须要确认你目前的路径才能正确的去到想要去的目录。例如上表当中最后一个例子，你必须确认你是在/var/spool/mail 当中，并且知道在/var/spool 当中有个mqueue 的目录才行啊~ 这样才能使用 cd../mqueue 去到正确的目录说，否则就要直接输入 cd /var/spool/mqueue 啰~

其实，我们的提示字符，亦即那个 [root@www ~]# 当中，就已经有指出当前目录了，刚登入时会到自己的家目录，而家目录还有一个代码，那就是『 ~ 』符号！例如上面的例子可以发现，使用『 cd ~ 』可以回到个人的家目录里头去呢！另外，针对 cd 的使用方法，如果仅输入 cd 时，代表的就是『 cd ~ 』的意思喔~ 亦即是会回到自己的家目录啦！而那个『 cd - 』比较难以理解，请自行多做几次练习，就会比较明白了。

Tips:

还是要一再地提醒，我们的 Linux 的默认指令列模式 (bash shell) 具有档案补齐功能，你要常常利用 [tab] 按键来达成你的目录完整性啊！这可是个好习惯啊~ 可以避免你按错键盘输入错字说~ ^_^



- pwd (显示目前所在的目录)

```
[root@www ~]# pwd [-P]
选项:
-P : 显示出确实的路径, 而非使用链接 (link) 路径。

范例: 单纯显示出目前的工作目录:
[root@www ~]# pwd
/root    <== 显示出目录啦~

范例: 显示出实际的工作目录, 而非链接文件本身的目录名而已
[root@www ~]# cd /var/mail    <==注意, /var/mail 是一个连结档
[root@www mail]# pwd
/var/mail    <==列出目前的工作目录
[root@www mail]# pwd -P
/var/spool/mail    <==怎么回事? 有没有加 -P 差很多~
[root@www mail]# ls -ld /var/mail
lrwxrwxrwx 1 root root 10 Sep  4 17:54 /var/mail ->
spool/mail
# 看到这里应该知道为啥了吧? 因为 /var/mail 是连结
档, 连结到 /var/spool/mail
# 所以, 加上 pwd -P 的选项后, 会不以连结文件的数据显示,
而是显示正确的完整路径啊!
```

pwd 是 Print Working Directory 的缩写, 也就是显示目前所在目录的指令, 例如在上个表格最后的目录是/var/mail 这个目录, 但是提示字符仅显示 mail, 如果你想要知道目前所在的目录, 可以输入 pwd 即可。此外, 由于很多的套件所使用的目录名称都相同, 例如 /usr/local/etc 还有/etc, 但是通常 Linux 仅列出最后面那一个目录而已, 这个时候你就可以使用 pwd 来知道你的所在目录啰! 免得搞错目录, 结果...

其实有趣的是那个 -P 的选项啦! 他可以让我们取得正确的目录名称, 而不是以链接文件的路径来显示的。如果你使用的是 CentOS 5.x 的话, 刚刚好/var/mail 是/var/spool/mail 的连结档, 所以, 透过到/var/mail 下达 pwd -P 就能够知道这个选项的意义啰~ ^_^

-
- mkdir (建立新目录)

```

[root@www ~]# mkdir [-mp] 目录名称
选项：
-m : 配置文件案的权限喔！直接设定，不需要看预设权限
      (umask) 的脸色～
-p : 帮助你直接将所需要的目录(包含上层目录)递归建立
      起来！

范例：请到/tmp 底下尝试建立数个新目录看看：
[root@www ~]# cd /tmp
[root@www tmp]# mkdir test    <==建立一名为 test 的
      新目录
[root@www tmp]# mkdir test1/test2/test3/test4
mkdir: cannot create directory
`test1/test2/test3/test4':
No such file or directory    <== 没办法直接建立此
      目录啊！
[root@www tmp]# mkdir -p test1/test2/test3/test4
# 加了这个 -p 的选项，可以自行帮你建立多层目录！

范例：建立权限为 rwx--x--x 的目录
[root@www tmp]# mkdir -m 711 test2
[root@www tmp]# ls -l
drwxr-xr-x  3 root  root 4096 Jul 18 12:50 test
drwxr-xr-x  3 root  root 4096 Jul 18 12:53 test1
drwx--x--x  2 root  root 4096 Jul 18 12:54 test2
# 仔细看上面的权限部分，如果没有加上 -m 来强制设定属
      性，系统会使用默认属性。
# 那么你的默认属性为何？这要透过底下介绍的 umask 才
      能了解喔！ ^_^

```

如果想要建立新的目录的话，那么就使用 mkdir (make directory) 吧！不过，在预设的情况下，你所需要的目录得一层一层的建立才行！例如：假如你要建立一个目录为 /home/bird/testing/test1，那么首先必须要有 /home 然后 /home/bird，再来 /home/bird/testing 都必须要有存在，才可以建立 /home/bird/testing/test1 这个目录！假如没有 /home/bird/testing 时，就没有办法建立 test1 的目录啰！

不过，现在有个更简单有效的方法啦！那就是加上 -p 这个选项喔！你可以直接下达：『mkdir -p /home/bird/testing/test1』则系统会自动的帮你将 /home, /home/bird, /home/bird/testing 依序的建立起目录！并且，如果该目录本来就已经存在时，系统也不会显示错误讯息喔！挺快乐的吧！^_^。不过鸟哥不建议常用 -p 这个选项，因为担心如果妳打错字，那么目录名称就会变的乱七八糟的！

另外，有个地方你必须要先有概念，那就是『预设权限』的地方。我们可以利用 `-m` 来强制给予一个新的目录相关的权限，例如上表当中，我们给予 `-m 711` 来给予新的目录 `drwx--x--x` 的权限。不过，如果没有给予 `-m` 选项时，那么默认的新建目录权限又是什么呢？这个跟 `umask` 有关，我们在本章后头会加以介绍的。

- `rmdir` (删除『空』的目录)

```
[root@www ~]# rmdir [-p] 目录名称
选项：
-p : 连同上层『空的』目录也一起删除

范例：将于 mkdir 范例中建立的目录(/tmp 底下)删除掉！
[root@www tmp]# ls -l    <==看看有多少目录存在？
drwxr-xr-x  3 root  root 4096 Jul 18 12:50 test
drwxr-xr-x  3 root  root 4096 Jul 18 12:53 test1
drwx--x--x  2 root  root 4096 Jul 18 12:54 test2
[root@www tmp]# rmdir test    <==可直接删除掉，没问题
[root@www tmp]# rmdir test1  <==因为尚有内容，所以无法删除！
rmdir: `test1': Directory not empty
[root@www tmp]# rmdir -p test1/test2/test3/test4
[root@www tmp]# ls -l          <==您看看，底下的输出中
test 与 test1 不见了！
drwx--x--x  2 root  root 4096 Jul 18 12:54 test2
# 瞧！利用 -p 这个选项，立刻就可以将
test1/test2/test3/test4 一次删除～
# 不过要注意的是，这个 rmdir 仅能『删除空的目录』喔！
```

如果想要删除旧有的目录时，就使用 `rmdir` 吧！例如将刚刚建立的 `test` 杀掉，使用『`rmdir test`』即可！请注意哟！目录需要一层一层的删除才行！而且被删除的目录里面必定不能存在其他的目录或档案！这也是所谓的空的目录 (empty directory) 的意思啊！那如果要将所有目录下的东西都杀掉呢？！这个时候就必须使用『`rm -r test`』啰！不过，还是使用 `rmdir` 比较不危险！你也可以尝试以 `-p` 的选项加入，来删除上层的目录喔！

💡关于执行文件路径的变量： `$PATH`

经过第六章 FHS 的说明后，我们知道查阅文件属性的指令 `ls` 完整文件名为：
`/bin/ls`(这是绝对路径)，那你会不会觉得很奇怪：『为什么我可以在任何地方

执行/bin/ls 这个指令呢？』为什么我在任何目录下输入 ls 就一定可以显示出一些讯息而不会说找不到该 /bin/ls 指令呢？这是因为环境变量 PATH 的帮助所致呀！

当我们在执行一个指令的时候，举例来说『ls』好了，系统会依照 PATH 的设定去每个 PATH 定义的目录下搜寻文件名为 ls 的可执行文件，如果在 PATH 定义的目录中含有多个文件名为 ls 的可执行文件，那么先搜寻到的同名指令先被执行！

现在，请下达『echo \$PATH』来看看到底有哪些目录被定义出来了？echo 有『显示、印出』的意思，而 PATH 前面加的 \$ 表示后面接的是变量，所以会显示出目前的 PATH ！

范例：先用 root 的身份列出搜寻的路径为何？

```
[root@www ~]# echo $PATH
/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin
:/sbin
:/bin:/usr/sbin:/usr/bin:/root/bin <==这是同一行！
```

范例：用 vbird 的身份列出搜寻的路径为何？

```
[root@www ~]# su - vbird
[vbird@www ~]# echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/home/vbird/bin
# 仔细看，一般用户 vbird 的 PATH 中，并不包含任何『sbin』的目录存在喔！
```

PATH(一定是大写)这个变量的内容是由一堆目录所组成的，每个目录中间用冒号(:)来隔开，每个目录是有『顺序』之分的。仔细看一下上面的输出，妳可以发现到无论是 root 还是 vbird 都有/bin 这个目录在 PATH 变量内，所以当然能够在任何地方执行 ls 来找到/bin/ls 执行档啰！

我们用几个范例来让你了解一下，为什么 PATH 是那么重要的项目！

例题：

请问你能不能使用一般身份使用者下达 ifconfig eth0 这个指令呢？

答：

如上面的范例所示，当你使用 vbird 这个账号执行 ifconfig 时，会出现『-bash: ifconfig: command not found』的字样，因为 ifconfig 的是放置到/sbin 底下，而由上表的结果中我们可以发现 vbird 的 PATH 并没有设置/sbin，所以预设无法执行。

但是你可以使用『/sbin/ifconfig eth0』来执行这个指令喔！因为一般用户还是可以使用 ifconfig 来查询系统 IP 的参数，既然 PATH 没有规范到/sbin，那么我们使用『绝对路径』也可以执行到该指令的！

例题：

假设你是 root，如果你将 ls 由 /bin/ls 移动成为 /root/ls (可用『mv /bin/ls /root』指令达成)，然后你自己本身也在 /root 目录下，请问 (1) 你能不能直接输入 ls 来执行？(2) 若不能，你该如何执行 ls 这个指令？(3) 若要直接输入 ls 即可执行，又该如何进行？

答：

由于这个例题的重点是将某个执行文件移动到非正规目录去，所以我们先要进行底下的动作才行：(务必使用 root 的身份)

```
[root@www ~]# mv /bin/ls /root
# mv 为移动，可将档案在不同的目录间进行移动作业
```

(1) 接下来不论你在那个目录底下输入任何与 ls 相关的指令，都没有办法顺利的执行 ls 了！也就是说，你不能直接输入 ls 来执行，因为 /root 这个目录并不在 PATH 指定的目录中，所以，即使你在 /root 目录下，也不能够搜寻到 ls 这个指令！

(2) 因为这个 ls 确实存在于 /root 底下，并不是被删除了！所以我们可以透过使用绝对路径或者是相对路径直接指定这个执行档档名，底下的两个方法都能够执行 ls 这个指令：

```
[root@www ~]# /root/ls <==直接用绝对路径指定该文件名
[root@www ~]# ./ls <==因为在 /root 目录下，就用 ./ls 来指定
```

(3) 如果想要让 root 在任何目录均可执行 /root 底下的 ls，那么就将 /root 加入 PATH 当中即可。加入的方法很简单，就像底下这样：

```
[root@www ~]# PATH="$PATH":/root
```

上面这个作法就能够将 /root 加入到执行文件搜寻路径 PATH 中了！不相信的话请您自行使用『echo \$PATH』去查看吧！如果确定这个例题进行没有问题了，请将 ls 搬回 /bin 底下，不然系统会挂点的！

```
[root@www ~]# mv /root/ls /bin
```

例题：

如果我有两个 ls 指令在不同的目录中，例如 /usr/local/bin/ls 与 /bin/ls 那么当我下达 ls 的时候，哪个 ls 会被执行？

答：

那还用说，就找出 PATH 里面哪个目录先被查询，则那个目录下的指令就会被先执行了！

例题：

为什么 PATH 搜寻的目录不加入本目录(.)？加入本目录的搜寻不是也

不错？

答：

如果在 PATH 中加入本目录(.)后，确实我们就能够在指令所在目录进行指令的执行了。但是由于你的工作目录并非固定(常常会使用 cd 来切换到不同的目录)，因此能够执行的指令会有变动(因为每个目录底下的可执行文件都不相同嘛！)，这对使用者来说并非好事。

另外，如果有个坏心使用者在/tmp 底下做了一个指令，因为/tmp 是大家都能够写入的环境，所以他当然可以这样做。假设该指令可能会窃取用户的一些数据，如果你使用 root 的身份来执行这个指令，那不是很糟糕？如果这个指令的名称又是经常会被用到的 ls 时，那『中标』的机率就更高了！

所以，为了安全起见，不建议将『.』加入 PATH 的搜寻目录中。

而由上面的几个例题我们也可以知道几件事情：

- 不同身份使用者预设的 PATH 不同，默认能够随意执行的指令也不同(如 root 与 vbird)；
- PATH 是可以修改的，所以一般使用者还是可以透过修改 PATH 来执行某些位于/sbin 或/usr/sbin 下的指令来查询；
- 使用绝对路径或相对路径直接指定某个指令的文件名来执行，会比搜寻 PATH 来的正确；
- 指令应该要放置到正确的目录下，执行才会比较方便；
- 本目录(.)最好不要放到 PATH 当中。

对于 PATH 更详细的『变量』说明，我们会在第三篇的 [bash shell](#) 中详细说的！



档案与目录管理：

谈了谈目录与路径之后，再来讨论一下关于档案的一些基本管理吧！档案与目录的管理上，不外乎『显示属性』、『拷贝』、『删除档案』及『移动档案或目录』等等，由于档案与目录的管理在 Linux 当中是很重要的，尤其是每个人自己家目录的数据也都需要注意管理！所以我们来谈一谈有关档案与目录的一些基础管理部分吧！



