LINUX 檔案與目錄管理

陳建良

內容

- 目錄與路徑
- 目錄與檔案管理
- 檔案內容查閱
- 檔案與目錄的預設權限與隱藏權限
- 指令與檔案的搜尋
- 極重要！權限與指令間的關係

目錄與路徑

- 相對路徑與絕對路徑
- 絕對路徑：路徑的寫法『一定由根目錄 / 寫起』，例如：/usr/share/doc 這個目錄。
- 相對路徑：路徑的寫法『不是由 / 寫起』，例如由 /usr/share/doc 要到 /usr/share/man 底下時，可以寫成：『cd ../man』這就是相對路徑的寫法啦！相對路徑意指『相對於目前工作目錄的路徑！』

目錄的相關操作

- cd：變換目錄
- pwd：顯示目前的目錄
- mkdir：建立一個新的目錄
- rmdir：刪除一個空的目錄

```
.           代表此層目錄
..          代表上一層目錄
-           代表前一個工作目錄
~           代表『目前使用者身份』所在的家目錄
~account    代表 account 這個使用者的家目錄(account是個帳號名稱)
```

關於執行檔路徑的變數： $PATH

- 不同身份使用者預設的PATH不同，預設能夠隨意執行的指令也不同(如root與vbird)；
- PATH是可以修改的，所以一般使用者還是可以透過修改PATH來執行某些位於/sbin或/usr/sbin下的指令來查詢；
- 使用絕對路徑或相對路徑直接指定某個指令的檔名來執行，會比搜尋PATH來的正確；
- 指令應該要放置到正確的目錄下，執行才會比較方便；
- 本目錄(.)最好不要放到PATH當中。

- 範例：先用root的身份列出搜尋的路徑為何？

  [root@www ~]# **echo $PATH**

  /usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin <==這是同一行！

- 範例：用vbird的身份列出搜尋的路徑為何？

  [root@www ~]# **su - vbird**

  [vbird@www ~]# **echo $PATH**

  /usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/home/vbird/bin

  # 仔細看，一般用戶vbird的PATH中，並不包含任何『sbin』的目錄存在喔！

- 例題：請問你能不能使用一般身份使用者下達ifconfig eth0這個指令呢？
- 答：如上面的範例所示，當你使用vbird這個帳號執行ifconfig時，會出現『-bash: ifconfig: command not found』的字樣， 因為ifconfig的是放置到/sbin底下，而由上表的結果中我們可以發現vbird的PATH並沒有設置/sbin， 所以預設無法執行。
  但是你可以使用『/sbin/ifconfig eth0』來執行這個指令喔！因為一般用戶還是可以使用ifconfig來查詢系統IP的參數，既然PATH沒有規範到/sbin，那麼我們使用『絕對路徑』也可以執行到該指令的！

- 例題：假設你是root，如果你將ls由/bin/ls移動成為/root/ls(可用『mv /bin/ls /root』指令達成)，然後你自己本身也在/root目錄下， 請問(1)你能不能直接輸入ls來執行？(2)若不能，你該如何執行ls這個指令？(3)若要直接輸入ls即可執行，又該如何進行？

1. [root@www ~]# **mv /bin/ls /root** # mv 為移動，可將檔案在不同的目錄間進行移動作業
2. [root@www ~]# **/root/ls** <==直接用絕對路徑指定該檔名

   [root@www ~]# **./ls** <==因為在 /root 目錄下，就用./ls來指定
3. [root@www ~]# **PATH="$PATH":/root**

- 例題：如果我有兩個ls指令在不同的目錄中，例如/usr/local/bin/ls與/bin/ls那麼當我下達 ls 的時候，哪個ls會被執行？
- 答：找出 PATH 裡面哪個目錄先被查詢，則那個目錄下的指令就會被先執行了！

- 例題：為什麼PATH搜尋的目錄不加入本目錄(.)？加入本目錄的搜尋不是也不錯？
- 答：如果在PATH中加入本目錄(.)後，確實我們就能夠在指令所在目錄進行指令的執行了。 但是由於你的工作目錄並非固定(常常會使用cd來切換到不同的目錄)， 因此能夠執行的指令會有變動(因為每個目錄底下的可執行檔都不相同嘛！)，這對使用者來說並非好事。
  另外，如果有個壞心使用者在/tmp底下做了一個指令，因為/tmp是大家都能夠寫入的環境，所以他當然可以這樣做。 假設該指令可能會竊取使用者的一些資料，如果你使用root的身份來執行這個指令，那不是很糟糕？ 如果這個指令的名稱又是經常會被用到的ls時，那『中標』的機率就更高了！
  所以，為了安全起見，不建議將『.』加入PATH的搜尋目錄中。

檔案與目錄的檢視： ls

```
[root@www ~]# ls [-aAdfFhilnrRSt] 目錄名稱
[root@www ~]# ls [--color={never,auto,always}] 目錄名稱
[root@www ~]# ls [--full-time] 目錄名稱
選項與參數：
-a   ：全部的檔案，連同隱藏檔( 開頭為 . 的檔案) 一起列出來(常用)
-A   ：全部的檔案，連同隱藏檔，但不包括 . 與 .. 這兩個目錄
-d   ：僅列出目錄本身，而不是列出目錄內的檔案資料(常用)
-f   ：直接列出結果，而不進行排序 (ls 預設會以檔名排序！)
-F   ：根據檔案、目錄等資訊，給予附加資料結構，例如：
       *:代表可執行檔； /:代表目錄； =:代表 socket 檔案； |:代表 FIFO 檔案；
-h   ：將檔案容量以人類較易讀的方式(例如 GB, KB 等等)列出來；
-i   ：列出 inode 號碼，inode 的意義下一章將會介紹；
-l   ：長資料串列出，包含檔案的屬性與權限等等資料；(常用)
-n   ：列出 UID 與 GID 而非使用者與群組的名稱 (UID與GID會在帳號管理提到！)
-r   ：將排序結果反向輸出，例如：原本檔名由小到大，反向則為由大到小；
-R   ：連同子目錄內容一起列出來，等於該目錄下的所有檔案都會顯示出來；
-S   ：以檔案容量大小排序，而不是用檔名排序；
-t   ：依時間排序，而不是用檔名。
--color=never   ：不要依據檔案特性給予顏色顯示；
--color=always  ：顯示顏色
--color=auto    ：讓系統自行依據設定來判斷是否給予顏色
--full-time     ：以完整時間模式 (包含年、月、日、時、分) 輸出
--time={atime,ctime} ：輸出 access 時間或改變權限屬性時間 (ctime)
                       而非內容變更時間 (modification time)
```

