

Linux檔案權限與基礎帳號管理

Reading Material

- Linux 基礎檔案權限與基礎帳號管理
 - http://linux.vbird.org/linux_basic_train/unit04.php
- 權限在目錄與檔案應用上的意義
 - http://linux.vbird.org/linux_basic_train/unit05.php#5.1
- 使用者與群組
 - http://linux.vbird.org/linux_basic/0210filepermission.php#UserandGroup
- Linux 檔案權限概念
 - http://linux.vbird.org/linux_basic/0210filepermission.php#filepermission

檔案權限的功用

- 權限都是設定在檔案/目錄後，應用在某個/某群帳號上面，會造成哪個個人或某群人的讀寫開放或保護。
- **Eg:** 有一份成績單的檔案，老師能編輯它。班上學生可以查它，別班的人不能看

Linux 傳統權限:使用者、群組與其他人

- Linux 檔案權限在設定上，主要依據三種身份來決定
- user / owner / 檔案擁有者 / 使用者：就是檔案所屬人
- group / 群組：這個檔案附屬於那一個群組團隊
- others / 其他人：不是 user 也沒加入 group 的帳號，就是其他人。
- Eg: 成績單是檔案，owner是老師，班上的同學屬於同一個group，別班的人是others

觀察檔案權限

```
[student@localhost ~]$ ls -ld /var/spool/mail
drwxrwxr-x. 2 root mail 4096 6月 29 03:29 /var/spool/mail
[   A   ][B][C ][D ][E ][   F   ][   G   ]
```

- A: 檔案類型與權限，第 1 個字元為檔案類型，後續 9 個字元每 3 個一組，共分 3 組，為三種身份的權限；
- B: 檔案連結數，這與檔案系統有關，讀者可暫時略過；
- C: 該檔案的擁有者，本例當中，擁有者身份為 root
- D: 該檔案的所屬群組，本例當中這個檔案屬於 mail 這個群組底下
- E: 檔案的容量
- F: 該檔案最後一次被修改/修訂的日期時間
- G: 檔名。
 - . 開頭的為隱藏檔

檔案的類型

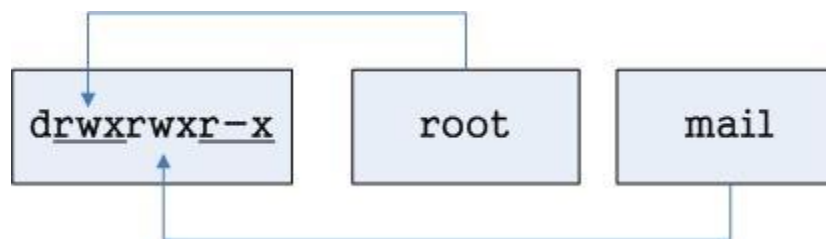
- -: 代表後面的檔名為一般檔案
- d: 代表後面的檔名為目錄檔
- l: 代表後面的檔名為連結檔 (有點類似 windows 的捷徑概念)
- b: 代表後面的檔名為一個裝置檔，該裝置主要為區塊裝置，亦即儲存媒體的裝置較多
- c: 代表後面的檔名為一個週邊裝置檔，例如滑鼠、鍵盤等

檔案權限類型

- r: read，可讀的意思
- w: write，可寫入/編輯/修改的意思
- x: eXecutable，可以執行的意思
- -: 沒有設定權限

判讀檔案權限

```
[student@localhost ~]$ ls -ld /var/spool/mail
drwxrwxr-x. 2 root mail 4096 6月 29 03:29 /var/spool/mail
[  A  ][B][C ][D ][E ][  F  ][  G  ]
```



- 擁有者為 **root**，**root** 具有 **rwx** 的權限 (第一組權限)
- 群組設定為 **mail**，則所有加入 **mail** 這個群組的帳號可以具有 **rwx** 的權限 (第二組權限)
- 不是 **root** 也沒有加入 **mail** 的其，只有 **rx** 的權限 (第三組權限)

相關指令

- 觀察帳號的指令
 - `id`

```
jimmy@cluster-f:~$ id jimmy
uid=1000(jimmy) gid=1000(jimmy) groups=1000(jimmy),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lxd),115(lpadmin),116(sambashare),999(docker)
```

- 觀察檔案的擁有者與相關的權限
 - `getfacl`

```
jimmy@cluster-f:~$ getfacl checking
# file: checking
# owner: daemon
# group: bin
user::rwx
group::r--
other::---
```

例題：

假設有帳號資料如下：

帳號名稱	加入的群組
test1	test1, testgroup
test2	test2, testgroup
test3	test3, testgroup
test4	test4