档案与目录的检视： ls

```
[root@www ~]# ls [-aAdfFhilnrRSt] 目录名称
```

```
[root@www ~]# ls [--color={never,auto,always}] 目录名称
[root@www ~]# ls [--full-time] 目录名称
选项与参数:
-a : 全部的档案, 连同隐藏档( 开头为 . 的档案) 一起列出来(常用)
-A : 全部的档案, 连同隐藏档, 但不包括 . 与 .. 这两个目录
-d : 仅列出目录本身, 而不是列出目录内的档案数据(常用)
-f : 直接列出结果, 而不进行排序 (ls 预设会以档名排序!)
-F : 根据档案、目录等信息, 给予附加数据结构, 例如:
    *:代表可执行文件;  /:代表目录;  =:代表 socket 档案;  |:代表 FIFO 档案;
-h : 将档案容量以人类较易读的方式(例如 GB, KB 等等)列出来;
-i : 列出 inode 号码, inode 的意义下一章将会介绍;
-l : 长数据串行出, 包含档案的属性与权限等等数据; (常用)
-n : 列出 UID 与 GID 而非使用者与群组的名称 (UID 与 GID 会在账号管理提到!)
-r : 将排序结果反向输出, 例如: 原本档名由小到大, 反向则为由大到小;
-R : 连同子目录内容一起列出来, 等于该目录下的所有档案都会显示出来;
-S : 以档案容量大小排序, 而不是用档名排序;
-t : 依时间排序, 而不是用档名。
--color=never : 不要依据档案特性给予颜色显示;
--color=always : 显示颜色
--color=auto : 让系统自行依据设定来判断是否给予颜色
--full-time : 以完整时间模式 (包含年、月、日、时、分) 输出
--time={atime,ctime} : 输出 access 时间或改变权限属性时间 (ctime)
                        而非内容变更时间
(modification time)
```

在 Linux 系统当中, 这个 ls 指令可能是最常被执行的吧! 因为我们随时都要知道档案或者是目录的相关信息啊~ 不过, 我们 Linux 的档案所记录的信息实在是太多了, ls 没有需要全部都列出来呢~ 所以, 当你只有下达 ls 时, 默认显示的只有: 非隐藏档的档名、以档名进行排序及文件名代表的颜色显示如此而

已。举例来说，你下达『ls/etc』之后，只有经过排序的文件名以及以蓝色显示目录及白色显示一般档案，如此而已。

那如果我还想要加入其他的显示信息时，可以加入上头提到的那些有用的选项呢～举例来说，我们之前一直用到的 -l 这个长串显示数据内容，以及将隐藏档也一起列示出来的 -a 选项等等。底下则是一些常用的范例，实际试做看看：

范例一：将家目录下的所有档案列出来(含属性与隐藏文件)

```
[root@www ~]# ls -al ~
total 156
drwxr-x---  4 root root  4096 Sep 24 00:07 .
drwxr-xr-x 23 root root  4096 Sep 22 12:09 ..
-rw-----  1 root root 1474 Sep  4 18:27
anaconda-ks.cfg
-rw-----  1 root root   955 Sep 24
00:08 .bash_history
-rw-r--r--  1 root root    24 Jan  6
2007 .bash_logout
-rw-r--r--  1 root root   191 Jan  6
2007 .bash_profile
-rw-r--r--  1 root root   176 Jan  6 2007 .bashrc
drwx-----  3 root root  4096 Sep  5 10:37 .gconf
-rw-r--r--  1 root root 42304 Sep  4 18:26 install.log
-rw-r--r--  1 root root  5661 Sep  4 18:25
install.log.syslog
# 这个时候你会看到以 . 为开头的几个档案，以及目录文件
# (.) (..) .gconf 等等，
# 不过，目录文件文件名都是以深蓝色显示，有点不容易看清楚就是了。
```

范例二：承上题，不显示颜色，但在文件名末显示出该文件名代表的类型(type)

```
[root@www ~]# ls -alF --color=never ~
total 156
drwxr-x---  4 root root  4096 Sep 24 00:07 ./
drwxr-xr-x 23 root root  4096 Sep 22 12:09 ../
-rw-----  1 root root 1474 Sep  4 18:27
anaconda-ks.cfg
-rw-----  1 root root   955 Sep 24
00:08 .bash_history
-rw-r--r--  1 root root    24 Jan  6
2007 .bash_logout
-rw-r--r--  1 root root   191 Jan  6
2007 .bash_profile
```

```

-rw-r--r--  1 root root   176 Jan  6  2007 .bashrc
drwx-----  3 root root  4096 Sep  5 10:37 .gconf/
-rw-r--r--  1 root root 42304 Sep  4 18:26 install.log
-rw-r--r--  1 root root  5661 Sep  4 18:25
install.log.syslog
# 注意看到显示结果的第一行，嘿嘿～知道为何我们会下达
类似 ./command
# 之类的指令了吧？因为 ./ 代表的是『目前目录下』的意
思啊！至于什么是 FIFO/Socket ？
# 请参考前一章节的介绍啊！另外，那个.bashrc 时间仅写
2007，能否知道详细时间？

范例三：完整的呈现档案的修改时间 *(modification time)
[root@www ~]# ls -al --full-time ~
total 156
drwxr-x---  4 root root  4096 2008-09-24
00:07:00.000000 +0800 .
drwxr-xr-x 23 root root  4096 2008-09-22
12:09:32.000000 +0800 ..
-rw-----  1 root root  1474 2008-09-04
18:27:10.000000 +0800 anaconda-ks.cfg
-rw-----  1 root root   955 2008-09-24
00:08:14.000000 +0800 .bash_history
-rw-r--r--  1 root root    24 2007-01-06
17:05:04.000000 +0800 .bash_logout
-rw-r--r--  1 root root   191 2007-01-06
17:05:04.000000 +0800 .bash_profile
-rw-r--r--  1 root root   176 2007-01-06
17:05:04.000000 +0800 .bashrc
drwx-----  3 root root  4096 2008-09-05
10:37:49.000000 +0800 .gconf
-rw-r--r--  1 root root 42304 2008-09-04
18:26:57.000000 +0800 install.log
-rw-r--r--  1 root root  5661 2008-09-04
18:25:55.000000 +0800 install.log.syslog
# 请仔细看，上面的『时间』字段变了喔！变成较为完整的
格式。
# 一般来说， ls -al 仅列出目前短格式的时间，有时不会
列出年份，
# 藉由 --full-time 可以查阅到比较正确的完整时间格式
啊！

```

其实 ls 的用法还有很多，包括查阅档案所在 i-node 号码的 ls -i 选项，以

及用来进行档案排序的 `-S` 选项，还有用来查阅不同时间的动作的 `--time=atime` 等选项(更多时间说明请参考本章后面 [touch](#) 的说明)。而这些选项的存在都是因为 Linux 文件系统记录了很多有用的信息的缘故。那么 Linux 的文件系统中，这些与权限、属性有关的数据放在哪里呢？放在 `i-node` 里面。关于这部分，我们会在下一章继续为你作比较深入的介绍啊！

无论如何，`ls` 最常被使用到的功能还是那个 `-l` 的选项，为此，很多 `distribution` 在预设的情况下，已经将 `ll` (`l` 的小写) 设定成为 `ls -l` 的意思了！其实，那个功能是 [Bash shell](#) 的 [alias](#) 功能呢～也就是说，我们直接输入 `ll` 就等于是输入 `ls -l` 是一样的～关于这部分，我们会在后续 `bash shell` 时再次的强调滴～

🐼复制、删除与移动：`cp`, `rm`, `mv`

要复制档案，请使用 `cp` (`copy`) 这个指令即可～不过，`cp` 这个指令的用途可多了～除了单纯的复制之外，还可以建立连结档（就是快捷方式啰），比对两档案的新旧而予以更新，以及复制整个目录等等的功能呢！至于移动目录与档案，则使用 `mv` (`move`)，这个指令也可以直接拿来作更名 (`rename`) 的动作喔！至于移除吗？那就是 `rm` (`remove`) 这个指令啰～底下我们就来瞧一瞧先～

-
- `cp` (复制档案或目录)

```
[root@www ~]# cp [-adfilprsu] 来源文件(source) 目标文件(destination)
[root@www ~]# cp [options] source1 source2 source3 ....
directory
选项与参数：
-a : 相当于 -pdr 的意思，至于 pdr 请参考下列说明；(常用)
-d : 若来源文件为链接文件的属性(link file)，则复制链接文件属性而非档案本身；
-f : 为强制(force)的意思，若目标档案已经存在且无法开启，则移除后再尝试一次；
-i : 若目标文件(destination)已经存在时，在覆盖时会先询问动作的进行(常用)
-l : 进行硬式连结(hard link)的连结档建立，而非复制档案本身；
-p : 连同档案的属性一起复制过去，而非使用默认属性(备份常用)；
-r : 递归持续复制，用于目录的复制行为；(常用)
```

```
-s : 复制成为符号链接文件 (symbolic link), 亦即『快捷方式』档案;  
-u : 若 destination 比 source 旧才更新 destination !  
最后需要注意的, 如果来源档有两个以上, 则最后一个目的文件一定要是『目录』才行!
```

复制(cp)这个指令是非常重要的, 不同身份者执行这个指令会有不同的结果产生, 尤其是那个-a, -p 的选项, 对于不同身份来说, 差异则非常的大! 底下的练习中, 有的身份为root 有的身份为一般账号(在我这里用vbird这个账号), 练习时请特别注意身份的差别喔! 好! 开始来做复制的练习与观察:

```
范例一: 用 root 身份, 将家目录下的 .bashrc 复制到 /tmp  
下, 并更名为 bashrc  
[root@www ~]# cp ~/.bashrc /tmp/bashrc  
[root@www ~]# cp -i ~/.bashrc /tmp/bashrc  
cp: overwrite '/tmp/bashrc'? n <==n 不覆盖, y 为覆盖  
# 重复作两次动作, 由于 /tmp 底下已经存在 bashrc 了,  
加上 -i 选项后,  
# 则在覆盖前会询问使用者是否确定! 可以按下 n 或者 y  
来二次确认呢!
```

```
范例二: 变换目录到/tmp, 并将/var/log/wtmp 复制到/tmp  
且观察属性:  
[root@www ~]# cd /tmp  
[root@www tmp]# cp /var/log/wtmp . <==想要复制到当前  
目录, 最后的 . 不要忘  
[root@www tmp]# ls -l /var/log/wtmp wtmp  
-rw-rw-r-- 1 root utmp 96384 Sep 24 11:54 /var/log/wtmp  
-rw-r--r-- 1 root root 96384 Sep 24 14:06 wtmp  
# 注意上面的特殊字体, 在不加任何选项的情况下, 档案的  
某些属性/权限会改变;  
# 这是个很重要的特性! 要注意喔! 还有, 连档案建立的时  
间也不一样了!  
# 那如果你想要将档案的所有特性都一起复制过来该怎  
办? 可以加上 -a 喔! 如下所示:
```

```
[root@www tmp]# cp -a /var/log/wtmp wtmp_2  
[root@www tmp]# ls -l /var/log/wtmp wtmp_2  
-rw-rw-r-- 1 root utmp 96384 Sep 24 11:54 /var/log/wtmp  
-rw-rw-r-- 1 root utmp 96384 Sep 24 11:54 wtmp_2  
# 瞭解了吧! 整个资料特性完全一模一样! 真是不赖~这就  
是 -a 的特性!
```


这个 cp 的功能很多，由于我们常常会进行一些数据的复制，所以也会常常用到这个指令的。一般来说，我们如果去复制别人的数据（当然，该档案你必须要有 read 的权限才行啊！^_^）时，总是希望复制到的数据最后是我们自己的，所以，在预设的条件中，cp 的来源档与目的档的权限是不同的，目的档的拥有者通常会是指令操作者本身。举例来说，上面的范例二中，由于我是 root 的身份，因此复制过来的档案拥有者与群组就改变成为 root 所有了！这样说，可以明白吗？^_^

由于具有这个特性，因此当我们在进行备份的时候，某些需要特别注意的特殊权限档案，例如密码文件 (/etc/shadow) 以及一些配置文件，就不能直接以 cp 来复制，而必须要加上 -a 或者是 -p 等等可以完整复制档案权限的选项才行！另外，如果你想要复制档案给其他的使用者，也必须要注意到档案的权限(包含读、写、执行以及档案拥有者等等)，否则，其他人还是无法针对你给予的档案进行修订的动作喔！注意注意！

```
范例三：复制 /etc/ 这个目录下的所有内容到 /tmp 底下
[root@www tmp]# cp /etc/ /tmp
cp: omitting directory `/etc' <== 如果是目录则不能
直接复制，要加上 -r 的选项
[root@www tmp]# cp -r /etc/ /tmp
# 还是要再次的强调喔！ -r 是可以复制目录，但是，档案
与目录的权限可能会被改变
# 所以，也可以利用『 cp -a /etc /tmp 』来下达指令喔！
尤其是在备份的情况下！
```

```
范例四：将范例一复制的 bashrc 建立一个连结档
(symbolic link)
[root@www tmp]# ls -l bashrc
-rw-r--r-- 1 root root 176 Sep 24 14:02 bashrc <==
先观察一下档案情况
[root@www tmp]# cp -s bashrc bashrc_slink
[root@www tmp]# cp -l bashrc bashrc_hlink
[root@www tmp]# ls -l bashrc*
-rw-r--r-- 2 root root 176 Sep 24 14:02 bashrc <==
与源文件不太一样了！
-rw-r--r-- 2 root root 176 Sep 24 14:02 bashrc_hlink
lrwxrwxrwx 1 root root 6 Sep 24 14:20 bashrc_slink
-> bashrc
```

范例四可有趣了！使用 -l 及 -s 都会建立所谓的连结档(link file)，但是这两种连结档却有不一样的情况。这是怎么一回事啊？那个 -l 就是所谓的实体链接(hard link)，至于 -s 则是符号链接(symbolic link)，简单来说，bashrc_slink 是一个『快捷方式』，这个快捷方式会连结到 bashrc 去！所以你

会看到档名右侧会有个指向(->)的符号！

至于 `bashrc_hlink` 档案与 `bashrc` 的属性与权限完全一模一样，与尚未进行连结前的差异则是第二栏的 `link` 数由 1 变成 2 了！鸟哥这里先不介绍实体链接，因为实体链接涉及 `i-node` 的相关知识，我们下一章谈到文件系统(filesystem)时再来讨论这个问题。

```
范例五：若 ~/.bashrc 比 /tmp/bashrc 新才复制过来
[root@www tmp]# cp -u ~/.bashrc /tmp/bashrc
# 这个 -u 的特性，是在目标档案与来源档案有差异时，才会复制的。
# 所以，比较常被用于『备份』的工作当中喔！ ^_^

范例六：将范例四造成的 bashrc_slink 复制成为
bashrc_slink_1 与 bashrc_slink_2
[root@www tmp]# cp bashrc_slink bashrc_slink_1
[root@www tmp]# cp -d bashrc_slink bashrc_slink_2
[root@www tmp]# ls -l bashrc bashrc_slink*
-rw-r--r-- 2 root root 176 Sep 24 14:02 bashrc
lrwxrwxrwx 1 root root 6 Sep 24 14:20 bashrc_slink
-> bashrc
-rw-r--r-- 1 root root 176 Sep 24 14:32 bashrc_slink_1
<==与源文件相同
lrwxrwxrwx 1 root root 6 Sep 24 14:33 bashrc_slink_2
-> bashrc <==是连结档！
# 这个例子也是很有趣喔！原本复制的是连结档，但是却将
连结档的实际档案复制过来了
# 也就是说，如果没有加上任何选项时，cp 复制的是源文件，
而非链接文件的属性！
# 若要复制链接文件的属性，就得要使用 -d 的选项了！如
bashrc_slink_2 所示。

范例七：将家目录的 .bashrc 及 .bash_history 复制到
/tmp 底下
[root@www tmp]# cp ~/.bashrc ~/.bash_history /tmp
# 可以将多个数据一次复制到同一个目录去！最后面一定是
目录！
```