範例一：將家目錄下的所有檔案列出來(含屬性與隱藏檔)

```
[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
# 這個時候你會看到以 . 為開頭的幾個檔案，以及目錄檔 (.) (..) .gconf 等等，
# 不過，目錄檔檔名都是以深藍色顯示，有點不容易看清楚就是了。
```

範例二：承上題，不顯示顏色，但在檔名末顯示出該檔名代表的類型(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
```

```
# 注意看到顯示結果的第一行，嘿嘿～知道為何我們會下達類似 ./command
# 之類的指令了吧？因為 ./ 代表的是『目前目錄下』的意思啊！至於什麼是 FIFO/Socket ？
# 請參考前一章節的介紹啊！另外，那個.bashrc 時間僅寫2007，能否知道詳細時間？
範例三：完整的呈現檔案的修改時間 *(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
# 請仔細看，上面的『時間』欄位變了喔！變成較為完整的格式。
# 一般來說，ls -al 僅列出目前短格式的時間，有時不會列出年份，
# 藉由 --full-time 可以查閱到比較正確的完整時間格式啊！
```

複製、刪除與移動： cp, rm, mv

cp (複製檔案或目錄)

```
[root@www ~]# cp [-adfilprsu] 來源檔(source) 目標檔(destination)
[root@www ~]# cp [options] source1 source2 source3 .... directory
```

選項與參數：

-a ：相當於 -pdr 的意思，至於 pdr 請參考下列說明；(常用)

-d ：若來源檔為連結檔的屬性(link file)，則複製連結檔屬性而非檔案本身；

-f ：為強制(force)的意思，若目標檔案已經存在且無法開啟，則移除後再嘗試一次；

-i ：若目標檔(destination)已經存在時，在覆蓋時會先詢問動作的進行(常用)

-l ：進行硬式連結(hard link)的連結檔建立，而非複製檔案本身；

-p ：連同檔案的屬性一起複製過去，而非使用預設屬性(備份常用)；

-r ：遞迴持續複製，用於目錄的複製行為；(常用)

-s ：複製成為符號連結檔 (symbolic link)，亦即『捷徑』檔案；

-u ：若 destination 比 source 舊才更新 destination ！

最後需要注意的，如果來源檔有兩個以上，則最後一個目的檔一定要是『目錄』才行！

rm (移除檔案或目錄)

```
[root@www ~]# rm [-fir] 檔案或目錄
```

選項與參數：

-f　：就是 force 的意思，忽略不存在的檔案，不會出現警告訊息；

-i　：互動模式，在刪除前會詢問使用者是否動作

-r　：遞迴刪除啊！最常用在目錄的刪除了！這是非常危險的選項！！！

mv (移動檔案與目錄，或更名)

```
[root@www ~]# mv [-fiu] source destination
[root@www ~]# mv [options] source1 source2 source3 ....
directory
```

選項與參數：

```
-f   ：force 強制的意思，如果目標檔案已經存在，不會詢問而直接覆蓋；
-i   ：若目標檔案 (destination) 已經存在時，就會詢問是否覆蓋！
-u   ：若目標檔案已經存在，且 source 比較新，才會更新 (update)
```

mv (移動檔案與目錄，或更名)

[root@www ~]# **mv [-fiu] source destination**
[root@www ~]# **mv [options] source1 source2 source3 .... directory**
選項與參數：
-f ：force 強制的意思，如果目標檔案已經存在，不會詢問而直接覆蓋；
-i ：若目標檔案 (destination) 已經存在時，就會詢問是否覆蓋！
-u ：若目標檔案已經存在，且 source 比較新，才會更新 (update)
範例一：複製一檔案，建立一目錄，將檔案移動到目錄中
[root@www ~]# **cd /tmp**
[root@www tmp]# **cp ~/.bashrc bashrc**
[root@www tmp]# **mkdir mvtest**
[root@www tmp]# **mv bashrc mvtest**
將某個檔案移動到某個目錄去，就是這樣做！
範例二：將剛剛的目錄名稱更名為 mvtest2
[root@www tmp]# **mv mvtest mvtest2** <== 這樣就更名了！簡單～
其實在 Linux 底下還有個有趣的指令，名稱為 rename ，
該指令專職進行多個檔名的同時更名，並非針對單一檔名變更，與mv不同。請man rename。
範例三：再建立兩個檔案，再全部移動到 /tmp/mvtest2 當中
[root@www tmp]# **cp ~/.bashrc bashrc1**
[root@www tmp]# **cp ~/.bashrc bashrc2**
[root@www tmp]# **mv bashrc1 bashrc2 mvtest2**
注意到這邊，如果有多個來源檔案或目錄，則最後一個目標檔一定是『目錄！』
意思是說，將所有的資料移動到該目錄的意思！

直接檢視檔案內容

- cat 由第一行開始顯示檔案內容
- tac 從最後一行開始顯示，可以看出 tac 是 cat 的倒著寫！
- nl 顯示的時候，順道輸出行號！
- more 一頁一頁的顯示檔案內容
- less 與 more 類似，但是比 more 更好的是，它可以往前翻頁！
- head 只看頭幾行
- tail 只看尾巴幾行
- od 以二進位的方式讀取檔案內容！

可翻頁檢視

- more (一頁一頁翻動)
  - 空白鍵 (space) ：代表向下翻一頁；
  - Enter ：代表向下翻『一行』；
  - /字串 ：代表在這個顯示的內容當中，向下搜尋『字串』這個關鍵字；
  - :f ：立刻顯示出檔名以及目前顯示的行數；
  - q ：代表立刻離開 more ，不再顯示該檔案內容。
  - b 或 [ctrl]-b ：代表往回翻頁，不過這動作只對檔案有用，對管線無用。

less (一頁一頁翻動)

- 空白鍵　　：向下翻動一頁；
- [pagedown]　：向下翻動一頁；
- [pageup]　　：向上翻動一頁；
- /字串　　：向下搜尋『字串』的功能；
- ?字串　　：向上搜尋『字串』的功能；
- n　　：重複前一個搜尋 (與 / 或 ? 有關！)
- N　　：反向的重複前一個搜尋 (與 / 或 ? 有關！)
- q　　：離開 less 這個程式；

資料擷取 -- head (取出前面幾行)

