程序之觀察與基本管理

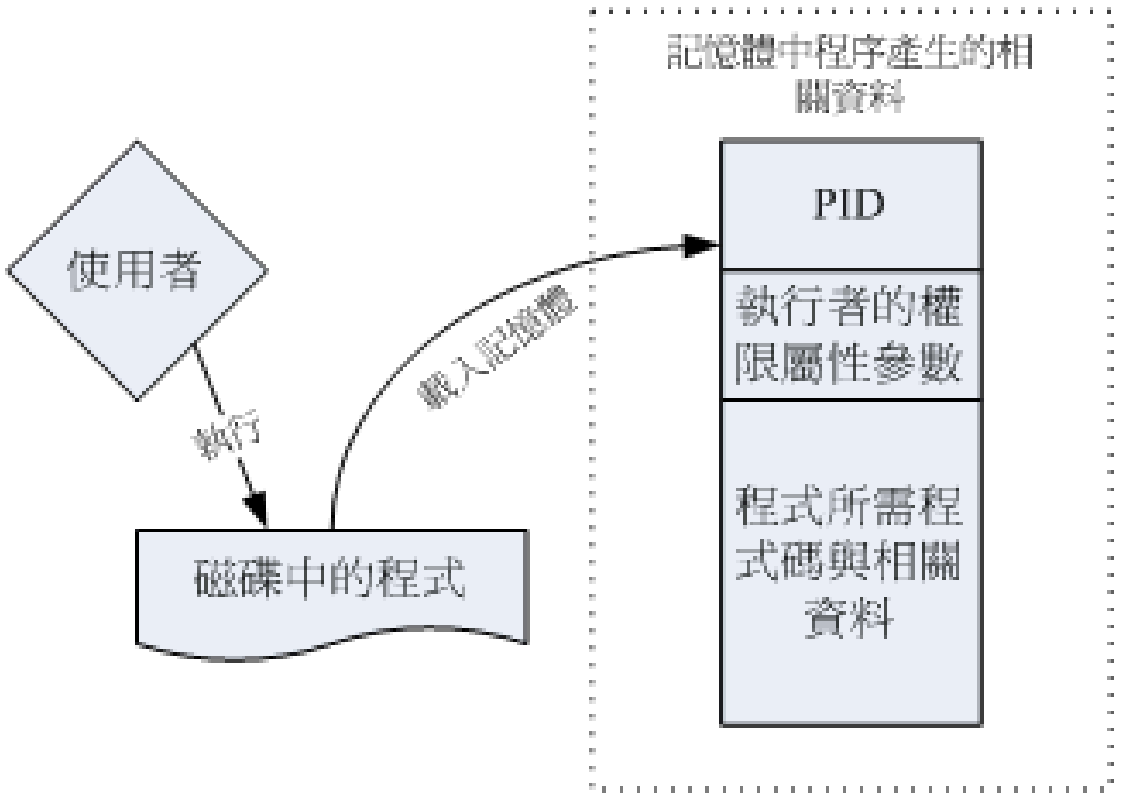
Reading material

- 程序管理初探
  - http://linux.vbird.org/linux_basic_train/unit05.php#5.2
- 程序管理
  - http://linux.vbird.org/linux_basic/0440processcontrol.php
- 特殊權限 SUID/SGID/SBIT 的功能
  - http://linux.vbird.org/linux_basic_train/unit05.php#5.3

程式 &程序

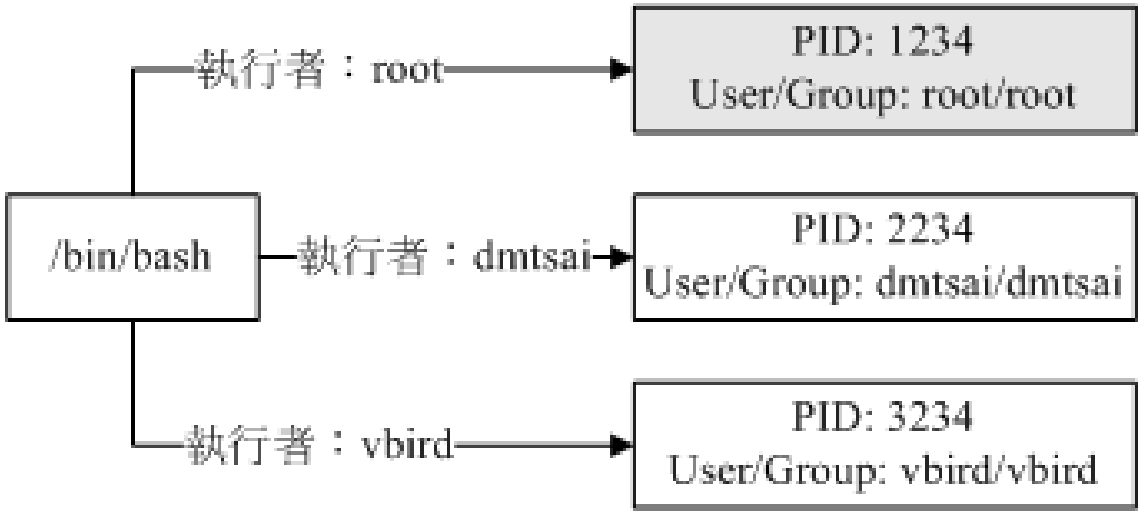
- 程式 (program)：
  - 通常為 binary program ，放置在儲存媒體中 (如硬碟、光碟、軟碟、磁帶等)， 為實體檔案的型態存在；
- 程序 (process)：
  - 程式被觸發後，執行者的權限與屬性、程式的程式碼與所需資料等都會被載入記憶體中，作業系統並給予這個記憶體內的單元一個識別碼 (PID)，可以說，程序就是一個正在運作中的程式。