例题：

你能否使用 `vbird` 的身份，完整的复制 `/var/log/wtmp` 档案到 `/tmp` 底下，并更名为 `vbird_wtmp` 呢？

答：

实际做看看的结果如下：

```
[vbird@www ~]$ cp -a /var/log/wtmp /tmp/vbird_wtmp
[vbird@www ~]$ ls -l /var/log/wtmp /tmp/vbird_wtmp
-rw-rw-r-- 1 vbird vbird 96384  9月 24 11:54
/tmp/vbird_wtmp
-rw-rw-r-- 1 root  utmp  96384  9月 24 11:54
/var/log/wtmp
```

由于 vbird 的身份并不能随意修改档案的拥有者与群组，因此虽然能够复制 wtmp 的相关权限与时间等属性，但是与拥有者、群组相关的，原本 vbird 身份无法进行的动作，即使加上 -a 选项，也是无法达成完整复制权限的！

总之，由于 cp 有种种的文件属性与权限的特性，所以在复制时，你必须要清楚的了解到：

- 是否需要完整的保留来源档案的信息？
- 来源档案是否为连结档 (symbolic link file)？
- 来源档是否为特殊的档案，例如 FIFO, socket 等？
- 来源文件是否为目录？

-
- rm (移除档案或目录)

```
[root@www ~]# rm [-fir] 档案或目录
```

选项与参数：

-f : 就是 force 的意思，忽略不存在的档案，不会出现警告讯息；

-i : 互动模式，在删除前会询问使用者是否动作

-r : 递归删除啊！最常用在目录的删除了！这是非常危险的选项！！

范例一：将刚刚在 cp 的范例中建立的 bashrc 删除掉！

```
[root@www ~]# cd /tmp
```

```
[root@www tmp]# rm -i bashrc
```

```
rm: remove regular file `bashrc'? y
```

如果加上 -i 的选项就会主动询问喔，避免你删除到错误的档名！

范例二：透过通配符*的帮忙，将/tmp 底下开头为 bashrc 的档名通通删除：

```
[root@www tmp]# rm -i bashrc*
```

注意那个星号，代表的是 0 到无穷多个任意字符喔！很好用的东西！

范例三：将 cp 范例中所建立的 /tmp/etc/ 这个目录删除掉！

```
[root@www tmp]# rmdir /tmp/etc
rmdir: etc: Directory not empty <== 删不掉啊！因为这不是空的目录！
[root@www tmp]# rm -r /tmp/etc
rm: descend into directory `/tmp/etc'? y
.... (中间省略)....
# 因为身份是 root，预设已经加入了 -i 的选项，所以你要一直按 y 才会删除！
# 如果不想要继续按 y，可以按下『[ctrl]-c』来结束 rm 的工作。
# 这是一种保护的動作，如果确定要删除掉此目录而不要询问，可以这样做：
[root@www tmp]# \rm -r /tmp/etc
# 在指令前加上反斜杠，可以忽略掉 alias 的指定选项喔！至于 alias 我们在 bash 再谈！
```

范例四：删除一个带有 - 开头的档案

```
[root@www tmp]# touch ./-aaa- <==touch 这个指令可以建立空档案！
[root@www tmp]# ls -l
-rw-r--r-- 1 root root      0 Sep 24 15:03 -aaa- <==
档案大小为 0，所以是空档案
[root@www tmp]# rm -aaa-
Try `rm --help' for more information. <== 因为 "-" 是选项嘛！所以系统误判了！
[root@www tmp]# rm ./-aaa-
```

这是移除的指令(remove)，要注意的是，通常在 Linux 系统下，为了怕档案被误杀，所以很多 distributions 都已经默认加入 -i 这个选项了！而如果要连目录下的东西都一起杀掉的话，例如子目录里面还有子目录时，那就要使用 -r 这个选项了！不过，使用『rm -r』这个指令之前，请千万注意了，因为该目录或档案『肯定』会被 root 杀掉！因为系统不会再次询问你是否要砍掉呦！所以那是个超级严重的指令下达呦！得特别注意！不过，如果你确定该目录不要了，那么使用 rm -r 来循环杀掉是不错的方式！

另外，范例四也是很有趣的例子，我们在之前就谈过，档名最好不要使用 "-" 号开头，因为 "-" 后面接的是选项，因此，单纯的使用『rm -aaa-』系统的指令就会误判啦！那如果使用后面会谈到的正规表示法时，还是会出问题的！所以，只能用避开首位字符是 "-" 的方法啦！就是加上本目录『./』即可！如

果 `man rm` 的话，其实还有一种方法，那就是『 `rm -- -aaa-` 』也可以啊！

- `mv`（移动档案与目录，或更名）

```
[root@www ~]# mv [-fiu] source destination
[root@www ~]# mv [options] source1 source2 source3 ....
directory
选项与参数：
-f  : force 强制的意思，如果目标档案已经存在，不会询问而直接覆盖；
-i  : 若目标档案 (destination) 已经存在时，就会询问是否覆盖！
-u  : 若目标档案已经存在，且 source 比较新，才会更新 (update)
```

范例一：复制一档案，建立一目录，将档案移动到目录中

```
[root@www ~]# cd /tmp
[root@www tmp]# cp ~/.bashrc bashrc
[root@www tmp]# mkdir mvtest
[root@www tmp]# mv bashrc mvtest
# 将某个档案移动到某个目录去，就是这样做！
```

范例二：将刚刚的目录名称更名为 `mvtest2`

```
[root@www tmp]# mv mvtest mvtest2 <== 这样就更名了！
简单～
# 其实在 Linux 底下还有个有趣的指令，名称为 rename，
# 该指令专职进行多个档名的同时更名，并非针对单一档名变更，与 mv 不同。请 man rename。

范例三：再建立两个档案，再全部移动到 /tmp/mvtest2 当中



```
[root@www tmp]# cp ~/.bashrc bashrc1
[root@www tmp]# cp ~/.bashrc bashrc2
[root@www tmp]# mv bashrc1 bashrc2 mvtest2
注意到这边，如果有多个来源档案或目录，则最后一个目标文件一定是『目录！』
意思是说，将所有数据移动到该目录的意思！
```


```

这是搬移 (move) 的意思！当你要移动档案或目录的时后，呵呵！这个指令就很重要啦！同样的，你也可以使用 `-u` (update) 来测试新旧档案，看看是否需要搬移啰！另外一个用途就是『变更档名！』，我们可以很轻易的使用 `mv` 来变更一个档案的档名呢！不过，在 Linux 才有的指令当中，有个 `rename`，可

以用来更改大量档案的档名，你可以利用 `man rename` 来查阅一下，也是挺有趣的指令喔！

取得路径的文件名与目录名称

我们前面介绍的完整文件名（包含目录名称与文件名）当中提到，完整档名最长可以到达 4096 个字符。那么你怎么知道那个是档名？那个是目录名？嘿嘿！就是利用斜线（/）来分辨啊！其实，取得文件名或者是目录名称，一般的用途应该是在写程序的时候，用来判断之用的啦～所以，这部分的指令可以用在第三篇内的 shell scripts 里头喔！底下我们简单的以几个范例来谈一谈 `basename` 与 `dirname` 的用途！

```
[root@www ~]# basename /etc/sysconfig/network
network      <== 很简单！就取得最后的档名～
[root@www ~]# dirname /etc/sysconfig/network
/etc/sysconfig <== 取得的变成目录名了！
```

档案内容查阅：

如果我们要查阅一个档案的内容时，该如何是好呢？这里有相当多有趣的指令可以来分享一下：最常使用的显示档案内容的指令可以说是 `cat` 与 `more` 及 `less` 了！此外，如果我们要查看一个很大型的档案（好几百 MB 时），但是我们只需要后端的几行字而已，那么该如何是好？呵呵！用 `tail` 呀，此外，`tac` 这个指令也可以达到！好了，说说各个指令的用途吧！

- `cat` 由第一行开始显示档案内容
- `tac` 从最后一行开始显示，可以看出 `tac` 是 `cat` 的倒着写！
- `nl` 显示的时候，顺道输出行号！
- `more` 一页一页的显示档案内容
- `less` 与 `more` 类似，但是比 `more` 更好的是，他可以往前翻页！
- `head` 只看头几行
- `tail` 只看尾巴几行
- `od` 以二进制的方式读取档案内容！

🐼 直接检视档案内容

直接查阅一个档案的内容可以使用 `cat/tac/nl` 这几个指令啊！

- `cat` (concatenate)

```
[root@www ~]# cat [-AbEnTv]
```

选项与参数：

-A : 相当于 -vET 的整合选项，可列出一些特殊字符而不是空白而已；

-b : 列出行号，仅针对非空白行做行号显示，空白行不标行号！

-E : 将结尾的断行字符 \$ 显示出来；

-n : 打印出行号，连同空白行也会有行号，与 -b 的选项不同；

-T : 将 [tab] 按键以 ^I 显示出来；

-v : 列出一些看不出来的特殊字符

范例一：检阅 `/etc/issue` 这个档案的内容

```
[root@www ~]# cat /etc/issue
```

```
CentOS release 5.2 (Final)
```

```
Kernel \r on an \m
```

范例二：承上题，如果还要加印行号呢？

```
[root@www ~]# cat -n /etc/issue
```

```
1  Fedora Core release 4 (Stentz)
```

```
2  Kernel \r on an \m
```

```
3
```

看到了吧！可以印出行号呢！这对于大档案要找某个特定的行时，有点用处！

如果不想要编排空白行的行号，可以使用『`cat -b /etc/issue`』，自己测试看看：

范例三：将 `/etc/xinetd.conf` 的内容完整的显示出来(包含特殊字符)

```
[root@www ~]# cat -A /etc/xinetd.conf
```

```
#$
```

```
.... (中间省略)....
```

```
$
```

```
defaults$
```



```
{
$
# The next two items are intended to be a quick access
place to$
.... (中间省略)....
^Ilog_type^I= SYSLOG daemon info $
^Ilog_on_failure^I= HOST$
^Ilog_on_success^I= PID HOST DURATION EXIT$
.... (中间省略)....
includedir /etc/xinetd.d$
$
# 上面的结果限于篇幅，鸟哥删除掉很多数据了。另外，输出
的结果并不会有特殊字体，
# 鸟哥上面的特殊字体是要让您发现差异点在哪里就是了。
基本上，在一般的环境中，
# 使用 [tab] 与空格键的效果差不多，都是一堆空白啊！
我们无法知道两者的差别。
# 此时使用 cat -A 就能够发现那些空白的地方是啥鬼东西
了！[tab]会以 ^I 表示，
# 断行字符则是以 $ 表示，所以你可以发现每一行后面都
是 $ 啊！不过断行字符
# 在 Windows/Linux 则不太相同，Windows 的断行字符是
^M$ 啰。
# 这部分我们会在第十章 vim 软件的介绍时，再次的说明
到喔！
```

嘿嘿！Linux 里面有『猫』指令？喔！不是的，cat 是 Concatenate（连续）的简写，主要的功能是将一个档案的内容连续的印出在屏幕上面！例如上面的例子中，我们将 /etc/issue 印出来！如果加上 -n 或 -b 的话，则每一行前面还会加上行号呦！

鸟哥个人是比较少用 cat 啦！毕竟当你的档案内容的行数超过 40 行以上，嘿嘿！根本来不及在屏幕上看到结果！所以，配合等一下要介绍的 more 或者是 less 来执行比较好！此外，如果是一般的 DOS 档案时，就需要特别留意一些奇奇怪怪的符号了，例如断行与 [tab] 等，要显示出来，就得加入 -A 之类的选项了！

- tac（反向列示）

```
[root@www ~]# tac /etc/issue

Kernel \r on an \m
```

```
CentOS release 5.2 (Final)
# 嘿嘿！与刚刚上面的范例一比较，是由最后一行先显示喔！
```

tac 这个好玩了！怎么说呢？详细的看一下，cat 与 tac，有没有发现呀！对啦！tac 刚好是将 cat 反写过来，所以他的功能就跟 cat 相反啦，cat 是由『第一行到最后一行连续显示在屏幕上』，而 tac 则是『由最后一行到第一行反向在屏幕上显示出来』，很好玩吧！

- nl (添加行号打印)

```
[root@www ~]# nl [-bnw] 档案
选项与参数：
-b  : 指定行号指定的方式，主要有两种：
      -b a : 表示不论是否为空行，也同样列出行号(类似 cat -n)；
      -b t : 如果有空行，空的那一行不要列出行号(默认值)；
-n  : 列出行号表示的方法，主要有三种：
      -n ln : 行号在屏幕的最左方显示；
      -n rn : 行号在自己字段的最右方显示，且不加 0；
      -n rz : 行号在自己字段的最右方显示，且加 0；
-w  : 行号字段的占用的位数。

范例一：用 nl 列出 /etc/issue 的内容
[root@www ~]# nl /etc/issue
    1 CentOS release 5.2 (Final)
    2 Kernel \r on an \m

# 注意看，这个档案其实有三行，第三行为空白(没有任何字符)，
# 因为他是空白行，所以 nl 不会加上行号喔！如果确定要加上行号，可以这样做：

[root@www ~]# nl -b a /etc/issue
    1 CentOS release 5.2 (Final)
    2 Kernel \r on an \m
    3

# 呵呵！行号加上来啰～那么如果能让行号前面自动补上 0 呢？可这样

[root@www ~]# nl -b a -n rz /etc/issue
```

```

000001  CentOS release 5.2 (Final)
000002  Kernel \r on an \m
000003
# 嘿嘿！自动在自己字段的地方补上 0 了～预设字段是六
位数，如果想要改成 3 位数？

[root@www ~]# nl -b a -n rz -w 3 /etc/issue
001      CentOS release 5.2 (Final)
002      Kernel \r on an \m
003
# 变成仅有 3 位数啰～

```

nl 可以将输出的档案内容自动的加上行号！其预设的结果与 cat -n 有点不太一样，nl 可以将行号做比较多的显示设计，包括位数与是否自动补齐 0 等的功能呢。

💡可翻页检视

前面提到的 nl 与 cat, tac 等等，都是一次性的将数据一口气显示到屏幕上，那有没有可以进行一页一页翻动的指令啊？让我们可以一页一页的观察，才不会前面的数据看不到啊～呵呵！有的！那就是 more 与 less 啰～

- more (一页一页翻动)

```

[root@www ~]# more /etc/man.config
#
# Generated automatically from man.conf.in by the
# configure script.
#
# man.conf from man-1.6d
.... (中间省略)....
--More--(28%) <== 重点在这一行喔！你的光标也会在这
里等待你的指令

```

仔细的给他看到上面的范例，如果 more 后面接的档案内容行数大于屏幕输出的行数时，就会出现类似上面的图示。重点在最后一行，最后一行会显示出目前显示的百分比，而且还可以在最后一行输入一些有用的指令喔！在 more 这个程序的运作过程中，你有几个按键可以按的：

- 空格键 (space): 代表向下翻一页;
- Enter : 代表向下翻『一行』;
- /字符串 : 代表在这个显示的内容当中, 向下搜寻『字符串』这个关键词;
- :f : 立刻显示出文件名以及目前显示的行数;
- q : 代表立刻离开 more , 不再显示该档案内容。
- b 或 [ctrl]-b : 代表往回翻页, 不过这动作只对档案有用, 对管线无用。

要离开 more 这个指令的显示工作, 可以按下 q 就能够离开了。而要向下翻页, 就使用空格键即可。 比较有用的是搜寻字符串的功能, 举例来说, 我们使用『 more /etc/man.config 』来观察该档案, 若想要在该档案内搜寻 MANPATH 这个字符串时, 可以这样做:

```
[root@www ~]# more /etc/man.config
#
# Generated automatically from man.conf.in by the
# configure script.
#
# man.conf from man-1.6d
.... (中间省略)....
/MANPATH <== 输入了 / 之后, 光标就会自动跑到最底下一行等待输入!
```

如同上面的说明, 输入了 / 之后, 光标就会跑到最底下一行, 并且等待你的输入, 你输入了字符串并按下[enter]之后, 嘿嘿! more 就会开始向下搜寻该字符串啰~而重复搜寻同一个字符串, 可以直接按下 n 即可啊! 最后, 不想要看了, 就按下 q 即可离开 more 啦!