```
[root@www ~]# head [-n number] 檔案
選項與參數：
-n   ：後面接數字，代表顯示幾行的意思
[root@www ~]# head /etc/man.config
# 預設的情況中，顯示前面十行！若要顯示前 20 行，就得要這樣：
[root@www ~]# head -n 20 /etc/man.config
```

tail (取出後面幾行)

[root@www ~]# **tail [-n number]** 檔案
選項與參數：
-n ：後面接數字，代表顯示幾行的意思
-f ：表示持續偵測後面所接的檔名，要等到按下[ctrl]-c才會結束tail的偵測

[root@www ~]# **tail /etc/man.config**
預設的情況中，顯示最後的十行！若要顯示最後的 20 行，就得要這樣：
[root@www ~]# **tail -n 20 /etc/man.config**

- 例題：假如我想要顯示 /etc/man.config 的第 11 到第 20 行呢？
- 答：這個應該不算難，想一想，在第 11 到第 20 行，那麼我取前 20 行，再取後十行，所以結果就是：『 head -n 20 /etc/man.config | tail -n 10 』，這樣就可以得到第 11 到第 20 行之間的內容了！

od (查閱非純文字檔)

[[root@www ~]# **od [-t TYPE]** 檔案
選項或參數：
-t ：後面可以接各種『類型 (TYPE)』的輸出，例如：
　　a　　：利用預設的字元來輸出；
　　c　　：使用 ASCII 字元來輸出
　　d[size] ：利用十進位(decimal)來輸出資料，每個整數佔用 size bytes ；
　　f[size] ：利用浮點數值(floating)來輸出資料，每個數佔用 size bytes ；
　　o[size] ：利用八進位(octal)來輸出資料，每個整數佔用 size bytes ；
　　x[size] ：利用十六進位(hexadecimal)來輸出資料，每個整數佔用 size bytes ；

touch (修改檔案時間或建置新檔)

- modification time (mtime)：
- 當該檔案的『內容資料』變更時，就會更新這個時間！內容資料指的是檔案的內容，而不是檔案的屬性或權限喔！
- status time (ctime)：
- 當該檔案的『狀態 (status)』改變時，就會更新這個時間，舉例來說，像是權限與屬性被更改了，都會更新這個時間啊。
- access time (atime)：
- 當『該檔案的內容被取用』時，就會更新這個讀取時間 (access)。舉例來說，我們使用 cat 去讀取 /etc/man.config ，就會更新該檔案的 atime 了。

```
[root@www ~]# touch [-acdmt] 檔案
選項與參數：
-a   ：僅修訂 access time；
-c   ：僅修改檔案的時間，若該檔案不存在則不建立新檔案；
-d   ：後面可以接欲修訂的日期而不用目前的日期，也可以使用 --date="日期或時間"
-m   ：僅修改 mtime ；
-t   ：後面可以接欲修訂的時間而不用目前的時間，格式為[YYMMDDhhmm]

範例一：新建一個空的檔案並觀察時間
[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，這個檔案的大小是 0 呢！在預設的狀態下，如果 touch 後面有接檔案，
# 則該檔案的三個時間 (atime/ctime/mtime) 都會更新為目前的時間。若該檔案不存在，
# 則會主動的建立一個新的空的檔案喔！例如上面這個例子！

範例二：將 ~/.bashrc 複製成為 bashrc，假設複製完全的屬性，檢查其日期
[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  <==這是 mtime
-rw-r--r-- 1 root root 176 Sep 25 21:11 bashrc  <==這是 atime
-rw-r--r-- 1 root root 176 Sep 25 21:12 bashrc  <==這是 ctime
```

檔案與目錄的預設權限與隱藏權限

- 例題：你的系統有個一般身份使用者 dmtsai，他的群組屬於 users，他的家目錄在 /home/dmtsai， 你是root，你想將你的 ~/.bashrc 複製給他，可以怎麼作？
- 答：由上一章的權限概念我們可以知道 root 雖然可以將這個檔案複製給 dmtsai，不過這個檔案在 dmtsai 的家目錄中卻可能讓 dmtsai 沒有辦法讀寫(因為該檔案屬於 root 的嘛！而 dmtsai 又不能使用 chown 之故)。 此外，我們又擔心覆蓋掉 dmtsai 自己的 .bashrc 設定檔，因此，我們可以進行如下的動作喔：
  複製檔案： cp ~/.bashrc ~dmtsai/bashrc
  修改屬性： chown dmtsai:users ~dmtsai/bashrc

- 例題：我想在 /tmp 底下建立一個目錄，這個目錄名稱為 chapter7_1 ，並且這個目錄擁有者為 dmtsai， 群組為 users ，此外，任何人都可以進入該目錄瀏覽檔案，不過除了 dmtsai 之外，其他人都不能修改該目錄下的檔案。
- 答：因為除了 dmtsai 之外，其他人不能修改該目錄下的檔案，所以整個目錄的權限應該是 drwxr-xr-x 才對！ 因此你應該這樣做：
  建立目錄： mkdir /tmp/chapter7_1
  修改屬性： chown -R dmtsai:users /tmp/chapter7_1
  修改權限： chmod -R 755 /tmp/chapter7_1

檔案預設權限：umask

- umask 就是指定『目前使用者在建立檔案或目錄時候的權限預設值』，那麼如何得知或設定 umask 呢？他的指定條件以底下的方式來指定：
  - [root@www ~]# umask
    0022 <==與一般權限有關的是後面三個數字！
    [root@www ~]# umask -S
    u=rwx,g=rx,o=rx
- 若使用者建立為『檔案』則預設『沒有可執行(x)權限』，亦即只有 rw 這兩個項目，也就是最大為 666 分，預設權限如下：

  -rw-rw-rw-
- 若使用者建立為『目錄』，則由於 x 與是否可以進入此目錄有關，因此預設為所有權限均開放，亦即為 777 分，預設權限如下：

  drwxrwxrwx

- umask 的分數指的是『該預設值需要減掉的權限！』因為 r、w、x 分別是 4、2、1 分，所以囉！也就是說，當要拿掉能寫的權限，就是輸入 2 分，而如果要拿掉能讀的權限，也就是 4 分，那麼要拿掉讀與寫的權限，也就是 6 分，而要拿掉執行與寫入的權限，也就是 3 分

- 想像一個狀況，如果你跟你的同學在同一部主機裡面工作時，因為你們兩個正在進行同一個專題， 老師也幫你們兩個的帳號建立好了相同群組的狀態，並且將 /home/class/ 目錄做為你們兩個人的專題目錄。 想像一下，有沒有可能你所製作的檔案你的同學無法編輯？果真如此的話，那就傷腦筋了！
- 這個問題很常發生啊！舉上面的案例來看就好了，你看一下 test1 的權限是幾分？ 644 呢！意思是『如果 umask 訂定為 022 ，那新建的資料只有使用者自己具有 w 的權限， 同群組的人只有 r 這個可讀的權限而已，並無法修改喔！』這樣要怎麼共同製作專題啊！

- 所以，當我們需要新建檔案給同群組的使用者共同編輯時，那麼 umask 的群組就不能拿掉 2 這個 w 的權限！所以囉， umask 就得要是 002 之類的才可以！這樣新建的檔案才能夠是 -rw-rw-r-- 的權限模樣喔！ 那麼如何設定 umask 呢？簡單的很，直接在 umask 後面輸入 002 就好了！