如果有下面兩個檔案，請分別說明 test1, test2, test3, test4 針對底下兩個檔案的權限各為何？

-rw-r--r--	1	root	root	238	Jun 18 17:22	test.txt
-rwxr-xr--	1	test1	testgroup	5238	Jun 19 10:25	ping_tsai

權限在目錄與一般檔的意義

- Linux 的檔案權限設定上有三種身份，每種身份則有三種權限 (rwx)
- 基本上就有一般檔案與目錄檔案，這兩種檔案類型的權限意義並不相同。

權限對檔案的重要性

- 檔案類型包括一般文字檔、資料庫內容檔、二進位可執行檔(binary program)等等
 - r (read)：可讀取此一檔案的實際內容，如讀取文字檔的文字內容等；
 - w (write)：可以編輯、新增或者是修改該檔案的內容
 - x (eXecute)：該檔案具有可以被系統執行的權限。
- **rwX**權限都是針對『檔案的內容』而言，與檔案檔名的存在與否沒有關係。
 - Linux裡的執行檔，不一定有**exe**副檔名
 - Linux裡的壓縮檔，不一定有**zip**副檔名

Linux檔案副檔名

- Linux的檔案是沒有所謂的『副檔名』
- Linux系統上的檔名只是讓你瞭解該檔案可能的用途
 - *.sh : 腳本或批次檔 (scripts)
 - *Z, *.tar, *.tar.gz, *.zip, *.tgz : 經過打包的壓縮檔
 - *.html, *.php : 網頁相關檔案
- 一個Linux檔案能不能被執行，與他的第一欄的十個屬性有關，與檔名根本一點關係也沒有

權限對目錄的重要性

- 目錄主要的功能在記錄檔名清單
- **r (read contents in directory) :**
 - 用戶具有讀取目錄權限時，表示該用戶可以查詢該目錄下的檔名資料。
- **w (modify contents of directory) :**
 - 用戶具有寫入目錄的權限時，表示用戶具有異動該目錄結構清單的權限
 - 建立新的檔案與目錄；
 - 刪除已經存在的檔案與目錄(不論該檔案的權限為何！)
 - 將已存在的檔案或目錄進行更名；
 - 搬移該目錄內的檔案、目錄位置。
- **x (access directory) :**
 - 使用者能否進入該目錄成為工作目錄的用途

例題：

例題：

有個目錄的權限如下所示：

```
drwxr--r--  3  root  root  4096   Jun 25 08:35  .ssh
```

系統有個帳號名稱為**vbird**，這個帳號並沒有支援**root**群組，請問**vbird**對這個目錄有何權限？是否可切換到此目錄中？

例題：

操作動作	/dir1	/dir1/file1	/dir2	重點
讀取 file1 內容				要能夠進入 /dir1 才能讀到裡面的文件資料！
修改 file1 內容				能夠進入 /dir1 且修改 file1 才行！
執行 file1 內容				能夠進入 /dir1 且 file1 能運作才行！
刪除 file1 檔案				能夠進入 /dir1 具有目錄修改的權限即可！
將 file1 複製到 /dir2				要能夠讀 file1 且能夠修改 /dir2 內的資料
將 file1 移動到 /dir2				兩個目錄均需要被更改

Linux 傳統權限:檔案屬性與權限的修改方式

```
[student@localhost shm]$ touch checking
[student@localhost shm]$ ls -l checking
-rw-rw-r--. 1 student student 0  6月 30 15:16 checking
[ chmod ]    [ chown ] [ chgrp ]    [ touch ] [ mv ]
```

- 一般帳號僅能修改自己檔案的檔名、時間與權限，即只能用mv、touch、chmod
- 只有root能切換使用者與群組的設定，即chown、chgrp

使用 chown 修改檔案擁有者

- 先用id判斷daemon是否存在
- 用chown修改owner

```
[root@localhost shm]# id daemon
uid=2(daemon) gid=2(daemon) groups=2(daemon)

[root@localhost shm]# chown daemon checking
[root@localhost shm]# ll checking
-rw-rw-r--. 1 daemon student 0 6月 30 15:16 checking
```

使用 chgrp 修改檔案擁有的群組

- 先查bin group是否存在
- 再用chgrp修改

```
[root@localhost shm]# grep myname /etc/group
# 不會出現任何資訊，因為沒有這個群組存在的意思。

[root@localhost shm]# grep bin /etc/group
bin:x:1:      ←代表確實有這個群組存在！

[root@localhost shm]# chgrp bin checking
[root@localhost shm]# ll checking
-rw-rw-r--. 1 daemon bin 0 6月 30 15:16 checking
```