-
- less (一页一页翻动)

```
[root@www ~]# less /etc/man.config
#
# Generated automatically from man.conf.in by the
# configure script.
#
```

```
# man.conf from man-1.6d
.... (中间省略)....
: <== 这里可以等待你输入指令！
```

less 的用法比起 more 又更加的有弹性，怎么说呢？在 more 的时候，我们并没有办法向前面翻，只能往后面看，但若使用了 less 时，呵呵！就可以使用 [pageup] [pagedown] 等按键的功能来往前往后翻看文件，你瞧，是不是更容易使用来观看一个档案的内容了呢！

除此之外，在 less 里头可以拥有更多的『搜寻』功能喔！不止可以向下搜寻，也可以向上搜寻～ 实在是很不错用～基本上，可以输入的指令有：

- 空格键 : 向下翻动一页；
- [pagedown]: 向下翻动一页；
- [pageup] : 向上翻动一页；
- /字符串 : 向下搜寻『字符串』的功能；
- ?字符串 : 向上搜寻『字符串』的功能；
- n : 重复前一个搜寻（与 / 或 ? 有关！）
- N : 反向的重复前一个搜寻（与 / 或 ? 有关！）
- q : 离开 less 这个程序；

查阅档案内容还可以进行搜寻的动作～瞧～ less 是否很不错用啊！其实 less 还有很多的功能喔！详细的使用方式请使用 `man less` 查询一下啊！ ^_^

你是否会觉得 less 使用的画面与环境与 [man page](#) 非常的类似呢？没错啦！因为 man 这个指令就是呼叫 less 来显示说明文件的内容的！现在你是否觉得 less 很重要呢？ ^_^

资料撷取

我们可以将输出的资料作一个最简单的撷取，那就是取出前面 (head) 与取出后面 (tail) 文字的功能。不过，要注意的是，head 与 tail 都是以『行』为单位来进行数据撷取的喔！

-
- head (取出前面几行)

```
[root@www ~]# head [-n number] 档案
选项与参数:
-n : 后面接数字, 代表显示几行的意思

[root@www ~]# head /etc/man.config
# 默认的情况中, 显示前面十行! 若要显示前 20 行, 就得要这样:
[root@www ~]# head -n 20 /etc/man.config

范例: 如果后面 100 行的数据都不打印, 只打印
/etc/man.config 的前面几行, 该如何是好?
[root@www ~]# head -n -100 /etc/man.config
```

head 的英文意思就是『头』啦, 那么这个东西的用法自然就是显示出一个档案的前几行啰! 没错! 就是这样! 若没有加上 -n 这个选项时, 默认只显示十行, 若只要一行呢? 那就加入『head -n 1 filename』即可!

另外那个 -n 选项后面的参数较有趣, 如果接的是负数, 例如上面范例的 -n -100 时, 代表列前的所有行数, 但不包括后面 100 行。举例来说, /etc/man.config 共有 141 行, 则上述的指令『head -n -100 /etc/man.config』就会列出前面 41 行, 后面 100 行不会打印出来了。这样说, 比较容易懂了吧? ^_^

- tail (取出后面几行)

```
[root@www ~]# tail [-n number] 档案
选项与参数:
-n : 后面接数字, 代表显示几行的意思
-f : 表示持续侦测后面所接的档名, 要等到按下[ctrl]-c
才会结束 tail 的侦测

[root@www ~]# tail /etc/man.config
# 默认的情况中, 显示最后的十行! 若要显示最后的 20 行,
就得要这样:
[root@www ~]# tail -n 20 /etc/man.config

范例一: 如果不知道/etc/man.config 有几行, 却只想列出
100 行以后的数据时?
```

```
[root@www ~]# tail -n +100 /etc/man.config
```

范例二：持续侦测/var/log/messages 的内容

```
[root@www ~]# tail -f /var/log/messages
```

<==要等到输入[ctrl]-c 之后才会离开 tail 这个指令的侦测！

有 head 自然就有 tail（尾巴）啰！没错！这个 tail 的用法跟 head 的用法差不多类似，只是显示的是后面几行就是了！默认也是显示十行，若要显示非十行，就加 -n number 的选项即可。

范例一的内容就有趣啦！其实与 head -n -xx 有异曲同工之妙。当下达『tail -n +100 /etc/man.config』代表该档案从 100 行以后都会被列出来，同样的，在 man.config 共有 141 行，因此第 100~141 行就会被列出来啦！前面的 99 行都不会被显示出来喔！

至于范例二中，由于/var/log/messages 随时会有数据写入，你想要让该档案有数据写入时就立刻显示到屏幕上，就利用 -f 这个选项，他可以一直侦测 /var/log/messages 这个档案，新加入的数据都会被显示到屏幕上。直到你按下[ctrl]-c 才会离开 tail 的侦测喔！

例题：

假如我想要显示 /etc/man.config 的第 11 到第 20 行呢？

答：

这个应该不算难，想一想，在第 11 到第 20 行，那么我取前 20 行，再取后十行，所以结果就是：『head -n 20 /etc/man.config | tail -n 10』，这样就可以得到第 11 到第 20 行之间的内容了！但是里面涉及到管线命令，需要在第三篇的时候才讲的到！

💡非纯文本档： od

我们上面提到的，都是在查阅纯文本档的内容。那么万一我们想要查阅非文本文件，举例来说，例如 /usr/bin/passwd 这个执行档的内容时，又该如何去读出信息呢？事实上，由于执行档通常是 binary file，使用上头提到的指令来读取他的内容时，确实会产生类似乱码的数据啊！那怎么办？没关系，我们可以利用 od 这个指令来读取喔！

```
[root@www ~]# od [-t TYPE] 档案
```

选项或参数：

-t ：后面可以接各种『类型 (TYPE)』的输出，例如：

a : 利用默认的字符来输出;
c : 使用 ASCII 字符来输出
d[size] : 利用十进制(decimal)来输出数据, 每个整数占用 size bytes ;
f[size] : 利用浮点数(floating)来输出数据, 每个数占用 size bytes ;
o[size] : 利用八进制(octal)来输出数据, 每个整数占用 size bytes ;
x[size] : 利用十六进制(hexadecimal)来输出数据, 每个整数占用 size bytes ;

范例一: 请将/usr/bin/passwd 的内容使用 ASCII 方式来展现!

```
[root@www ~]# od -t c /usr/bin/passwd
0000000 177 E L F 001 001 001 \0 \0 \0 \0
\0 \0 \0 \0 \0
0000020 002 \0 003 \0 001 \0 \0 \0 260 225 004
\b 4 \0 \0 \0
0000040 020 E \0 \0 \0 \0 \0 \0 4 \0
\0 \a \0 ( \0
0000060 035 \0 034 \0 006 \0 \0 \0 4 \0 \0
\0 4 200 004 \b
0000100 4 200 004 \b 340 \0 \0 \0 340 \0 \0
\0 005 \0 \0 \0
.....(后面省略)....
```

最左边第一栏是以 8 进位来表示 bytes 数。以上面范例来说, 第二栏 0000020 代表开头是
第 16 个 bytes (2x8) 的内容之意。

范例二: 请将/etc/issue 这个档案的内容以 8 进位列出储存值与 ASCII 的对照表

```
[root@www ~]# od -t oCc /etc/issue
0000000 103 145 156 164 117 123 040 162 145 154 145 141
163 145 040 065
C e n t O S r e l e
a s e 5
0000020 056 062 040 050 106 151 156 141 154 051 012 113
145 162 156 145
. 2 ( F i n a l ) \n
K e r n e
0000040 154 040 134 162 040 157 156 040 141 156 040 134
155 012 012
l \ r o n a n
```

```
\ m \n \n
0000057
# 如上所示，可以发现每个字符可以对应到的数值为何！
# 例如 e 对应的记录数值为 145，转成十进制：
1x8^2+4x8+5=101。
```

利用这个指令，可以将 data file 或者是 binary file 的内容数据给他读出来喔！虽然读出的来数值预设是使用非文本文件，亦即是 16 进位的数值来显示的，不过，我们还是可以透过 `-t c` 的选项与参数来将数据内的字符以 ASCII 类型的字符来显示，虽然对于一般使用者来说，这个指令的用处可能不大，但是对于工程师来说，这个指令可以将 binary file 的内容作一个大致的输出，他们可以看得出东西的啦～ ^_^

如果对纯文本文件使用这个指令，你甚至可以发现到 ASCII 与字符的对照表！非常有趣！例如上述的范例二，你可以发现到每个英文字 e 对照到的数字都是 145，转成十进制你就能够发现那是 101 啰！如果你有任何程序语言的书，拿出来对照一下 ASCII 的对照表，就能够发现真是正确啊！呵呵！

💡修改档案时间或建置新档： `touch`

我们在 [ls 这个指令的介绍](#)时，有稍微提到每个档案在 linux 底下都会记录许多的时间参数，其实是有三个主要的变动时间，那么三个时间的意义是什么呢？

- **modification time (mtime):**
当该档案的『内容数据』变更时，就会更新这个时间！内容数据指的是档案的内容，而不是档案的属性或权限喔！
- **status time (ctime):**
当该档案的『状态 (status)』改变时，就会更新这个时间，举例来说，像是权限与属性被更改了，都会更新这个时间啊。
- **access time (atime):**
当『该档案的内容被取用』时，就会更新这个读取时间 (access)。举例来说，我们使用 `cat` 去读取 `/etc/man.config`，就会更新该档案的 `atime` 了。

这是个挺有趣的现象，举例来说，我们来看一看你自己的 `/etc/man.config` 这个档案的时间吧！

```
[root@www ~]# ls -l /etc/man.config
-rw-r--r-- 1 root root 4617 Jan  6 2007
/etc/man.config
```

```
[root@www ~]# ls -l --time=atime /etc/man.config
-rw-r--r-- 1 root root 4617 Sep 25 17:54
/etc/man.config
[root@www ~]# ls -l --time=ctime /etc/man.config
-rw-r--r-- 1 root root 4617 Sep  4 18:03
/etc/man.config
```

看到了吗？在默认的情况下，ls 显示出来的是该档案的 mtime，也就是这个档案的内容上次被更动的时间。至于鸟哥的系统是在 9 月 4 号的时候安装的，因此，这个档案被产生导致状态被更动的时间就回溯到那个时间点了(ctime)！而还记得刚刚我们使用的范例当中，有使用到 man.config 这个档案啊，所以啊，他的 atime 就会变成刚刚使用的时间了！

档案的时间是很重要的，因为，如果档案的时间误判的话，可能会造成某些程序无法顺利的运作。OK！那么万一我发现了一个档案来自未来，该如何让该档案的时间变成『现在』的时刻呢？很简单啊！就用『touch』这个指令即可！

Tips:

嘿嘿！不要怀疑系统时间会『来自未来』喔！很多时候会有这个问题的！举例来说在安装过后系统时间可能会被改变！因为台湾时区在国际标准时间『格林威治时间，GMT』的右边，所以会比较早看到阳光，也就是说，台湾时间比 GMT 时间快了八小时！如果安装行为不当，我们的系统可能会有八小时快转，你的档案就有可能来自八小时候了。



至于某些情况下，由于 BIOS 的设定错误，导致系统时间跑到未来时间，并且你又建立了某些档案。等你将时间改回正确的时间时，该档案不就变成来自未来了？^_^

```
[root@www ~]# touch [-acdmt] 档案
选项与参数：
-a  : 仅修订 access time；
-c  : 仅修改档案的时间，若该档案不存在则不建立新档案；
-d  : 后面可以接欲修订的日期而不用目前的日期，也可以使用 --date="日期或时间"
-m  : 仅修改 mtime ；
-t  : 后面可以接欲修订的时间而不用目前的时间，格式为 [YYMMDDhhmm]

范例一：新建一个空的档案并观察时间
[root@www ~]# cd /tmp
[root@www tmp]# touch testtouch
[root@www tmp]# ls -l testtouch
-rw-r--r-- 1 root root 0 Sep 25 21:09 testtouch
```

```
# 注意到，这个档案的大小是 0 呢！在预设的状态下，如果 touch 后面有接档案，  
# 则该档案的三个时间 (atime/ctime/mtime) 都会更新为目前的时间。若该档案不存在，  
# 则会主动的建立一个新的空的档案喔！例如上面这个例子！
```

范例二：将 ~/.bashrc 复制成为 bashrc，假设复制完全的属性，检查其日期

```
[root@www tmp]# cp -a ~/.bashrc bashrc  
[root@www tmp]# ll bashrc; ll --time=atime bashrc; ll --time=ctime bashrc  
-rw-r--r-- 1 root root 176 Jan  6  2007 bashrc <==  
这是 mtime  
-rw-r--r-- 1 root root 176 Sep 25 21:11 bashrc <==  
这是 atime  
-rw-r--r-- 1 root root 176 Sep 25 21:12 bashrc <==  
这是 ctime
```

在上面这个案例当中我们使用了『ll』这个指令(两个英文L的小写)，这个指令其实就是『ls -l』的意思，ll 本身不存在，是被『做出来』的一个命令别名。相关的[命令别名我们会在 bash 章节](#)当中详谈的，这里先知道 ll="ls -l"即可。至于分号『；』则代表连续指令的下达啦！你可以在一行指令当中写入多重指令，这些指令可以『依序』执行。由上面的指令我们会知道 ll 那一行有三个指令被下达在同一行中。