```
[root@study ~]# umask 002
[root@study ~]# touch test3
[root@study ~]# mkdir test4
[root@study ~]# ll -d test[34]   # 中括號 [ ] 代表中間有個指定的字元，而不是任意字元的意思
-rw-rw-r--. 1 root root 0  6月 16 01:12 test3
drwxrwxr-x. 2 root root 6  6月 16 01:12 test4
```

- 例題：假設你的 umask 為 003 ，請問該 umask 情況下，建立的檔案與目錄權限為？
- 答：

  umask 為 003 ，所以拿掉的權限為 --------wx，因此：

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

  目錄： (drwxrwxrwx) - (d-------wx) = drwxrwxr--

檔案隱藏屬性 -- chattr (設定檔案隱藏屬性)

```
[root@www ~]# chattr [+-=][ASacdistu] 檔案或目錄名稱
```

選項與參數：

+ ：增加某一個特殊參數，其他原本存在參數則不動。

- ：移除某一個特殊參數，其他原本存在參數則不動。

= ：設定一定，且僅有後面接的參數

A ：當設定了 A 這個屬性時，若你有存取此檔案(或目錄)時，他的存取時間 atime將不會被修改，可避免I/O較慢的機器過度的存取磁碟。這對速度較慢的電腦有幫助

S ：一般檔案是非同步寫入磁碟的(原理請參考第五章sync的說明)，如果加上 S 這個屬性時，當你進行任何檔案的修改，該更動會『同步』寫入磁碟中。

a ：當設定 a 之後，這個檔案將只能增加資料，而不能刪除也不能修改資料，只有root 才能設定這個屬性。

c ：這個屬性設定之後，將會自動的將此檔案『壓縮』，在讀取的時候將會自動解壓縮，但是在儲存的時候，將會先進行壓縮後再儲存(看來對於大檔案似乎蠻有用的！)

d　：當 dump 程序被執行的時候，設定 d 屬性將可使該檔案(或目錄)不會被 dump 備份

i　：這個 i 可就很厲害了！它可以讓一個檔案『不能被刪除、改名、設定連結也無法寫入或新增資料！』對於系統安全性有相當大的助益！只有 root 能設定此屬性

s　：當檔案設定了 s 屬性時，如果這個檔案被刪除，它將會被完全的移除出這個硬碟空間，所以如果誤刪了，完全無法救回來了喔！

u　：與 s 相反的，當使用 u 來設定檔案時，如果該檔案被刪除了，則資料內容其實還存在磁碟中，可以使用來救援該檔案喔！

注意：屬性設定常見的是 a 與 i 的設定值，而且很多設定值必須要身為 root 才能設定

```
範例：請嘗試到/tmp底下建立檔案，並加入 i 的參數，嘗試刪除看看。
[root@study ~]# cd /tmp
[root@study tmp]# touch attrtest      <==建立一個空檔案
[root@study tmp]# chattr +i attrtest <==給予 i 的屬性
[root@study tmp]# rm attrtest         <==嘗試刪除看看
rm: remove regular empty file `attrtest'? y
rm: cannot remove `attrtest': Operation not permitted
# 看到了嗎？呼呼！連 root 也沒有辦法將這個檔案刪除呢！趕緊解除設定！

範例：請將該檔案的 i 屬性取消！
[root@study tmp]# chattr -i attrtest
```

lsattr (顯示檔案隱藏屬性)

```
[root@www ~]# lsattr [-adR] 檔案或目錄
選項與參數：
-a ：將隱藏檔的屬性也秀出來；
-d ：如果接的是目錄，僅列出目錄本身的屬性而非目錄內的檔名；
-R ：連同子目錄的資料也一併列出來！
[root@www tmp]# chattr +aij attrtest
[root@www tmp]# lsattr attrtest
----ia---j--- attrtest
```

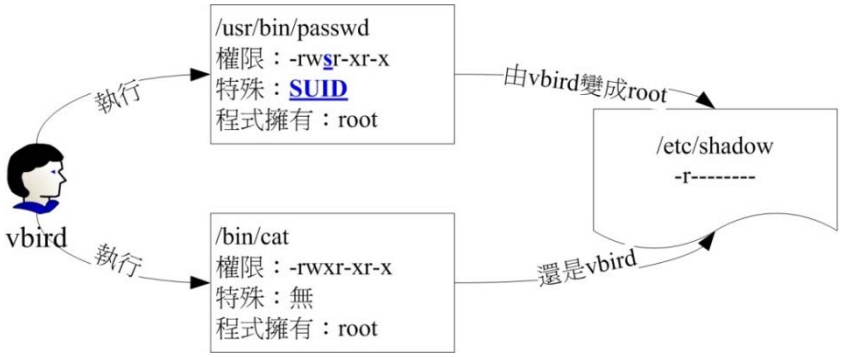
檔案特殊權限：SUID, SGID, SBIT