使用 chmod 修改權限: 數字法

- 檔案紀錄了三種身份，每種身份都擁有 **rwX** 的最大權限與 --- 沒權限的情況。
- 用二進法表示
 - $r \implies \text{read} \implies 2^2 \implies 4$
 - $w \implies \text{write} \implies 2^1 \implies 2$
 - $x \implies \text{eXecute} \implies 2^0 \implies 1$
 - $- \implies \text{沒權限} \implies 0$
- $rwX \implies 4 + 2 + 1 \implies 7$
- $rw- \implies 4 + 2 + 0 \implies 6$

例題：

- 讓 daemon 可讀、可寫、可執行 checking，讓加入 bin 群組的用戶可唯讀該檔案，讓其他人沒有權限！
 - daemon 為使用者，可讀可寫可執行則為 $rw\textcolor{red}{x} = 7$
 - 加入 bin 的群組為唯讀，亦即為 $r\textcolor{red}{--} = 4$
 - 其他人沒權限，因此為 $\textcolor{red}{---} = 0$

```
[root@localhost shm]# chmod 740 checking  
[root@localhost shm]# ll checking  
-rwxr-----. 1 daemon bin 0 6月 30 15:16 checking
```

使用 chmod 修改權限:符號法

chmod	u(user) g(group) o(other) a(all)	+(加入) -(減去) =(設定)	r w x	檔案或目錄
-------	---	-------------------------	-------------	-------

例題：

- 讓 daemon 可讀、可寫、可執行 checking，讓加入 bin 群組的用戶可唯讀該檔案，讓其他人沒有權限！
 - daemon 為使用者，可讀可寫可執行則為 **u=rwx**
 - 加入 bin 的群組為唯讀，亦即為 **g=r**
 - 其他人沒權限，因此為 **o=**

```
root@cluster-f:/home/jimmy# chmod u=rwx,g=r,o= checking
root@cluster-f:/home/jimmy# ll checking
-rwxr----- 1 daemon bin 0 Oct 17 08:44 checking*
```

帳號管理: 基本帳號管理

- 登入系統的時候，需要輸入兩個資料，
 - 一個是點選帳號名稱
 - 該帳號的密碼。
- 最基本的帳號管理，即是建立帳號與給予密碼
 - `useradd (or adduser)`
 - `userdel`

- 建立一個名為 myuser1 的帳號，以及給予 MypassworD 的密碼

```
[root@localhost ~]# useradd myuser1
[root@localhost ~]# passwd myuser1
Changing password for user myuser1.
New password:      <==此處輸入密碼
BAD PASSWORD: The password fails the dictionary check - it is based on a dictionary word
Retype new password: <==再輸入密碼一次
passwd: all authentication tokens updated successfully.

[root@localhost ~]# id myuser1
uid=1014(myuser1) gid=1015(myuser1) groups=1015(myuser1)
```

- 使用 userdel 刪除帳號
 - -r 的目的是要該帳號連同家目錄通通刪除

```
[root@localhost ~]# userdel -r myuser1
```

Useradd 做的事情 (for centos)

- 在 `/etc/passwd` 裡面建立一行與帳號相關的資料
- 在 `/etc/shadow` 裡面將此帳號的密碼相關參數填入，但是尚未有密碼，所以要再用 `passwd` 設定密碼
- 在 `/etc/group` 裡面加入一個與帳號名稱一模一樣的群組名稱；
- 在 `/home` 底下建立一個與帳號同名的目錄作為使用者家目錄，且權限為 `700`

例題: for centos (ubuntu使用adduser -m)

- 建立名為 myuser2 的帳號，該帳號密碼為 mypassWORD
- 建立名為 myuser3 的帳號，該帳號密碼為 mypassWORD
- 觀察 myuser2 與 myuser3 的 id 情況
- 觀察 /home 目錄的內容，是否有名為 myuser2 及 myuser3 的檔名存在？
- 使用 userdel myuser2 刪除帳號 (注意，不要加上 -r 的參數)
- 再次觀察 /home 與 /var/spool/mail 的內容，myuser2 檔名是否存在？該檔名的權限為何？
- 重新建立名為 myuser2 的帳號，密碼亦同為 mypassWORD，嘗試討論
 - (1)建立過程中出現的問題原因為何？
 - (2)是否能夠順利建立該帳號？
- 承上，請在 tty2 以後的終端機，使用 myuser2 登入系統，登入後是否出現問題？為什麼？
- 再次使用 userdel -r 的方式刪除 myuser2 與 myuser3，是否能夠順利刪除？
- 承上，若無法順利刪除帳號，請以手動的方式自行刪除餘留的使用者家目錄與郵件檔案。

useradd v.s. adduser (for ubuntu)

- 使用useradd，如果不加任何選項，如# useradd myuser1，建出來的使用者
 - 無home目錄
 - 無登入shell
 - 無password
- 使用adduser
 - 會有對話過程，根據提供的資訊建立使用者