至于执行的结果当中，我们可以发现数据的内容与属性是被复制过来的，因此档案内容时间(mtime)与原本档案相同。但是由于这个档案是刚刚被建立的，因此状态(ctime)与读取时间就便呈现在的时间啦！那如果你想要变更这个档案的时间呢？可以这样做：

```
范例三：修改案例二的 bashrc 档案，将日期调整为两天前  
[root@www tmp]# touch -d "2 days ago" bashrc  
[root@www tmp]# ll bashrc; ll --time=atime bashrc; ll --time=ctime bashrc  
-rw-r--r-- 1 root root 176 Sep 23 21:23 bashrc  
-rw-r--r-- 1 root root 176 Sep 23 21:23 bashrc  
-rw-r--r-- 1 root root 176 Sep 25 21:23 bashrc  
# 跟上个范例比较看看，本来是 25 日的变成了 23 日了 (atime/mtime)~  
# 不过， ctime 并没有跟着改变喔！
```

范例四：将上个范例的 bashrc 日期改为 2007/09/15 2:02

```
[root@www tmp]# touch -t 0709150202 bashrc
[root@www tmp]# ll bashrc; ll --time=atime bashrc; ll
--time=ctime bashrc
-rw-r--r-- 1 root root 176 Sep 15 2007 bashrc
-rw-r--r-- 1 root root 176 Sep 15 2007 bashrc
-rw-r--r-- 1 root root 176 Sep 25 21:25 bashrc
# 注意看看，日期在 atime 与 mtime 都改变了，但是
ctime 则是记录目前的时间！
```

透过 touch 这个指令，我们可以轻易的修订档案的日期与时间。并且也可以建立一个空的档案喔！不过，要注意的是，即使我们复制一个档案时，复制所有的属性，但也没有办法复制 ctime 这个属性的。ctime 可以记录这个档案最近的状态 (status) 被改变的时间。无论如何，还是要告知大家，我们平时看的文件属性中，比较重要的还是属于那个 mtime 啊！我们关心的常常是这个档案的『内容』是什么时候被更动的说～瞭乎？

无论如何，touch 这个指令最常被使用的情况是：

- 建立一个空的档案；
- 将某个档案日期修订为目前 (mtime 与 atime)



档案与目录的默认权限与隐藏权限

由第六章、Linux 档案权限的内容我们可以知道一个档案有若干个属性，包括读写执行 (r, w, x) 等基本权限，及是否为目录 (d) 与档案 (-) 或者是连结档 (l) 等等的属性！要修改属性的方法在前面也约略提过了 (chgrp, chown, chmod)，本小节会再加强补充一下！

除了基本 r, w, x 权限外，在 Linux 的 Ext2/Ext3 文件系统下，我们还可以设定其他的系统隐藏属性，这部份可使用 chattr 来设定，而以 lsattr 来查看，最重要的属性就是可以设定其不可修改的特性！让连档案的拥有者都不能进行修改！这个属性可是相当重要的，尤其是在安全机制上面 (security)！

首先，先来复习一下上一章谈到的权限概念，将底下的例题看一看先：

例题：

你的系统有个一般身份用户 dmtsai，他的群组属于 users，他的家目录在 /home/dmtsai，你是 root，你想将你的 ~/.bashrc 复制给他，可以怎么作？

答：

由上一章的权限概念我们可以知道 root 虽然可以将这个档案复制给 dmtsai，不过这个档案在 dmtsai 的家目录中却可能让 dmtsai 没有

办法读写(因为该档案属于 root 的嘛!而 dmtsai 又不能使用 chown 之故)。此外,我们又担心覆盖掉 dmtsai 自己的 .bashrc 配置文件,因此,我们可以进行如下的动作喔:

复制档案: `cp ~/.bashrc ~dmtsai/bashrc`

修改属性: `chown dmtsai:users ~dmtsai/bashrc`

例题:

我想在 /tmp 底下建立一个目录,这个目录名称为 chapter7_1,并且这个目录拥有者为 dmtsai,群组为 users,此外,任何人都可以进入该目录浏览档案,不过除了 dmtsai 之外,其他人都不能修改该目录下的档案。

答:

因为除了 dmtsai 之外,其他人不能修改该目录下的档案,所以整个目录的权限应该是 drwxr-xr-x 才对!因此你应该这样做:

建立目录: `mkdir /tmp/chapter7_1`

修改属性: `chown -R dmtsai:users /tmp/chapter7_1`

修改权限: `chmod -R 755 /tmp/chapter7_1`

在上面这个例题当中,如果你知道 755 那个分数是怎么计算出来的,那么你应该对于权限有一定程度的概念了。如果你不知道 755 怎么来的?那么...赶快回去前一章看看 `chmod` 那个指令的介绍部分啊!这部分很重要喔!你得要先清楚的了解到才行~否则就进行不下去啰~假设你对于权限都认识的差不多了,那么底下我们就要来谈一谈,『新增一个档案或目录时,默认的权限是什么?』这个议题!

 档案预设权限: `umask`

OK! 那么现在我们知道如何建立或者是改变一个目录或档案的属性了,不过,你知道当你建立一个新的档案或目录时,他的默认权限会是什么吗?呵呵!那就与 `umask` 这个玩意儿有关了!那么 `umask` 是在搞什么呢?基本上, `umask` 就是指 定『目前用户在建立档案或目录时候的权限默认值』,那么如何得知或设定 `umask` 呢?他的指定条件以底下的方式来指定:

```
[root@www ~]# umask
```

```
0022
```

<==与一般权限有关的是后面三个数字!

```
[root@www ~]# umask -S
u=rwx,g=rx,o=rx
```

查阅的方式有两种，一种可以直接输入 `umask`，就可以看到数字型态的权限设定分数，一种则是加入 `-S` (Symbolic) 这个选项，就会以符号类型的方式来显示出权限了！奇怪的是，怎么 `umask` 会有四组数字啊？不是只有三组吗？是没错啦。第一组是特殊权限用的，我们先不要理他，所以先看后面三组即可。

在默认权限的属性上，目录与档案是不一样的。从第六章我们知道 `x` 权限对于目录是非常重要的！但是一般档案的建立则不应该有执行的权限，因为一般档案通常是用在于数据的记录嘛！当然不需要执行的权限了。因此，预设的情况如下：

- 若使用者建立为『档案』则预设『没有可执行(`x`)权限』，亦即只有 `rw` 这两个项目，也就是最大为 666 分，预设权限如下：

```
-rw-rw-rw-
```

- 若用户建立为『目录』，则由于 `x` 与是否可以进入此目录有关，因此默认为所有权限均开放，亦即为 777 分，预设权限如下：

```
drwxrwxrwx
```

要注意的是，`umask` 的分数指的是『该默认值需要减掉的权限！』因为 `r`、`w`、`x` 分别是 4、2、1 分，所以啰！也就是说，当要拿掉能写的权限，就是输入 2 分，而如果要拿掉能读的权限，也就是 4 分，那么要拿掉读与写的权限，也就是 6 分，而要拿掉执行与写入的权限，也就是 3 分，这样了解吗？请问你，5 分是什么？呵呵！就是读与执行的权限啦！

如果以上面的例子来说明的话，因为 `umask` 为 022，所以 `user` 并没有被拿掉任何权限，不过 `group` 与 `others` 的权限被拿掉了 2（也就是 `w` 这个权限），那么当使用者：

- 建立档案时：(`-rw-rw-rw-`) - (`-----w--w-`) ==> `-rw-r--r--`
- 建立目录时：(`drwxrwxrwx`) - (`d-----w--w-`) ==> `drwxr-xr-x`

不相信吗？我们就来测试看看吧！


```
[root@www ~]# umask
0022
[root@www ~]# touch test1
[root@www ~]# mkdir test2
[root@www ~]# ll
-rw-r--r-- 1 root root    0 Sep 27 00:25 test1
drwxr-xr-x 2 root root 4096 Sep 27 00:25 test2
```

呵呵！瞧见了吧！确定新建档案的权限是没有错的。

- umask 的利用与重要性：专题制作

想象一个状况，如果你跟你的同学在同一部主机里面工作时，因为你们两个正在进行同一个专题，老师也帮你们两个的账号建立好了相同群组的状态，并且将 /home/class/ 目录做为你们两个人的专题目录。想象一下，有没有可能你所制作的档案你的同学无法编辑？果真如此的话，那就伤脑筋了！

这个问题很常发生啊！举上面的案例来看就好了，你看一下 test1 的权限是几分？644 呢！意思是『如果 umask 订定为 022，那新建的数据只有用户自己具有 w 的权限，同群组的人只有 r 这个可读的权限而已，并无法修改喔！』这样要怎么共同制作专题啊！您说是吧！

所以，当我们需要新建档案给同群组的使用者共同编辑时，那么 umask 的群组就不能拿掉 2 这个 w 的权限！所以啰，umask 就得要是 002 之类的才可以！这样新建的档案才能够是 -rw-rw-r-- 的权限模样喔！那么如何设定 umask 呢？简单的很，直接在 umask 后面输入 002 就好了！

```
[root@www ~]# umask 002
[root@www ~]# touch test3
[root@www ~]# mkdir test4
[root@www ~]# ll
-rw-rw-r-- 1 root root    0 Sep 27 00:36 test3
drwxrwxr-x 2 root root 4096 Sep 27 00:36 test4
```

所以说，这个 umask 对于新建档案与目录的默认权限是很有关系的！这个概念可以用在任何服务器上面，尤其是未来在你架设文件服务器 (file server)，举例来说，[SAMBA Server](#) 或者是 [FTP server](#) 时，都是很重要的观念！这牵涉到你的使用者是否能够将档案进一步利用的问题喔！不要等闲视之！

例题：

假设你的 umask 为 003，请问该 umask 情况下，建立的档案与目录权限为？

答：

umask 为 003 ，所以拿掉的权限为 -----wx，因此：

档案： (-rw-rw-rw-) - (-----wx) = -rw-rw-r--

目录： (drwxrwxrwx) - (-----wx) = drwxrwxr--

Tips:

关于 umask 与权限的计算方式中，教科书喜欢使用二进制的方式来进行 AND 与 NOT 的计算，不过，鸟哥还是比较喜欢使用符号方式来计算～联想上面比较容易一点～

但是，有的书籍或者是 BBS 上面的朋友，喜欢使用档案默认属性 666 与目录默认属性 777 来与 umask 进行相减的计算～这是不好的喔！以上面例题来看，如果使用默认属性相加减，则档案变成：666-003=663，亦即是 -rw-rw--wx，这可是完全不对的喔！想想看，原本档案就已经去除 x 的默认属性了，怎么可能突然间冒出来了？所以，这个地方得要特别小心喔！



在预设的情况下，root 的 umask 会拿掉比较多的属性，root 的 umask 默认是 022，这是基于安全的考虑啦～至于一般身份使用者，通常他们的 umask 为 002，亦即保留同群组的写入权力！其实，关于预设 umask 的设定可以参考 /etc/bashrc 这个档案的内容，不过，不建议修改该档案，你可以参考 bash shell 提到的环境参数配置文件 (~/.bashrc) 的说明～这部分我们在第三章的时候会提到！

🔒档案隐藏属性：

什么？档案还有隐藏属性？光是那九个权限就快要疯掉了，竟然还有隐藏属性，真是要命～但是没办法，就是有档案的隐藏属性存在啊！不过，这些隐藏的属性确实对于系统有很大的帮助的～尤其是在系统安全 (Security) 上面，重要的紧呢！不过要先强调的是，底下的 chattr 指令只能在 Ext2/Ext3 的文件系统上面生效，其他的文件系统可能就无法支持这个指令了。底下我们就来谈一谈如何设定与检查这些隐藏的属性吧！

- chattr (配置文件案隐藏属性)

```
[root@www ~]# chattr [+-=][ASacdistu] 档案或目录名称  
选项与参数：
```

+ : 增加某一个特殊参数, 其他原本存在参数则不动。
- : 移除某一个特殊参数, 其他原本存在参数则不动。
= : 设定一定, 且仅有后面接的参数

A : 当设定了 A 这个属性时, 若你有存取此档案(或目录)时, 他的访问时间 atime

将不会被修改, 可避免 I/O 较慢的机器过度的存取磁盘。这对速度较慢的计算机有帮助

S : 一般档案是异步写入磁盘的(原理请参考第五章 sync 的说明), 如果加上 S 这个

属性时, 当你进行任何档案的修改, 该更动会『同步』写入磁盘中。

a : 当设定 a 之后, 这个档案将只能增加数据, 而不能删除也不能修改数据, 只有 root

才能设定这个属性。

c : 这个属性设定之后, 将会自动的将此档案『压缩』, 在读取的时候将会自动解压缩,

但是在储存的时候, 将会先进行压缩后再储存(看来对于大档案似乎蛮有用的!)

d : 当 dump 程序被执行的时候, 设定 d 属性将可使该档案(或目录)不会被 dump 备份

i : 这个 i 可就很厉害了! 他可以让一个档案『不能被删除、改名、设定连结也无法

写入或新增资料!』对于系统安全性有相当大的帮助! 只有 root 能设定此属性

s : 当档案设定了 s 属性时, 如果这个档案被删除, 他将会被完全的移除出这个硬盘

空间, 所以如果误删了, 完全无法救回来了喔!

u : 与 s 相反的, 当使用 u 来配置文件案时, 如果该档案被删除了, 则数据内容其实还

存在磁盘中, 可以使用来救援该档案喔!