```
[root@study ~]# ls -ld /tmp ; ls -l /usr/bin/passwd
drwxrwxrwt. 14 root root 4096 Jun 16 01:27 /tmp
-rwsr-xr-x. 1 root root 27832 Jun 10  2014 /usr/bin/passwd
```

- Set UID
- 當 s 這個標誌出現在檔案擁有者的 x 權限上時，例如剛剛提到的 /usr/bin/passwd 這個檔案的權限狀態：『-rwsr-xr-x』，此時就被稱為 Set UID，簡稱為 SUID 的特殊權限。 那麼SUID的權限對於一個檔案的特殊功能是什麼呢？基本上SUID有這樣的限制與功能：
  - SUID 權限僅對二進位程式(binary program)有效；
  - 執行者對於該程式需要具有 x 的可執行權限；
  - 本權限僅在執行該程式的過程中有效 (run-time)；
  - 執行者將具有該程式擁有者 (owner) 的權限。

- 舉個例子來說明好了。 我們的 Linux 系統中，所有帳號的密碼都記錄在 /etc/shadow 這個檔案裡面，這個檔案的權限為：『---------- 1 root root』，意思是這個檔案僅有root可讀且僅有root可以強制寫入而已。 既然這個檔案僅有 root 可以修改，那麼鳥哥的 dmtsai 這個一般帳號使用者能否自行修改自己的密碼呢？ 你可以使用你自己的帳號輸入『passwd』這個指令來看看，嘿嘿！一般使用者當然可以修改自己的密碼了！
- 明明 /etc/shadow 就不能讓 dmtsai 這個一般帳戶去存取的，為什麼 dmtsai 還能夠修改這個檔案內的密碼呢？ 這就是 SUID 的功能啦！藉由上述的功能說明，我們可以知道
  - dmtsai 對於 /usr/bin/passwd 這個程式來說是具有 x 權限的，表示 dmtsai 能執行 passwd；
  - passwd 的擁有者是 root 這個帳號；
  - dmtsai 執行 passwd 的過程中，會『暫時』獲得 root 的權限；
  - /etc/shadow 就可以被 dmtsai 所執行的 passwd 所修改。

- 但如果 dmtsai 使用 cat 去讀取 /etc/shadow 時，他能夠讀取嗎？因為 cat 不具有 SUID 的權限，所以 dmtsai 執行 『cat /etc/shadow』 時，是不能讀取 /etc/shadow 的。我們用一張示意圖來說明如下：



SUID程式執行的過程示意圖

- SUID 僅可用在binary program 上， 不能夠用在 shell script 上面！這是因為 shell script 只是將很多的 binary 執行檔叫進來執行而已！所以 SUID 的權限部分，還是得要看 shell script 呼叫進來的程式的設定， 而不是 shell script 本身。當然，SUID 對於目錄也是無效的～這點要特別留意。

Set GID

- 當 s 標誌在檔案擁有者的 x 項目為 SUID，那 s 在群組的 x 時則稱為 Set GID, SGID 囉！
- 與 SUID 不同的是，SGID 可以針對檔案或目錄來設定！如果是對檔案來說， SGID 有如下的功能：
  - SGID 對二進位程式有用；
  - 程式執行者對於該程式來說，需具備 x 的權限
  - 執行者在執行的過程中將會獲得該程式群組的支援

```
[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
```

- 使用 vbird 這個帳號去執行 locate 時，那 vbird 將會取得 slocate 群組的支援， 因此就能夠去讀取 mlocate.db
- 除了 binary program 之外，事實上 SGID 也能夠用在目錄上，這也是非常常見的一種用途！ 當一個目錄設定了 SGID 的權限後，他將具有如下的功能：
  - 使用者若對於此目錄具有 r 與 x 的權限時，該使用者能夠進入此目錄；
  - 使用者在此目錄下的有效群組(effective group)將會變成該目錄的群組；
  - 用途：若使用者在此目錄下具有 w 的權限(可以新建檔案)，則使用者所建立的新檔案，該新檔案的群組與此目錄的群組相同。

- 假設系統中有兩個帳號，分別是 alex 與 arod ，這兩個人除了自己群組之外還共同支援一個名為 project 的群組。假設這兩個用戶需要共同擁有 /srv/ahome/ 目錄的開發權，且該目錄不許其他人進入查閱。 請問該目錄的權限設定應為何？
  - 目標：瞭解到為何專案開發時，目錄最好需要設定 SGID 的權限！
  - 前提：多個帳號支援同一群組，且共同擁有目錄的使用權！
  - 需求：需要使用 root 的身份來進行 chmod, chgrp 等幫用戶設定好他們的開發環境才行！ 這也是管理員的重要任務之一！

- 首先我們得要先製作出這兩個帳號的相關資料

```
[root@study ~]# groupadd project          <==增加新的群組
[root@study ~]# useradd -G project alex <==建立 alex 帳號，且支援 project
[root@study ~]# useradd -G project arod <==建立 arod 帳號，且支援 project
[root@study ~]# id alex                   <==查閱 alex 帳號的屬性
uid=1001(alex) gid=1002(alex) groups=1002(alex),1001(project) <==確實有支援！
[root@study ~]# id arod
uid=1002(arod) gid=1003(arod) groups=1003(arod),1001(project) <==確實有支援！
```

- 建立所需要開發的專案目錄：

```
[root@study ~]# mkdir /srv/ahome
[root@study ~]# ll -d /srv/ahome
drwxr-xr-x. 2 root root 6 Jun 17 00:22 /srv/ahome
```

- 從上面的輸出結果可發現 alex 與 arod 都不能在該目錄內建立檔案，因此需要進行權限與屬性的修改。 由於其他人均不可進入此目錄，因此該目錄的群組應為project，權限應為770才合理。

```
[root@study ~]# chgrp project /srv/ahome
[root@study ~]# chmod 770 /srv/ahome
[root@study ~]# ll -d /srv/ahome
drwxrwx---. 2 root project 6 Jun 17 00:22 /srv/ahome
# 從上面的權限結果來看，由於 alex/arod 均支援 project，因此似乎沒問題了！
```

- 實際分別以兩個使用者來測試看看，情況會是如何？先用 alex 建立檔案，然後用 arod 去處理看看。

```
[root@study ~]# su - alex        <==先切換身份成為 alex 來處理
[alex@www ~]$ cd /srv/ahome    <==切換到群組的工作目錄去
[alex@www ahome]$ touch abcd   <==建立一個空的檔案出來！
[alex@www ahome]$ exit         <==離開 alex 的身份

[root@study ~]# su - arod
[arod@www ~]$ cd /srv/ahome
[arod@www ahome]$ ll abcd
-rw-rw-r--. 1 alex alex 0 Jun 17 00:23 abcd
# 仔細看一下上面的檔案，由於群組是 alex ，arod並不支援！
# 因此對於 abcd 這個檔案來說， arod 應該只是其他人，只有 r 的權限而已啊！
[arod@www ahome]$ exit
```

- 由上面的結果我們可以知道，若單純使用傳統的 rwx 而已，則對剛剛 alex 建立的 abcd 這個檔案來說， arod 可以讀取他，但是卻不能編輯！
- 加入 SGID 的權限在裡面，並進行測試看看：

```
[root@study ~]# chmod 2770 /srv/ahome
[root@study ~]# ll -d /srv/ahome
drwxrws---. 2 root project 17 Jun 17 00:23 /srv/ahome