- Eg: iphone rom 256gb，2gb ram
  - 下載後的app是放在 rom裡，執行時會佔用ram


記憶體中程序產生的相關資料
使用者
執行
載入記憶體
磁碟中的程式
PID
執行者的權限屬性參數
程式所需程式碼與相關資料

- 系統當中可能會有多個程式碼相同的process存在
  - 同一隻程式有時候會被多個使用者執行觸發
  - 同一個使用者執行同一隻程式多次

- PID (Process ID) :
  - 用來唯一識別process



觀察程序的工具指令: 觀查自己的程序

```
[student@localhost ~]$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  1685  1684  0  80   0 - 29011 wait   pts/0    00:00:00 bash
0 R  1000  4958  1685  0  80   0 - 34343 -      pts/0    00:00:00 ps
```

-
- UID/PID/PPID：代表『此程序被該 UID 所擁有/程序的 PID 號碼/此程序的父程序 PID 號碼
  - Root的uid為0，非0的uid都是一般user
- TTY：登入者的終端機位置
- CMD：造成此程序的觸發程式

觀察程序的工具指令: 觀查系統全部的程序

$ ps aux

```
[student@localhost ~]$ ps aux
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.4 128236  9068 ?        Ss    6月13   1:02 /usr/lib/systemd/systemd ...
root          2  0.0  0.0      0     0 ?        S     6月13   0:00 [kthreadd]
root          3  0.0  0.0      0     0 ?        S     6月13   0:00 [ksoftirqd/0]
root          7  0.0  0.0      0     0 ?        S     6月13   0:00 [migration/0]
root          8  0.0  0.0      0     0 ?        S     6月13   0:00 [rcu_bh]
.......(中間省略)......
student   17301  0.1  1.0 728996 22508 ?        Sl    18:34   0:01 /usr/libexec/gnome-terminal-server
student   17307  0.0  0.0   8480   720 ?        S     18:34   0:00 gnome-pty-helper
student   17308  0.0  0.1 116156  2864 pts/1    Ss+   18:34   0:00 bash
.......(底下省略)......
```

$ top

```
[student@localhost ~]$ top
top - 19:02:56 up 21 days, 19:16,  3 users,  load average: 0.00, 0.01, 0.05
Tasks: 184 total,   1 running, 183 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  2048964 total,   172968 free,   517972 used,  1358024 buff/cache
KiB Swap:  2097148 total,  2096800 free,      348 used.  1283612 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
18432 student   20   0  146148   2120   1436 R   0.5  0.1   0:00.09 top
    1 root      20   0  128236   9068   2640 S   0.0  0.4   1:02.41 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.43 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.01 ksoftirqd/0
    7 root      rt   0       0      0      0 S   0.0  0.0   0:00.42 migration/0
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/0
   10 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/1
   11 root      20   0       0      0      0 S   0.0  0.0   1:05.20 rcu_sched
```