注意: 属性设定常见的是 a 与 i 的设定值, 而且很多设定值必须要身为 root 才能设定

范例: 请尝试到/tmp 底下建立档案, 并加入 i 的参数, 尝试删除看看。

```
[root@www ~]# cd /tmp
```

```
[root@www tmp]# touch attrtest <==建立一个空档案
```

```
[root@www tmp]# chattr +i attrtest <==给予 i 的属性
```

```
[root@www tmp]# rm attrtest <==尝试删除看看
```

```
rm: remove write-protected regular empty file
```

```
`attrtest'? y
```

```
rm: cannot remove `attrtest': Operation not permitted
```

```
<==操作不许可
# 看到了吗？呼呼！连 root 也没有办法将这个档案删除
呢！赶紧解除设定！
```

范例：请将该档案的 i 属性取消！

```
[root@www tmp]# chattr -i attrtest
```

这个指令是很重要的，尤其是在系统的数据安全上面！由于这些属性是隐藏的性质，所以需要以 `lsattr` 才能看到该属性哟！其中，个人认为最重要的当属 `+i` 与 `+a` 这个属性了。`+i` 可以让一个档案无法被更动，对于需要强烈的系统安全的人来说，真是相当的重要的！里头还有相当多的属性是需要 root 才能设定的呢！

此外，如果是 log file 这种的登录档，就更需要 `+a` 这个可以增加，但是不能修改旧有的数据与删除的参数了！怎样？很棒吧！未来提到[登录档](#)的认知时，我们再来聊一聊如何设定他吧！

-
- `lsattr` (显示档案隐藏属性)

```
[root@www ~]# lsattr [-adR] 档案或目录
选项与参数：
-a : 将隐藏文件的属性也秀出来；
-d : 如果接的是目录，仅列出目录本身的属性而非目录内的文件名；
-R : 连同子目录的数据也一并列出来！

[root@www tmp]# chattr +aij attrtest
[root@www tmp]# lsattr attrtest
----ia--j--- attrtest
```

使用 `chattr` 设定后，可以利用 `lsattr` 来查阅隐藏的属性。不过，这两个指令在使用上必须要特别小心，否则会造成很大的困扰。例如：某天你心情好，突然将 `/etc/shadow` 这个重要的密码记录档案给他设定成为具有 `i` 的属性，那么过了若干天之后，你突然要新增使用者，却一直无法新增！别怀疑，赶快去将 `i` 的属性拿掉吧！

 档案特殊权限： SUID, SGID, SBIT

我们前面一直提到关于档案的重要权限，那就是 `rwX` 这三个读、写、执行的权限。但是，眼尖的朋友们在[第六章的目录树章节](#)中，一定注意到了一件事，那

就是，怎么我们的 /tmp 权限怪怪的？ 还有，那个 /usr/bin/passwd 也怪怪的？ 怎么回事啊？ 看看先：

```
[root@www ~]# ls -ld /tmp ; ls -l /usr/bin/passwd
drwxrwxrwt 7 root root 4096 Sep 27 18:23 /tmp
-rwsr-xr-x 1 root root 22984 Jan 7 2007
/usr/bin/passwd
```

不是应该只有 rwx 吗？ 还有其他的特殊权限(s 跟 t)啊？ 啊.....头又开始昏了～ @_@ 因为 s 与 t 这两个权限的意义与[系统的账号及系统的程序\(process\)](#)较为相关， 所以等到后面的章节谈完后你才会比较有概念！ 底下的说明先看看就好， 如果看不懂也没有关系， 先知道 s 放在哪里称为 SUID/SGID 以及如何设定即可， 到系统程序章节读完后， 再回来看看喔！

- Set UID

当 s 这个标志出现在档案拥有者的 x 权限上时， 例如刚刚提到的 /usr/bin/passwd 这个档案的权限状态：『-rwsr-xr-x』， 此时就被称为 Set UID， 简称为 SUID 的特殊权限。 那么 SUID 的权限对于一个档案的特殊功能是什么呢？ 基本上 SUID 有这样的限制与功能：

- SUID 权限仅对二进制程序(binary program)有效；
- 执行者对于该程序需要具有 x 的可执行权限；
- 本权限仅在执行该程序的过程中有效 (run-time)；
- 执行者将具有该程序拥有者 (owner) 的权限。

讲这么硬的东西你可能对于 SUID 还是没有概念， 没关系， 我们举个例子来说明好了。 我们的 Linux 系统中， 所有账号的密码都记录在 /etc/shadow 这个档案里面， 这个档案的权限为：『-r----- 1 root root』， 意思是这个档案仅有 root 可读且仅有 root 可以强制写入而已。 既然这个档案仅有 root 可以修改， 那么鸟哥的 vbird 这个一般账号使用者能否自行修改自己的密码呢？ 你可以使用你自己的账号输入『passwd』这个指令来看看， 嘿嘿！ 一般用户当然可以修改自己的密码了！

唔！有没有冲突啊！明明 /etc/shadow 就不能让 vbird 这个一般账户去存取的， 为什么 vbird 还能够修改这个档案内的密码呢？ 这就是 SUID 的功能啦！藉由上述的功能说明， 我们可以知道

1. vbird 对于 /usr/bin/passwd 这个程序来说是具有 x 权限的， 表示 vbird 能执行 passwd；
2. passwd 的拥有者是 root 这个账号；
3. vbird 执行 passwd 的过程中， 会『暂时』获得 root 的权限；

4. /etc/shadow 就可以被 vbird 所执行的 passwd 所修改。

但如果 vbird 使用 cat 去读取 /etc/shadow 时,他能够读取吗? 因为 cat 不具有 SUID 的权限, 所以 vbird 执行『cat /etc/shadow』时,是不能读取 /etc/shadow 的。我们用一张示意图来说明如下:

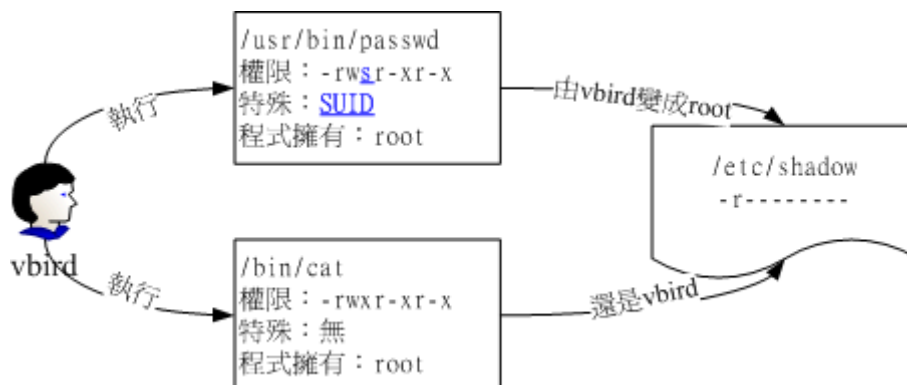


图 4.4.1、SUID 程序执行的过程示意图

另外, SUID 仅可用在 binary program 上, 不能够用在 shell script 上面! 这是因为 shell script 只是将很多的 binary 执行档叫进来执行而已! 所以 SUID 的权限部分, 还是得要看 shell script 呼叫进来的程序的设定, 而不是 shell script 本身。当然, SUID 对于目录也是无效的~这点要特别留意。

- Set GID

当 s 标志在档案拥有者的 x 项目为 SUID, 那 s 在群组的 x 时则称为 Set GID, SGID 啰! 是这样没错! ^_^。举例来说, 你可以用底下的指令来观察到具有 SGID 权限的档案喔:

```
[root@www ~]# /usr/bin/locate
-rwx--s--x 1 root slocate 23856 Mar 15 2007
/usr/bin/locate
```

与 SUID 不同的是, SGID 可以针对档案或目录来设定! 如果是对档案来说, SGID 有如下的功能:

- SGID 对二进制程序有用;
- 程序执行者对于该程序来说, 需具备 x 的权限;
- 执行者在执行的过程中将会获得该程序群组的支持!

举例来说, 上面的 /usr/bin/locate 这个程序可以去搜寻 /var/lib/mlocate/mlocate.db 这个档案的内容 (详细说明会在下节讲述),

mlocate.db 的权限如下：

```
[root@www ~]# ll /usr/bin/locate
/var/lib/mlocate/mlocate.db
-rwx--s--x 1 root slocate 23856 Mar 15 2007
/usr/bin/locate
-rw-r----- 1 root slocate 3175776 Sep 28 04:02
/var/lib/mlocate/mlocate.db
```

与 SUID 非常的类似，若我使用 vbird 这个账号去执行 locate 时，那 vbird 将会取得 slocate 群组的支持，因此就能够去读取 mlocate.db 啦！非常有趣吧！

除了 binary program 之外，事实上 SGID 也能够用在目录上，这也是非常常见的一种用途！当一个目录设定了 SGID 的权限后，他将具有如下的功能：

- 用户若对于此目录具有 r 与 x 的权限时，该用户能够进入此目录；
- 用户在此目录下的有效群组(effective group)将会变成该目录的群组；
- 用途：若用户在此目录下具有 w 的权限(可以新建档案)，则使用者所建立的新档案，该档案的群组与此目录的群组相同。

SGID 对于项目开发来说是非常重要的！因为这涉及群组权限的问题，您可以参考一下本章后续[情境模拟的案例](#)，应该就能够对于 SGID 有一些了解的！^_^

-
- Sticky Bit

这个 Sticky Bit, SBIT 目前只针对目录有效，对于档案已经没有效果了。SBIT 对于目录的作用是：

- 当用户对于此目录具有 w, x 权限，亦即具有写入的权限时；
- 当用户在该目录下建立档案或目录时，仅有自己与 root 才有权力删除该档案

换句话说：当甲这个用户于 A 目录是具有群组或其他人的身份，并且拥有该目录 w 的权限，这表示『甲用户对该目录内任何人建立的目录或档案均可进行“删除/更名/搬移”等动作。』不过，如果将 A 目录加上了 SBIT 的权限项目时，则甲只能够针对自己建立的档案或目录进行删除/更名/移动等动作，而无法删除他人的档案。

举例来说，我们的 /tmp 本身的权限是『drwxrwxrwt』，在这样的权限内容下，任何人都可以在 /tmp 内新增、修改档案，但仅有该档案/目录建立者与 root 能够删除自己的目录或档案。这个特性也是挺重要的啊！你可以这样做个简单的测试：

1. 以 root 登入系统，并且进入 /tmp 当中；
2. touch test，并且更改 test 权限成为 777 ；
3. 以一般使用者登入，并进入 /tmp；
4. 尝试删除 test 这个档案！

由于 SUID/SGID/SBIT 牵涉到程序的概念，因此再次强调，这部份的数据在您读完[第十七章关于程序方面](#)的知识后，要再次的回来瞧瞧喔！目前，你先有个简单的基础概念就好了！文末的参考数据也建议阅读一番喔！

- SUID/SGID/SBIT 权限设定

前面介绍过 SUID 与 SGID 的功能，那么如何配置文件案使成为具有 SUID 与 SGID 的权限呢？这就需要[第六章的数字更改权限](#)的方法了！现在你应该已经知道数字型态更改权限的方式为『三个数字』的组合，那么如果在这三个数字之前再加上一个数字的话，最前面的那个数字就代表这几个权限了！

- 4 为 SUID
- 2 为 SGID
- 1 为 SBIT

假设要将一个档案权限改为『-rwsr-xr-x』时，由于 s 在用户权力中，所以是 SUID，因此，在原先的 755 之前还要加上 4，也就是：『chmod 4755 filename』来设定！此外，还有大 S 与大 T 的产生喔！参考底下的范例啦！

Tips:

注意：底下的范例只是练习而已，所以鸟哥使用同一个档案来设定，你必须了解 SUID 不是用在目录上，而 SBIT 不是用在档案上的喔！




```
[root@www ~]# cd /tmp
[root@www tmp]# touch test <==建立一个测试用空档
[root@www tmp]# chmod 4755 test; ls -l test <==加入具有 SUID 的权限
-rwsr-xr-x 1 root root 0 Sep 29 03:06 test
[root@www tmp]# chmod 6755 test; ls -l test <==加入具有 SUID/SGID 的权限
-rwsr-sr-x 1 root root 0 Sep 29 03:06 test
[root@www tmp]# chmod 1755 test; ls -l test <==加入 SBIT 的功能!
-rwxr-xr-t 1 root root 0 Sep 29 03:06 test
[root@www tmp]# chmod 7666 test; ls -l test <==具有
```


空的 SUID/SGID 权限

```
-rwSrwsrwT 1 root root 0 Sep 29 03:06 test
```


最后一个例子就要特别小心啦！怎么会出现大写的 S 与 T 呢？不都是小写的吗？因为 s 与 t 都是取代 x 这个权限的，但是你有没有发现阿，我们是下达 7666 喔！也就是说，user, group 以及 others 都没有 x 这个可执行的标志（因为 666 嘛），所以，这个 S, T 代表的就是『空的』啦！怎么说？SUID 是表示『该档案在执行的时候，具有档案拥有者的权限』，但是档案拥有者都无法执行了，哪里来的权限给其他人使用？当然就是空的啦！^_^

 观察文件类型：file

如果你要知道某个档案的基本数据，例如是属于 ASCII 或者是 data 档案，或者是 binary，且其中有没有使用到动态函式库（share library）等的信息，就可以利用 file 这个指令来检阅喔！举例来说：

```
[root@www ~]# file ~/.bashrc
/root/.bashrc: ASCII text <==告诉我们是 ASCII 的纯
文本档啊！
[root@www ~]# file /usr/bin/passwd
/usr/bin/passwd: setuid ELF 32-bit LSB executable,
Intel 80386, version 1
(SYSV), for GNU/Linux 2.6.9, dynamically linked (uses
shared libs), for
GNU/Linux 2.6.9, stripped
# 执行文件的数据可就多的不得了！包括这个档案的 suid
权限、兼容于 Intel 386
# 等级的硬件平台、使用的是 Linux 核心 2.6.9 的动态函
式库链接等等。
[root@www ~]# file /var/lib/mlocate/mlocate.db
/var/lib/mlocate/mlocate.db: data <== 这是 data 档
案！
```

透过这个指令，我们可以简单的先判断这个档案的格式为何喔！

 指令与档案的搜寻：

档案的搜寻可就厉害了！因为我们常常需要知道那个档案放在哪里，才能够对该档案进行一些修改或维护等动作。有些时候某些软件配置文件的文件名是不变

的，但是各 distribution 放置的目录则不同。此时就得要利用一些搜寻指令将该配置文件的完整档名捉出来，这样才能修改嘛！您说是吧！^_^

🐼脚本文件名的搜寻：

我们知道在终端机模式当中，连续输入两次[tab]按键就能够知道用户有多少指令可以下达。那你知不知道这些指令的完整文件名放在哪里？举例来说，ls 这个常用的指令放在哪里呢？就透过 which 或 type 来找寻吧！

- which (寻找『执行档』)

```
[root@www ~]# which [-a] command
选项或参数：
-a : 将所有由 PATH 目录中可以找到的指令均列出，而不止第一个被找到的指令名称

范例一：分别用 root 与一般账号搜寻 ifconfig 这个指令的完整文件名
[root@www ~]# which ifconfig
/sbin/ifconfig          <==用 root 可以找到正确的执行档名喔！
[root@www ~]# su - vbird <==切换身份成为 vbird 去！
[vbird@www ~]$ which ifconfig
/usr/bin/which: no ifconfig in
(/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin
:/home/vbird/bin)      <==见鬼了！竟然一般身份账号找不到！
# 因为 which 是根据用户所设定的 PATH 变量内的目录去搜寻可执行文件的！所以，
# 不同的 PATH 设定内容所找到的指令当然不一样啦！因为 /sbin 不在 vbird 的
# PATH 中，找不到也是理所当然的啊！瞭乎？
[vbird@www ~]$ exit      <==记得将身份切换回原本的 root

范例二：用 which 去找出 which 的档名为何？
[root@www ~]# which which
alias which='alias | /usr/bin/which --tty-only
--read-alias --show-dot '
/usr/bin/which
# 竟然会有两个 which，其中一个是 alias 这玩意儿呢！那是
```

```
啥？
# 那就是所谓的『命令别名』，意思是输入 which 会等于后面接
的那串指令啦！
# 更多的数据我们会在 bash 章节中再来谈的！

范例三：请找出 cd 这个指令的完整文件名
[root@www ~]# which cd
/usr/bin/which: no cd in
(/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin
:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin)
# 瞎密？怎么可能没有 cd，我明明就能够用 root 执行 cd 的
啊！
```