測試：使用 alex 去建立一個檔案，並且查閱檔案權限看看：
[root@study ~]# su - alex
[alex@www ~]$ cd /srv/ahome
[alex@www ahome]$ touch 1234
[alex@www ahome]$ ll 1234
-rw-rw-r--. 1 alex project 0 Jun 17 00:25 1234
# 沒錯！這才是我們要的樣子！現在 alex, arod 建立的新檔案所屬群組都是 project，
# 由於兩人均屬於此群組，加上 umask 都是 002，這樣兩人才可以互相修改對方的檔案！
```

- 所以最終的結果顯示，此目錄的權限最好是『2770』，所屬檔案擁有者屬於root即可，至於群組必須要為兩人共同支援的project 這個群組才行！

Sticky Bit

- SBIT 目前只針對目錄有效，對於檔案已經沒有效果了。SBIT 對於目錄的作用是：
  - 當使用者對於此目錄具有 w, x 權限，亦即具有寫入的權限時；
  - 當使用者在該目錄下建立檔案或目錄時，僅有自己與 root 才有權力刪除該檔案
- 當甲這個使用者於 A 目錄是具有群組或其他人的身份，並且擁有該目錄 w 的權限， 這表示『甲使用者對該目錄內任何人建立的目錄或檔案均可進行 "刪除/更名/搬移" 等動作。』 不過，如果將 A 目錄加上了 SBIT 的權限項目時， 則甲只能夠針對自己建立的檔案或目錄進行刪除/更名/移動等動作，而無法刪除他人的檔案。

- /tmp 本身的權限是『drwxrwxrwt』，在這樣的權限內容下，任何人都可以在 /tmp 內新增、修改檔案，但僅有該檔案/目錄建立者與 root 能夠刪除自己的目錄或檔案。可以這樣做個簡單的測試：
  1. 以 root 登入系統，並且進入 /tmp 當中；
  2. touch test，並且更改 test 權限成為 777 ；
  3. 以一般使用者登入，並進入 /tmp；
  4. 嘗試刪除 test 這個檔案！

SUID/SGID/SBIT 權限設定

- 4 為 SUID
- 2 為 SGID
- 1 為 SBIT

```
[root@study ~]# cd /tmp
[root@study tmp]# touch test                    <==建立一個測試用空檔
[root@study tmp]# chmod 4755 test; ls -l test <==加入具有 SUID 的權限
-rwsr-xr-x 1 root root 0 Jun 16 02:53 test
[root@study tmp]# chmod 6755 test; ls -l test <==加入具有 SUID/SGID 的權限
-rwsr-sr-x 1 root root 0 Jun 16 02:53 test
[root@study tmp]# chmod 1755 test; ls -l test <==加入 SBIT 的功能！
-rwxr-xr-t 1 root root 0 Jun 16 02:53 test
[root@study tmp]# chmod 7666 test; ls -l test <==具有空的 SUID/SGID 權限
-rwSrwSrwT 1 root root 0 Jun 16 02:53 test
```

- 怎麼會出現大寫的 S 與 T 呢？不都是小寫的嗎？ 因為 s 與 t 都是取代 x 這個權限的，但是你有沒有發現阿，我們是下達 7666 喔！也就是說， user, group 以及 others 都沒有 x 這個可執行的標誌(因為 666 嘛)，所以，這個 S,T 代表的就是『空的』啦！怎麼說？ SUID 是表示『該檔案在執行的時候，具有檔案擁有者的權限』，但是檔案 擁有者都無法執行了，哪裡來的權限給其他人使用？當然就是空的啦！

觀察檔案類型：file

- 如果你想要知道某個檔案的基本資料，例如是屬於 ASCII 或者是 data 檔案，或者是 binary ， 且其中有沒有使用到動態函式庫 (share library) 等等的資訊，就可以利用 file 這個指令來檢閱

```
[root@study ~]# file ~/.bashrc
/root/.bashrc: ASCII text  <==告訴我們是 ASCII 的純文字檔啊！
[root@study ~]# file /usr/bin/passwd
/usr/bin/passwd: setuid ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically
linked (uses shared libs), for GNU/Linux 2.6.32,
BuildID[sha1]=0xbf35571e607e317bf107b9bcf65199988d0ed5ab, stripped
# 執行檔的資料可就多的不得了！包括這個檔案的 suid 權限、相容於 Intel x86-64 等級的硬體平台
# 使用的是 Linux 核心 2.6.32 的動態函式庫連結等等。
[root@study ~]# file /var/lib/mlocate/mlocate.db
/var/lib/mlocate/mlocate.db: data  <== 這是 data 檔案！
```

指令檔名的搜尋 -- which (尋找『執行檔』)

- 舉例來說，ls 這個常用的指令放在哪裡呢？ 就透過 which 來找尋吧！

```
[root@study ~]# which [-a] command
選項或參數：
-a ：將所有由 PATH 目錄中可以找到的指令均列出，而不止第一個被找到的指令名稱

範例一：搜尋 ifconfig 這個指令的完整檔名
[root@study ~]# which ifconfig
/sbin/ifconfig

範例二：用 which 去找出 which 的檔名為何？
[root@study ~]# which which
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'
        /bin/alias
        /usr/bin/which
# 竟然會有兩個 which ，其中一個是 alias 這玩意兒呢！那是啥？
# 那就是所謂的『命令別名』，意思是輸入 which 會等於後面接的那串指令啦！
# 更多的資料我們會在 bash 章節中再來談的！

範例三：請找出 history 這個指令的完整檔名
[root@study ~]# which history
/usr/bin/which: no history in (/usr/local/sbin:/usr/local/bin:/sbin:/bin:
/usr/sbin:/usr/bin:/root/bin)

[root@study ~]# history --help
-bash: history: --: invalid option
history: usage: history [-c] [-d offset] [n] or history -anrw [filename] or history -ps arg
# 瞎密？怎麼可能沒有 history ，我明明就能夠用 root 執行 history 的啊！
```

檔案檔名的搜尋 – whereis , locate

- whereis 只找系統中某些特定目錄底下的檔案而已，locate 則是利用資料庫來搜尋檔名

```
[root@study ~]# whereis [-bmsu] 檔案或目錄名
選項與參數：
-l    :可以列出 whereis 會去查詢的幾個主要目錄而已
-b    :只找 binary 格式的檔案
-m    :只找在說明檔 manual 路徑下的檔案
-s    :只找 source 來源檔案
-u    :搜尋不在上述三個項目當中的其他特殊檔案

範例一：請找出 ifconfig 這個檔名
[root@study ~]# whereis ifconfig
ifconfig: /sbin/ifconfig /usr/share/man/man8/ifconfig.8.gz