觀察程序的工具指令: 父程序與子程序

- $ yum install psmise
- $ pstree

```
[student@localhost ~]$ pstree -A
systemd-+-ModemManager---2*[{ModemManager}]
        |-NetworkManager---2*[{NetworkManager}]
.......(中間省略)......
        |-gnome-shell-cal---4*[{gnome-shell-cal}]
        |-gnome-terminal--+-bash---pstree
        |                 |-gnome-pty-helpe
        |                 `-3*[{gnome-terminal-}]
.......(底下省略)......
```

```
[student@localhost ~]$ pstree -Aup
systemd(1)-+-ModemManager(822)-+-{ModemManager}(838)
           |                   `-{ModemManager}(863)
           |-NetworkManager(921)-+-{NetworkManager}(931)
           |                     `-{NetworkManager}(936)
.......(中間省略)......
           |-gnome-shell-cal(16734,student)-+-{gnome-shell-cal}(16741)
           |                                |-{gnome-shell-cal}(16784)
           |                                |-{gnome-shell-cal}(16785)
           |                                `-{gnome-shell-cal}(16891)
           |-gnome-terminal-(17301,student)-+-bash(17308)---pstree(17705)
           |                                |-gnome-pty-helpe(17307)
           |                                |-{gnome-terminal-}(17302)
           |                                |-{gnome-terminal-}(17303)
           |                                `-{gnome-terminal-}(17304)
.......(底下省略)......
```

例題：

- 透過各種方法，找到 PID 為 1 的那隻程序的指令名稱為何？
- 使用 student 身份登入系統後，
  - (1)使用 su - 切換身份，以 pstree 觀察程序情況
  - (2)再使用 su - student，以 pstree 觀察程序情況
  - (3)再使用 su - 切換成 root，以 pstree 觀察程序情況
  - (4)根據分析上述的程序相依性，你需要使用幾次 exit 才能回到原本的 student 帳號？
- 如到找出名為 crond 的程序的 PID 號碼。

特殊權限 SUID/SGID/SBIT

- 傳統權限的不足
  - 系統的密碼紀錄在 /etc/shadow 內，但是使用者並沒有權限可以更改，不過一般用戶確實有自己修改密碼的需求。

SUID功能與觀察

- SUID:
  - Set UID的技術來處理這方面的疑問
  - 系統設定一個 SUID 的權限旗標到 passwd 執行擋上， 當使用者執行 passwd 指令時，就能夠藉由 SUID 來切換成該程式(passwd)owner的權限。

```
[student@localhost ~]$ ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 27832  6月 10  2014 /usr/bin/passwd
```

- 使用者權限的 x 變成了 s :此即 SUID 的權限旗標
- 只要任何人具有 x 的執行權，當用戶執行 passwd 時，就會自動透過 SUID 轉換身份成為 owner ，亦即變成了 root 的身份

例題：

- 以一般帳號的身份執行 passwd，但停留在輸入密碼
- 按ctrl + z 將passwd放至背景執行
- 現在可以使用 pstree -pu 觀察 passwd 與前、後程序的擁有者變化
- 執行fg，將passwd放回前景，用ctrl+c刪除

- 若不使用後景/前景概念，可以開另一個terminal登入

SGID 的功能與觀察

- SUID 類似
- 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權限

例題：

- 使用 locate 查詢系統檔名
  - $ yum install mlocate
  - $ locate passwd 查詢passwd的路徑
  - locate 所取用的檔名資料庫放置於 /var/lib/mlocate 當中
- 請問 一般用戶 有沒有權限可以進入該目錄？
- 為何 一般用戶 操作 locate 可以進入 /var/lib/mlocate 目錄？
  - 查詢 locate 的權限，是否具有 SGID 的權限旗標？
  - locate 的擁有群組為何?

SBIT 的功能與觀察

- SBIT:
  - Sticky bit
  - 當使用者對於此目錄具有寫入的權限時，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

例題：

- 觀察 /tmp 的權限，看其他人的權限當中的 x 變成什麼？
- 以 root 登入系統，並且進入 /tmp 當中；
- 將 /etc/hosts 複製成為 /tmp/myhosts ，並且更改 /tmp/myhosts 權限成為 777 ；
- 以 一般帳號 登入，並進入 /tmp；
- 一般帳號能不能使用 vim 編輯這個檔案？為什麼？
- 一般帳號 能不能刪除這個檔案？為什麼？

設定SUID/SGID/SBIT 權限: 數字法

- 這三個特殊權限分別只用在owner，group，other，所以不用特地區分
  - 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
```

- 要加上SBIT ，用chmod 1xyz dirname
- 要加上SUID ，用chmod 4xyz filename
- 要加上SGID ，用chmod 2xyz filename

設定SUID/SGID/SBIT 權限: 符號法

- SUID: chmod u+s filename
- SGID: chmod g+s filename
- SBIT: chmod o+t filename

例題：

- 執行cat /etc/shadow 會有permission denied問題
- 復制/usr/bin/cat為mycat，並放在$PATH，設定mycat的權限，讓一般使用者執行 mycat /etc/shadow 會順利執行成功。