这个指令是根据『[PATH](#)』这个环境变量所规范的路径，去搜寻『执行档』的档名～所以，重点是找出『执行档』而已！且 which 后面接的是『完整档名』喔！若加上 -a 选项，则可以列出所有的可以找到的同名执行文件，而非仅显示第一个而已！

最后一个范例最有趣，怎么 cd 这个常用的指令竟然找不到啊！为什么呢？这是因为 cd 是『bash 内建的指令』啦！但是 which 预设是找 PATH 内所规范的目录，所以当然一定找不到的啊！那怎办？没关系！我们可以透过 type 这个指令喔！关于 type 的用法我们将在 [bash 章节](#)中再来谈！

🔍 档案档名的搜寻：

再来谈一谈怎么搜寻档案吧！在 Linux 底下也有相当优异的搜寻指令啦！通常 find 不很常用的！因为速度慢之外，也很操硬盘！通常我们都是先使用 whereis 或者是 locate 来检查，如果真的找不到了，才以 find 来搜寻啦！为什么呢？因为 whereis 与 locate 是利用数据库来搜寻数据，所以相当的快速，而且并没有实际的搜寻硬盘，比较省时间啦！

-
- whereis (寻找特定档案)

```
[root@www ~]# whereis [-bmsu] 档案或目录名
选项与参数：
-b      :只找 binary 格式的档案
-m      :只找在说明文件 manual 路径下的档案
-s      :只找 source 来源档案
-u      :搜寻不在上述三个项目当中的其他特殊档案
```

```

范例一：请用不同的身份找出 ifconfig 这个档名
[root@www ~]# whereis ifconfig
ifconfig: /sbin/ifconfig
/usr/share/man/man8/ifconfig.8.gz
[root@www ~]# su - vbird      <==切换身份成为 vbird
[vbird@www ~]$ whereis ifconfig <==找到同样的结果喔！
ifconfig: /sbin/ifconfig
/usr/share/man/man8/ifconfig.8.gz
[vbird@www ~]$ exit          <==回归身份成为
root 去！
# 注意看，明明 which 一般使用者找不到的 ifconfig 却
可以让 whereis 找到！
# 这是因为系统真的有 ifconfig 这个『档案』，但是使用
者的 PATH 并没有加入 /sbin
# 所以，未来你找不到某些指令时，先用档案搜寻指令找找
看再说！

范例二：只找出跟 passwd 有关的『说明文件』档名(man
page)
[root@www ~]# whereis -m passwd
passwd: /usr/share/man/man1/passwd.1.gz
/usr/share/man/man5/passwd.5.gz

```

等一下我们会提到 find 这个搜寻指令，find 是很强大的搜寻指令，但时间花用的很大！（因为 find 是直接搜寻硬盘，为如果你的硬盘比较老旧的话，嘿嘿！有的等！）这个时候 whereis 就相当的好用了！另外，whereis 可以加入选项来找寻相关的数据，例如如果你是要找可执行文件（binary）那么加上 -b 就可以啦！如果不加任何选项的话，那么就将所有的数据列出来啰！

那么 whereis 到底是使用什么咚咚呢？为何搜寻的速度会比 find 快这么多？其实那也没有什么！这是因为 Linux 系统会将系统内的所有档案都记录在一个数据库档案里面，而当使用 whereis 或者是底下要说的 locate 时，都会以此数据库档案的内容为准，因此，有的时后你还会发现使用这两个执行档时，会找到已经被杀掉的档案！而且也找不到最新的刚刚建立的档案呢！这就是因为这两个指令是由数据库当中的结果去搜寻档案的所在啊！更多与这个数据库有关的说明，请参考下列的 locate 指令。

-
- locate

```
[root@www ~]# locate [-ir] keyword
```

选项与参数：

- i : 忽略大小写的差异；
- r : 后面可接正规表示法的显示方式

范例一：找出系统中所有与 passwd 相关的档名

```
[root@www ~]# locate passwd
/etc/passwd
/etc/passwd-
/etc/news/passwd.nntp
/etc/pam.d/passwd
.... (底下省略)....
```

这个 locate 的使用更简单，直接在后面输入『档案的部分名称』后，就能够得到结果。举上面的例子来说，我输入 locate passwd，那么在完整文件名（包含路径名称）当中，只要有 passwd 在其中，就会被显示出来的！这也是个很方便好用的指令，如果你忘记某个档案的完整档名时～～

但是，这个东西还是有使用上的限制呦！为什么呢？你会发现使用 locate 来寻找数据的时候特别的快，这是因为 locate 寻找的数据是由『已建立的数据库 /var/lib/mlocate/』里面的数据所搜寻到的，所以不用直接去硬盘当中存取数据，呵呵！当然是很快速啰！

那么有什么限制呢？就是因为他是经由数据库来搜寻的，而数据库的建立默认是在每天执行一次（每个 distribution 都不同，CentOS 5.x 是每天更新数据库一次！），所以当你新建立起来的档案，却还在数据库更新之前搜寻该档案，那么 locate 会告诉你『找不到！』呵呵！因为必须要更新数据库呀！

那能否手动更新数据库哪？当然可以啊！更新 locate 数据库的方法非常简单，直接输入『updatedb』就可以了！updatedb 指令会去读取 /etc/updatedb.conf 这个配置文件的设定，然后再去硬盘里面进行搜寻文件名的动作，最后就更新整个数据库档案啰！因为 updatedb 会去搜寻硬盘，所以当你执行 updatedb 时，可能会等待数分钟的时间喔！

- updatedb: 根据 /etc/updatedb.conf 的设定去搜寻系统硬盘内的文件名，并更新 /var/lib/mlocate 内的数据库档案；
- locate: 依据 /var/lib/mlocate 内的数据库记载，找出用户输入的关键词文件名。

-
- find

```
[root@www ~]# find [PATH] [option] [action]
```

选项与参数:

1. 与时间有关的选项: 共有 `-atime`, `-ctime` 与 `-mtime`, 以 `-mtime` 说明

`-mtime n`: `n` 为数字, 意义为在 `n` 天之前的『一天之内』被更动过内容的档案;

`-mtime +n`: 列出在 `n` 天之前(不含 `n` 天本身)被更动过内容的档案档名;

`-mtime -n`: 列出在 `n` 天之内(含 `n` 天本身)被更动过内容的档案档名。

`-newer file`: `file` 为一个存在的档案, 列出比 `file` 还要新的档案档名

范例一: 将过去系统上面 24 小时内有更动过内容 (`mtime`) 的档案列出

```
[root@www ~]# find / -mtime 0
```

那个 0 是重点! 0 代表目前的时间, 所以, 从现在开始到 24 小时前,

有变动过内容的档案都会被列出来! 那如果是三天前的 24 小时内?

`find / -mtime 3` 有变动过的档案都被列出的意思!

范例二: 寻找 `/etc` 底下的档案, 如果档案日期比 `/etc/passwd` 新就列出

```
[root@www ~]# find /etc -newer /etc/passwd
```

`-newer` 用在分辨两个档案之间的新旧关系是很有用的!

时间参数真是挺有意思的! 我们现在知道 `atime`, `ctime` 与 `mtime` 的意义, 如果你想要找出一天内被更动过的文件名, 可以使用上述范例一的作法。但如果我想要找出『4 天内被更动过的档案档名』呢? 那可以使用『`find /var -mtime -4`』。那如果是『4 天前的那一天』就用『`find /var -mtime 4`』。有没有加上『+, -』差别很大喔! 我们可以用简单的图示来说明一下:

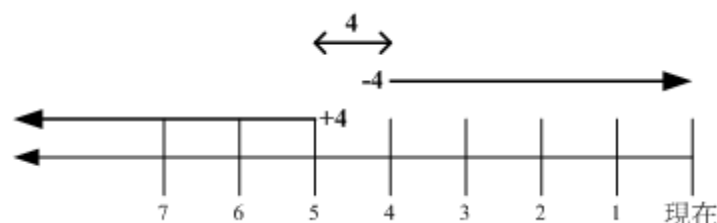


图 5.2.1、find 相关的时间参数意义

图中最右边为目前的时间, 越往左边则代表越早之前的时间轴啦。由图 5.2.1 我们可以清楚的知道:

- `+4` 代表大于等于 5 天前的档名: `ex> find /var -mtime +4`

- -4 代表小于等于 4 天内的档案档名：ex> find /var -mtime -4
- 4 则是代表 4-5 那一天的档案档名：ex> find /var -mtime 4

非常有趣吧！你可以在 /var/ 目录下搜寻一下，感受一下输出档案的差异喔！再来看看其他 find 的用法吧！

选项与参数：

2. 与使用者或组名有关的参数：

-uid n : n 为数字，这个数字是用户的账号 ID，亦即 UID，这个 UID 是记录在

/etc/passwd 里面与账号名称对应的数字。这方面我们会在第四篇介绍。

-gid n : n 为数字，这个数字是组名的 ID，亦即 GID，这个 GID 记录在

/etc/group，相关的介绍我们会第四篇说明～

-user name : name 为使用者账号名称喔！例如 dmtsai

-group name: name 为组名喔，例如 users；

-nouser : 寻找档案的拥有者不存在 /etc/passwd 的人！

-nogroup : 寻找档案的拥有群组不存在于 /etc/group 的档案！

当你自行安装软件时，很可能该软件的属性当中并没有档案拥有者，

这是可能的！在这个时候，就可以使用 -nouser 与 -nogroup 搜寻。

范例三：搜寻 /home 底下属于 vbird 的档案

```
[root@www ~]# find /home -user vbird
```

这个东西也很有用的～当我们要找出任何一个用户在系统当中的所有档案时，

就可以利用这个指令将属于某个使用者的所有档案都找出来喔！

范例四：搜寻系统中不属于任何人的档案

```
[root@www ~]# find / -nouser
```

透过这个指令，可以轻易的就找出那些不太正常的档案。

如果有找到不属于系统任何人的档案时，不要太紧张，

那有时候是正常的～尤其是你曾经以原始码自行编译软件时。

如果你想要找出某个用户在系统底下建立了啥咚咚，使用上述的选项与参数，就能够找出来啦！至于那个 -nouser 或 -nogroup 的选项功能中，除了你自行由网络上下载文件时会发生之外，如果你将系统里面某个账号删除了，但是该账号已经在系统内建立很多档案时，就可能会发生无主孤魂的档案存在！此时

你就得使用这个 `-nouser` 来找出该类型的档案啰！

选项与参数：

3. 与档案权限及名称有关的参数：

`-name filename`：搜寻文件名为 `filename` 的档案；

`-size [+ -]SIZE`：搜寻比 `SIZE` 还要大(+)或小(-)的档案。这个 `SIZE` 的规格有：

`c`：代表 byte，`k`：代表 1024bytes。

所以，要找比 50KB

还要大的档案，就是『`-size +50k`』

`-type TYPE`：搜寻档案的类型为 `TYPE` 的，类型主要有：一般正规档案 (`f`)，

装置档案 (`b`, `c`)，目录 (`d`)，连结档 (`l`)，socket (`s`)，

及 FIFO (`p`) 等属性。

`-perm mode`：搜寻档案权限『刚好等于』`mode` 的档案，这个 `mode` 为类似 `chmod`

的属性值，举例来说，`-rwsr-xr-x` 的属性为 4755 ！

`-perm -mode`：搜寻档案权限『必须要全部囊括 `mode` 的权限』的档案，举例来说，

我们要搜寻 `-rwxr--r--`，亦即 0744 的档案，使用 `-perm -0744`，

当一个档案的权限为 `-rwsr-xr-x`，亦即 4755 时，也会被列出来，

因为 `-rwsr-xr-x` 的属性已经囊括了 `-rwxr--r--` 的属性了。

`-perm +mode`：搜寻档案权限『包含任一 `mode` 的权限』的档案，举例来说，我们搜寻

`-rwxr-xr-x`，亦即 `-perm +755` 时，但一个文件属性为 `-rw-----`

也会被列出来，因为他有 `-rw....` 的属性存在！

范例五：找出档名为 `passwd` 这个档案

```
[root@www ~]# find / -name passwd
```

利用这个 `-name` 可以搜寻档名啊！

范例六：找出 `/var` 目录下，文件类型为 Socket 的档名有哪些？

```
[root@www ~]# find /var -type s
```

这个 `-type` 的属性也很有帮助喔！尤其是要找出那些怪异的档案，


```
# 例如 socket 与 FIFO 档案，可以用 find /var -type p  
或 -type s 来找！
```

范例七：搜寻档案当中含有 SGID 或 SUID 或 SBIT 的属性

```
[root@www ~]# find / -perm +7000
```

所谓的 7000 就是 ---s--s--t，那么只要含有 s 或 t 的就列出，

所以当然要使用 +7000，使用 -7000 表示要含有 ---s--s--t 的所有三个权限，

因此，就是 +7000 嘛！

上述范例中比较有趣的就属 -perm 这个选项啦！他的重点在找出特殊权限的档案！我们知道 SUID 与 SGID 都可以设定在二进制程序上，假设我想要找出来 /bin, /sbin 这两个目录下，只要具有 SUID 或 SGID 就列出来该档案，你可以这样做：

```
[root@www ~]# find /bin /sbin -perm +6000
```

因为 SUID 是 4 分，SGID 2 分，总共为 6 分，因此可用 +6000 来处理这个权限！至于 find 后面可以接多个目录来进行搜寻！另外，find 本来就会搜寻次目录，这个特色也要特别注意喔！最后，我们再来看一下 find 还有什么特殊功能吧！

选项与参数：

4. 额外可进行的动作：

-exec command : command 为其他指令，-exec 后面可再接额外的指令来处理搜寻到的结果。

-print : 将结果打印到屏幕上，这个动作是预设动作！

范例八：将上个范例找到的档案使用 ls -l 列出来～

```
[root@www ~]# find / -perm +7000 -exec ls -l {} \;
```

注意到，那个 -exec 后面的 ls -l 就是额外的指令，指令不支持命令别名，

所以仅能使用 ls -l 不可以使用 ll 喔！注意注意！

范例九：找出系统中，大于 1MB 的档案

```
[root@www ~]# find / -size +1000k
```

虽然在 man page 提到可以使用 M 与 G 分别代表 MB 与 GB，

不过，俺却试不出来这个功能～所以，目前应该是仅支持

```
到 c 与 k 吧!
```

find 的特殊功能就是能够进行额外的动作(action)。我们将范例八的例子以图解来说明如下：

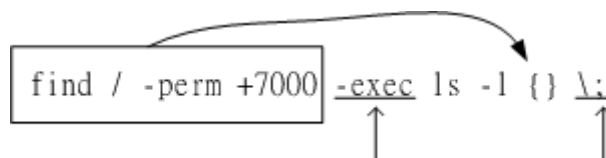


图 5.2.2、find 相关的额外动作

该范例中特殊的地方有 {} 以及 \; 还有 -exec 这个关键词，这些东西的意义为：

- {} 代表的是『由 find 找到的内容』，如上图所示，find 的结果会被放置到 {} 位置中；
- -exec 一直到 \; 是关键词，代表 find 额外动作的开始 (-exec) 到结束 (\;)，在这中间的就是 find 指令内的额外动作。在本例中就是『ls -l {}』啰！
- 因为『;』在 bash 环境下是有特殊意义的，因此利用反斜杠来跳脱。