範例二：只找出跟 passwd 有關的『說明文件』檔名(man page)
[root@study ~]# whereis passwd     # 全部的檔名通通列出來！
passwd: /usr/bin/passwd /etc/passwd /usr/share/man/man1/passwd.1.gz /usr/share/man/man5/passwd.5.gz
[root@study ~]# whereis -m passwd  # 只有在 man 裡面的檔名才抓出來！
passwd: /usr/share/man/man1/passwd.1.gz /usr/share/man/man5/passwd.5.gz
```

locate

- locate：依據 /var/lib/mlocate 內的資料庫記載，找出使用者輸入的關鍵字檔名。
- updatedb：根據 /etc/updatedb.conf 的設定去搜尋系統硬碟內的檔名，並更新 /var/lib/mlocate 內的資料庫檔案；

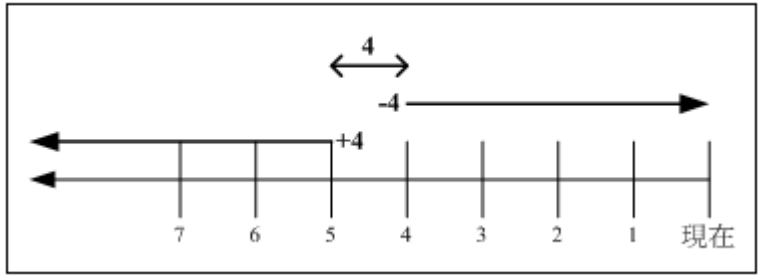
```
[root@study ~]# locate [-ir] keyword
選項與參數：
-i   ：忽略大小寫的差異；
-c   ：不輸出檔名，僅計算找到的檔案數量
-l   ：僅輸出幾行的意思，例如輸出五行則是 -l 5
-S   ：輸出 locate 所使用的資料庫檔案的相關資訊，包括該資料庫紀錄的檔案/目錄數量等
-r   ：後面可接正規表示法的顯示方式

範例一：找出系統中所有與 passwd 相關的檔名，且只列出 5 個
[root@study ~]# locate -l 5 passwd
/etc/passwd
/etc/passwd-
/etc/pam.d/passwd
/etc/security/opasswd
/usr/bin/gpasswd

範例二：列出 locate 查詢所使用的資料庫檔案之檔名與各資料數量
[root@study ~]# locate -S
Database /var/lib/mlocate/mlocate.db:
        8,086 directories      # 總紀錄目錄數
        109,605 files          # 總紀錄檔案數
        5,190,295 bytes in file names
        2,349,150 bytes used to store database
```

find

```
[root@study ~]# find [PATH] [option] [action]
選項與參數：
1. 與時間有關的選項：共有 -atime, -ctime 與 -mtime ，以 -mtime 說明
   -mtime  n ：n 為數字，意義為在 n 天之前的『一天之內』被更動過內容的檔案；
   -mtime +n ：列出在 n 天之前(不含 n 天本身)被更動過內容的檔案檔名；
   -mtime -n ：列出在 n 天之內(含 n 天本身)被更動過內容的檔案檔名。
   -newer file ：file 為一個存在的檔案，列出比 file 還要新的檔案檔名

範例一：將過去系統上面 24 小時內有更動過內容 (mtime) 的檔案列出
[root@study ~]# find / -mtime 0
# 那個 0 是重點！0 代表目前的時間，所以，從現在開始到 24 小時前，
# 有變動過內容的檔案都會被列出來！那如果是三天前的 24 小時內？
# find / -mtime 3 有變動過的檔案都被列出的意思！

範例二：尋找 /etc 底下的檔案，如果檔案日期比 /etc/passwd 新就列出
[root@study ~]# find /etc -newer /etc/passwd
# -newer 用在分辨兩個檔案之間的新舊關係是很有用的！
```

- 如果你想要找出一天內被更動過的檔案名稱， 可以使用上述範例一的作法。

- 如果我想要找出『4天內被更動過的檔案檔名』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 』。有沒有加上『+, -』差別很大喔！我們可以用簡單的圖示來說明一下：



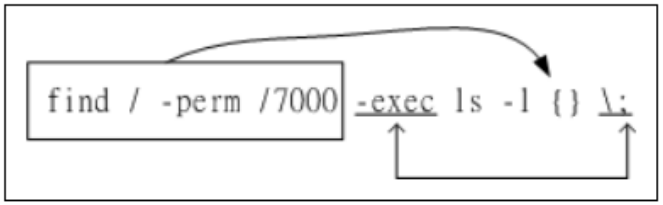
- 圖中最右邊為目前的時間，越往左邊則代表越早之前的時間軸啦。我們可以清楚的知道：
  - +4代表大於等於5天前的檔名：　　ex> find /var -mtime +4
  - -4代表小於等於4天內的檔案檔名：　ex> find /var -mtime -4
  - 4則是代表4-5那一天的檔案檔名：　　ex> find /var -mtime 4

```
選項與參數：
2. 與使用者或群組名稱有關的參數：
   -uid n  ：n 為數字，這個數字是使用者的帳號 ID，亦即 UID ，這個 UID 是記錄在
           /etc/passwd 裡面與帳號名稱對應的數字。這方面我們會在第四篇介紹。
   -gid n  ：n 為數字，這個數字是群組名稱的 ID，亦即 GID，這個 GID 記錄在
           /etc/group，相關的介紹我們會第四篇說明～
   -user name  ：name 為使用者帳號名稱喔！例如 dmtsai
   -group name：name 為群組名稱喔，例如 users ；
   -nouser     ：尋找檔案的擁有者不存在 /etc/passwd 的人！
   -nogroup    ：尋找檔案的擁有群組不存在於 /etc/group 的檔案！
                 當你自行安裝軟體時，很可能該軟體的屬性當中並沒有檔案擁有者，
                 這是可能的！在這個時候，就可以使用 -nouser 與 -nogroup 搜尋。

範例三：搜尋 /home 底下屬於 dmtsai 的檔案
[root@study ~]# find /home -user dmtsai
# 這個東西也很有用的～當我們要找出任何一個使用者在系統當中的所有檔案時，
# 就可以利用這個指令將屬於某個使用者的所有檔案都找出來喔！

範例四：搜尋系統中不屬於任何人的檔案
[root@study ~]# find / -nouser
# 透過這個指令，可以輕易的就找出那些不太正常的檔案。如果有找到不屬於系統任何人的檔案時，
# 不要太緊張，那有時候是正常的～尤其是你曾經以原始碼自行編譯軟體時。
```