帳號管理:帳號與群組關聯性管理

- 建立帳號時，會同時建立一個同名的group
- 一個帳號可以加入多個group，
 - 例如合作專題，三個帳號 prouser1, prouser2, prouser3除了本身的 prouser1, prouser2, prouser3 group外，可以再加入共有的群組 progroup

```
[root@localhost ~]# groupadd progroup
[root@localhost ~]# grep progroup /etc/group
progroup:x:1016:  <==確定有 progroup 在設定檔當中了

[root@localhost ~]# useradd -G progroup prouser1
[root@localhost ~]# useradd -G progroup prouser2
[root@localhost ~]# useradd -G progroup prouser3
[root@localhost ~]# id prouser1
uid=1015(prouser1) gid=1017(prouser1) groups=1017(prouser1),1016(progroup)

[root@localhost ~]# echo mypassword
mypassword  <== echo 會將訊息從螢幕上輸出

[root@localhost ~]# echo mypassword | passwd --stdin prouser1
Changing password for user prouser1.
passwd: all authentication tokens updated successfully.
[root@localhost ~]# echo mypassword | passwd --stdin prouser2
[root@localhost ~]# echo mypassword | passwd --stdin prouser3
```

帳號與權限功用:單一用戶所有權

- 使用者能使用系統上面的資源與權限有關，因此帳號建後之後，就需要與權限搭配設計
- 例如從/etc複制的檔案，通常只有root可以使用，若一般user要用，要修改權限

```
[root@localhost ~]# ls -l /etc/securetty
-rw-----. 1 root root 221 Aug 12 2015 /etc/securetty <==一般用戶根本沒權限

[root@localhost ~]# cp /etc/securetty ~student/
[root@localhost ~]# ls -l ~student/securetty
-rw-----. 1 root root 221 Jul  1 19:33 /home/student/securetty

[root@localhost ~]# chown student.student ~student/securetty
[root@localhost ~]# ls -l ~student/securetty
-rw-----. 1 student student 221 Jul  1 19:33 /home/student/securetty
```

- 將指令移動到自己的家目錄來處理，例如想要將 ls 複製成為 myls 並且直接執行 myls 來取代，可以這樣處理：
- 將權限改為 700 之後使用 ./mysls 來執行，
- 想要直接輸入 myls 即可，那需要放入使用者自己家目錄下的 bin 子目錄才行

```
[student@localhost ~]$ cp /bin/ls myls
[student@localhost ~]$ ls -l myls
-rwxr-xr-x. 1 student student 117616  7月  1 19:37 myls

[student@localhost ~]$ chmod 700 myls
[student@localhost ~]$ ls -l myls
-rwx-----. 1 student student 117616  7月  1 19:37 myls

[student@localhost ~]$ myls
bash: myls: 找不到指令...

[student@localhost ~]$ ./mysls
bin  myipshow.txt  myls  securetty  下載  公共  圖片  影片  文件  桌面  模板  音樂

[student@localhost ~]$ mkdir bin
[student@localhost ~]$ mv myls bin
[student@localhost ~]$ myls
bin  myipshow.txt  securetty  下載  公共  圖片  影片  文件  桌面  模板  音樂
```

環境變數

- 環境變數則是無時無刻都存在系統的背景環境中的資訊，可以讓我們去存取裝在變數裡面的資料。
- 環境變數有很多，不同的環境變數有不用的功用
- **PATH**環境變數
 - 作業系統會依照**PATH**環境變數中所設定的路徑順序，依序尋找各路徑下是否有要使用的指令。
 - 若找不到，就會顯示**command not found**
 - 若多個目錄下都有指令，已優先找到的為主

```
root@ubuntu:~# echo $PATH
/etc/ansible/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:
root/go/bin
```


帳號與權限功用:群組共用功能

- 透過權限設定，讓大家可以共用資料
- progroup 成員 prouser1, prouser2, prouser3，需要共用 /srv/project1/ 的目錄下的資料

```
[root@localhost ~]# mkdir /srv/project1
[root@localhost ~]# chgrp progroup /srv/project1
[root@localhost ~]# chmod 770 /srv/project1
[root@localhost ~]# ls -ld /srv/project1
drwxrwx---. 2 root progroup 6 7月 1 19:46 /srv/project1
```

例題：

- 讓 **student** 帳號直接執行 **mymore** 達成與 **more** 相同功能的目的
 - 若無**student**帳號，請自行建立
 - 亦即將 **more** 複製成為 **mymore**，並放置到正確的位置即可
 - 同時放一份至 **/usr/local/bin**
 - 若要 額外“僅” 讓**progroup** 的三個使用者可以使用**mymore**，權限要如何設定