透过图 5.2.2 你应该就比较容易了解 -exec 到 \; 之间的意义了吧！

如果你要找的档案是具有特殊属性的，例如 SUID、档案拥有者、档案大小等等，那么利用 locate 是没有办法达成你的搜寻的！此时 find 就显的很重要啦！另外，find 还可以利用通配符来找寻档名呢！举例来说，你想要找出 /etc 底下档名包含 httpd 的档案，那么你就可以这样做：

```
[root@www ~]# find /etc -name '*httpd*'
```

不但可以指定搜寻的目录(连同次目录)，并且可以利用额外的选项与参数来找到最正确的档名！真是好好用！不过由于 find 在寻找数据的时候相当的操硬盘！所以没事情不要使用 find 啦！有更棒的指令可以取代啦！那就是上面提到的 [whereis](#) 与 [locate](#) 啰！！



极重要！权限与指令间的关系：

我们知道权限对于使用者账号来说是非常重要的，因为他可以限制使用者能不能读取/建立/删除/修改档案或目录！在这一章我们介绍了很多文件系统的管理指令，第六章则介绍了很多档案权限的意义。在这个小节当中，我们就将这两者结合起来，说明一下什么指令在什么样的权限下才能够运作吧！^_^

一、让用户能进入某目录成为『可工作目录』的基本权限为何：

- 可使用的指令：例如 `cd` 等变换工作目录的指令；
- 目录所需权限：用户对这个目录至少需要具有 `x` 的权限
- 额外需求：如果用户想要在这个目录内利用 `ls` 查阅文件名，则用户对此目录还需要 `r` 的权限。

二、用户在某个目录内读取一个档案的基本权限为何？

- 可使用的指令：例如本章谈到的 `cat`, `more`, `less` 等等
- 目录所需权限：用户对这个目录至少需要具有 `x` 权限；
- 档案所需权限：使用者对档案至少需要具有 `r` 的权限才行！

三、让使用者可以修改一个档案的基本权限为何？

- 可使用的指令：例如 `nano` 或未来要介绍的 `vi` 编辑器等；
- 目录所需权限：用户在该档案所在的目录至少要有 `x` 权限；
- 档案所需权限：使用者对该档案至少要有 `r, w` 权限

四、让一个使用者可以建立一个档案的基本权限为何？

- 目录所需权限：用户在该目录要具有 `w, x` 的权限，重点在 `w` 啦！

五、让用户进入某目录并执行该目录下的某个指令之基本权限为何？

- 目录所需权限：用户在该目录至少要有 `x` 的权限；
- 档案所需权限：使用者在该档案至少需要有 `x` 的权限

例题：

让一个使用者 `vbird` 能够进行『`cp /dir1/file1 /dir2`』的指令时，请说明 `dir1`, `file1`, `dir2` 的最小所需权限为何？

答：

执行 `cp` 时，`vbird` 要『能够读取来源文件，并且写入目标文件！』所以应参考上述第二点与第四点的说明！因此各档案/目录的最小权限应该是：

- `dir1` : 至少需要有 `x` 权限；
- `file1`: 至少需要有 `r` 权限；
- `dir2` : 至少需要有 `w, x` 权限。

例题：

有一个档案全名为 `/home/student/www/index.html`，各相关档案/目录的权限如下：

```
drwxr-xr-x 23 root    root    4096 Sep 22 12:09 /
drwxr-xr-x  6 root    root    4096 Sep 29 02:21 /home
drwx----- 6 student student 4096 Sep 29 02:23 /home/student
drwxr-xr-x  6 student student 4096 Sep 29 02:24 /home/student/www
-rwxr--r--  6 student student 369 Sep 29 02:27
/home/student/www/index.html
```

请问 `vbird` 这个账号(不属于 `student` 群组)能否读取 `index.html` 这个档案呢？

答：

虽然 `www` 与 `index.html` 是可以让 `vbird` 读取的权限，但是因为目录结构是由根目录一层一层读取的，因此 `vbird` 可进入 `/home` 但是却不可进入 `/home/student/`，既然连进入 `/home/student` 都不许了，当然就读不到 `index.html` 了！所以答案是『`vbird` 不会读取到 `index.html` 的内容』喔！

那要如何修改权限呢？其实只要将 `/home/student` 的权限修改为最小 `711`，或者直接给予 `755` 就可以啰！这可是很重要的概念喔！



重点回顾

- 绝对路径：『一定由根目录 `/` 写起』；相对路径：『不是由 `/` 写起』
- 特殊目录有：`.`，`..`，`-`，`~`，`~account` 需要注意；
- 与目录相关的指令有：`cd`，`mkdir`，`rmdir`，`pwd` 等重要指令；
- `rmdir` 仅能删除空目录，要删除非空目录需使用『`rm -r`』指令；
- 用户能使用的指令是依据 `PATH` 变量所规定的目录去搜寻的；
- 不同的身份(`root` 与一般用户)系统默认的 `PATH` 并不相同。差异较大的地方在于 `/sbin`，`/usr/sbin`；
- `ls` 可以检视档案的属性，尤其 `-d`，`-a`，`-l` 等选项特别重要！
- 档案的复制、删除、移动可以分别使用：`cp`，`rm`，`mv` 等指令来操作；
- 检查档案的内容(读文件)可使用的指令包括有：`cat`，`tac`，`nl`，`more`，`less`，`head`，`tail`，`od` 等
- `cat -n` 与 `nl` 均可显示行号，但默认的情况下，空白行会不会编号并不相同；
- `touch` 的目的在修改档案的时间参数，但亦可用来建立空档案；
- 一个档案记录的时间参数有三种，分别是 `access time(ctime)`，`status time(mtime)`，`modification time(mtime)`，`ls` 默认显示的是 `mtime`。
- 除了传统的 `rwX` 权限之外，在 `Ext2/Ext3` 文件系统中，还可以使用 `chattr` 与 `lsattr` 设定及观察隐藏属性。常见的包括只能新增数据的 `+a` 与完全不能更动档案的 `+i` 属性。
- 新建档案/目录时，新档案的预设权限使用 `umask` 来规范。默认目录完全

权限为 drwxrwxrwx， 档案则为-rw-rw-rw-。

- 档案具有 SUID 的特殊权限时，代表当用户执行此一 binary 程序时，在执行过程中用户会暂时具有程序拥有者的权限
- 目录具有 SGID 的特殊权限时，代表用户在这个目录下新建的档案之群组都会与该目录的组名相同。
- 目录具有 SBIT 的特殊权限时，代表在该目录下用户建立的档案只有自己与 root 能够删除！
- 观察档案的类型可以使用 file 指令来观察；
- 搜寻指令的完整文件名可用 which 或 type，这两个指令都是透过 PATH 变量来搜寻文件名；
- 搜寻档案的完整档名可以使用 whereis 或 locate 到数据库档案去搜寻，而不实际搜寻文件系统；
- 利用 find 可以加入许多选项来直接查询文件系统，以获得自己想知道的档名。



本章习题：

（ 要看答案请将鼠标移动到『答：』底下的空白处，按下左键圈选空白处即可察看 ）

情境模拟题一：假设系统中有两个账号，分别是 alex 与 arod，这两个人除了自己群组之外还共同支持一个名为 project 的群组。假设这两个用户需要共同拥有 /srv/ahome/ 目录的开发权，且该目录不许其他人进入查阅。请问该目录的权限设定应为何？请先以传统权限说明，再以 SGID 的功能解析。

- 目标：了解到为何项目开发时，目录最好需要设定 SGID 的权限！
- 前提：多个账号支持同一群组，且共同拥有目录的使用权！
- 需求：需要使用 root 的身份来进行 chmod, chgrp 等帮用户设定好他们的开发环境才行！ 这也是管理员的重要任务之一！

首先我们得要先制作出这两个账号的相关数据，账号/群组的管理在后续我们会介绍，您这里先照着底下的指令来制作即可：

```
[root@www ~]# groupadd project          <==增加新的群组
[root@www ~]# useradd -G project alex <==建立 alex 账号，且支持 project
[root@www ~]# useradd -G project arod <==建立 arod 账号，且支持 project
[root@www ~]# id alex                   <==查阅 alex 账号的属性
uid=501(alex) gid=502(alex)
groups=502(alex), 501(project) <==确实有支持！
[root@www ~]# id arod
```

```
uid=502(arod) gid=503(arod)
groups=503(arod), 501(project)
```

然后开始来解决我们所需要的环境吧！

1. 首先建立所需要开发的项目目录：

```
[root@www ~]# mkdir /srv/ahome
[root@www ~]# ll -d /srv/ahome
drwxr-xr-x 2 root root 4096 Sep 29 22:36 /srv/ahome
```

2. 从上面的输出结果可发现 alex 与 arod 都不能在该目录内建立档案，因此需要进行权限与属性的修改。由于其他人均不可进入此目录，因此该目录的群组应为 project，权限应为 770 才合理。

```
[root@www ~]# chgrp project /srv/ahome
[root@www ~]# chmod 770 /srv/ahome
[root@www ~]# ll -d /srv/ahome
drwxrwx--- 2 root project 4096 Sep 29 22:36 /srv/ahome
# 从上面的权限结果来看，由于 alex/arod 均支持
project，因此似乎没问题了！
```

3. 实际分别以两个使用者来测试看看，情况会是如何？先用 alex 建立档案，然后用 arod 去处理看看。

```
[root@www ~]# su - alex          <==先切换身份成为 alex
来处理
[alex@www ~]$ cd /srv/ahome      <==切换到群组的工作目
录去
[alex@www ahome]$ touch abcd     <==建立一个空的档案出
来！
[alex@www ahome]$ exit           <==离开 alex 的身份

[root@www ~]# su - arod
[arod@www ~]$ cd /srv/ahome
[arod@www ahome]$ ll abcd
-rw-rw-r-- 1 alex alex 0 Sep 29 22:46 abcd
# 仔细看一下上面的档案，由于群组是 alex，arod 并不支
持！
# 因此对于 abcd 这个档案来说，arod 应该只是其他人，
只有 r 的权限而已啊！
[arod@www ahome]$ exit
```

由上面的结果我们可以知道，若单纯使用传统的 `rwX` 而已，则对刚刚 alex 建立的 `abcd` 这个档案来说，`arod` 可以删除他，但是却不能编辑他！这不是我们要的样子啊！赶紧来重新规划一下。

4. 加入 `SGID` 的权限在里面，并进行测试看看：

```
[root@www ~]# chmod 2770 /srv/ahome
[root@www ~]# ll -d /srv/ahome
drwxrws--- 2 root project 4096 Sep 29 22:46 /srv/ahome

测试：使用 alex 去建立一个档案，并且查阅档案权限看看：
[root@www ~]# su - alex
[alex@www ~]$ cd /srv/ahome
[alex@www ahome]$ touch 1234
[alex@www ahome]$ ll 1234
-rw-rw-r-- 1 alex project 0 Sep 29 22:53 1234
# 没错！这才是我们要的样子！现在 alex, arod 建立的新
档案所属群组都是 project，
# 由于两人均属于此群组，加上 umask 都是 002，这样两
人才可以互相修改对方的档案！
```

所以最终的结果显示，此目录的权限最好是『2770』，所属档案拥有者属于 `root` 即可，至于群组必须要为两人共同支持的 `project` 这个群组才行！

-
- 什么是绝对路径与相对路径

绝对路径的写法为由 `/` 开始写，至于相对路径则不由 `/` 开始写！此外，相对路径为相对于目前工作目录的路径！

- 如何更改一个目录的名称？例如由 `/home/test` 变为 `/home/test2`

```
mv /home/test /home/test2
```

- `PATH` 这个环境变量的意义？

这个是用来指定执行文件执行的时候，指令搜寻的目录路径。

- `umask` 有什么用处与优点？

`umask` 可以拿掉一些权限，因此，适当的定义 `umask` 有助于系统的安全，因为他可以用来建立默认的目录或档案的权限。

- 当一个使用者的 umask 分别为 033 与 044 他所建立的档案与目录的权限为何？

在 umask 为 033 时，则预设是拿掉 group 与 other 的 w(2)x(1) 权限，因此权限就成为『档案 -rw-r--r--，目录 drwxr--r--』而当 umask 044 时，则拿掉 r 的属性，因此就成为『档案 -rw--w--w-，目录 drwx-wx-wx』

- 什么是 SUID ？

当一个指令具有 SUID 的功能时，则：

- SUID 权限仅对二进制程序(binary program)有效；
 - 执行者对于该程序需要具有 x 的可执行权限；
 - 本权限仅在执行该程序的过程中有效 (run-time)；
 - 执行者将具有该程序拥有者 (owner) 的权限。
- 当我要查询 /usr/bin/passwd 这个档案的一些属性时(1)传统权限；(2)文件类型与(3)档案的隐藏属性，可以使用什么指令来查询？

```
ls -al
file
lsattr
```

- 尝试用 find 找出目前 linux 系统中，所有具有 SUID 的档案有哪些？

```
find / -perm +4000 -print
```



参考数据与延伸阅读

- 小洲大大回答 SUID/SGID 的一篇讨论：
<http://phorum.vbird.org/viewtopic.php?t=20256>

2002/06/26: 第一次完成 2003/02/06: 重新编排与加入 FAQ

2003/02/07: 加入 basename 与 dirname 的说明

2004/03/15: 将连结档的内容移动至下一章节: [Linux 磁盘与硬件管理](#)

2005/07/19: 将旧的文章移动到 [这里](#) 了。

2005/07/20: 呼呼!好不容易啊~在被台风尾扫到的七月份,终于写完这个咚咚~

2005/07/21: 在 find 部分,多增加了范例九,以及关于利用档案大小 (size) 搜寻的功能。

2005/07/25: 在 SUID/SGID/SBIT 部分,依据 netman 与小州 兄的建议,修改了部分的叙述!

2006/04/09: 在 rmdir 的范例内,少了一个 -p 的参数!

2006/06/15: 经由讨论区网友 dm421 的通知, 发现 chattr 的部分关于 d 写错了, 已订正。

2006/08/22: 增加 rm 的一些简单的说明! 尤其是『rm ./-aaa-』的删除方法!

2008/09/23: 将针对 FC4 版写的资料移到[此处](#)

2008/09/29: 加入[权限与指令的关系](#)一节, 并新增[情境模拟](#)题目喔! 大家帮忙纠错一下!

2002/03/13 以来统计人数

886219



本网页主要以 [firefox](#) 配合分辨率 1024x768 作为设计依据

<http://linux.vbird.org> is designed by [VBird](#) during 2001-2009. [Aerosol Lab](#).