```
選項與參數：
3. 與檔案權限及名稱有關的參數：
   -name filename：搜尋檔案名稱為 filename 的檔案；
   -size [+-]SIZE：搜尋比 SIZE 還要大(+)或小(-)的檔案。這個 SIZE 的規格有：
                   c: 代表 byte， k: 代表 1024bytes。所以，要找比 50KB
                   還要大的檔案，就是『 -size +50k 』
   -type TYPE     ：搜尋檔案的類型為 TYPE 的，類型主要有：一般正規檔案 (f), 裝置檔案 (b, c),
                   目錄 (d), 連結檔 (l), socket (s), 及 FIFO (p) 等屬性。
   -perm mode   ：搜尋檔案權限『剛好等於』 mode 的檔案，這個 mode 為類似 chmod
                  的屬性值，舉例來說， -rwsr-xr-x 的屬性為 4755 ！
   -perm -mode  ：搜尋檔案權限『必須要全部囊括 mode 的權限』的檔案，舉例來說，
                  我們要搜尋 -rwxr--r-- ，亦即 0744 的檔案，使用 -perm -0744，
                  當一個檔案的權限為 -rwsr-xr-x ，亦即 4755 時，也會被列出來，
                  因為 -rwsr-xr-x 的屬性已經囊括了 -rwxr--r-- 的屬性了。
   -perm /mode  ：搜尋檔案權限『包含任一 mode 的權限』的檔案，舉例來說，我們搜尋
                  -rwxr-xr-x ，亦即 -perm /755 時，但一個檔案屬性為 -rw-------
                  也會被列出來，因為他有 -rw.... 的屬性存在！

範例五：找出檔名為 passwd 這個檔案
[root@study ~]# find / -name passwd

範例五-1：找出檔名包含了 passwd 這個關鍵字的檔案
[root@study ~]# find / -name "*passwd*"
# 利用這個 -name 可以搜尋檔名啊！預設是完整檔名，如果想要找關鍵字，
# 可以使用類似 * 的任意字元來處理

範例六：找出 /run 目錄下，檔案類型為 Socket 的檔名有哪些？
[root@study ~]# find /run -type s
# 這個 -type 的屬性也很有幫助喔！尤其是要找出那些怪異的檔案，
# 例如 socket 與 FIFO 檔案，可以用 find /run -type p 或 -type s 來找！

範例七：搜尋檔案當中含有 SGID 或 SUID 或 SBIT 的屬性
[root@study ~]# find / -perm /7000
# 所謂的 7000 就是 ---s--s--t ，那麼只要含有 s 或 t 的就列出，所以當然要使用 /7000，
# 使用 -7000 表示要同時含有 ---s--s--t 的所有三個權限。而只需要任意一個，就是 /7000 ～瞭乎？
```

```
選項與參數：
4. 額外可進行的動作：
   -exec command ：command 為其他指令，-exec 後面可再接額外的指令來處理搜尋到的結果。
   -print        ：將結果列印到螢幕上，這個動作是預設動作！

範例八：將上個範例找到的檔案使用 ls -l 列出來～
[root@study ~]# find /usr/bin /usr/sbin -perm /7000 -exec ls -l {} \;
# 注意到，那個 -exec 後面的 ls -l 就是額外的指令，指令不支援命令別名，
# 所以僅能使用 ls -l 不可以使用 ll 喔！注意注意！

範例九：找出系統中，大於 1MB 的檔案
[root@study ~]# find / -size +1M
```

- find 的特殊功能就是能夠進行額外的動作(action)。我們將範例八的例子以圖解來說明如下：



- 該範例中特殊的地方有 {} 以及 \; 還有 -exec 這個關鍵字，這些東西的意義為：
  - {} 代表的是『由 find 找到的內容』，如上圖所示，find 的結果會被放置到 {} 位置中；
  - -exec 一直到 \; 是關鍵字，代表 find 額外動作的開始 (-exec) 到結束 (\;) ，在這中間的就是 find 指令內的額外動作。 在本例中就是『 ls -l {} 』囉！
  - 因為『 ; 』在 bash 環境下是有特殊意義的，因此利用反斜線來跳脫。

極重要！權限與指令間的關係

- 讓使用者能進入某目錄成為『可工作目錄』的基本權限為何：
  - 可使用的指令：例如 cd 等變換工作目錄的指令；
  - 目錄所需權限：使用者對這個目錄至少需要具有 x 的權限
  - 額外需求：如果使用者想要在這個目錄內利用 ls 查閱檔名，則使用者對此目錄還需要 r 的權限。
- 使用者在某個目錄內讀取一個檔案的基本權限為何？
  - 可使用的指令：例如本章談到的 cat, more, less等等
  - 目錄所需權限：使用者對這個目錄至少需要具有 x 權限；
  - 檔案所需權限：使用者對檔案至少需要具有 r 的權限才行！
- 讓使用者可以修改一個檔案的基本權限為何？
  - 可使用的指令：例如 nano 或未來要介紹的 vi 編輯器等；
  - 目錄所需權限：使用者在該檔案所在的目錄至少要有 x 權限；
  - 檔案所需權限：使用者對該檔案至少要有 r, w 權限

- 讓一個使用者可以建立一個檔案的基本權限為何？
  - 目錄所需權限：使用者在該目錄要具有 **w,x** 的權限，重點在 **w** 啦！
- 讓使用者進入某目錄並執行該目錄下的某個指令之基本權限為何？
  - 目錄所需權限：使用者在該目錄至少要有 **x** 的權限；
  - 檔案所需權限：使用者在該檔案至少需要有 **x** 的權限

- 讓一個使用者 dmtsai 能夠進行『cp /dir1/file1 /dir2』的指令時，請說明 dir1, file1, dir2 的最小所需權限為何？
- 執行 cp 時， dmtsai 要『能夠讀取來源檔，並且寫入目標檔！』因此各檔案/目錄的最小權限應該是：
  - dir1 ：至少需要有 x 權限；
  - file1：至少需要有 r 權限；
  - dir2 ：至少需要有 w, x 權限。

- 有一個檔案全名為 /home/student/www/index.html ，各相關檔案/目錄的權限如下：

```
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
```

- 請問 vbird 這個帳號(不屬於student群組)能否讀取 index.html 這個檔案呢？
- 雖然 www 與 index.html 是可以讓 vbird 讀取的權限，但是因為目錄結構是由根目錄一層一層讀取的， 因此 vbird 可進入 /home 但是卻不可進入 /home/student/ ，既然連進入 /home/student 都不許了， 當然就讀不到 index.html 了！所以答案是『vbird不會讀取到 index.html 的內容』喔！
- 那要如何修改權限呢？其實只要將 /home/student 的權限修改為最小 711 ，或者直接給予 755 就可以