Linux Shell程式設計實務

sed 和 awk 入門

13

悅知文化
Delight Press

設計 script 時，有時候需修改檔案，例如：刪除或置換某些關鍵字。像這種在 script 執行過程，動態修改檔案的作法，稱為串流編輯。具有串流編輯能力的工具，稱為串流編輯器（stream editer）。sed 是這方面的佼佼者，可補 bash 的不足。另外，script 執行時可能要製作報表，呈現各欄位資訊。傳統上，能和 bash 完美搭配的，當推 awk 莫屬。

本章將介紹這兩個強大工具的基本用法，熟悉這兩者，可讓你的 script 如虎添翼，威力大增。

13-1 正規表示式

在介紹 sed 和 awk 之前，要先具備正規表示式的基本知識。

正規表示式是組成「樣式」的基本語法，而「樣式」是運用 sed 和 awk 必備的知能。sed 和 awk 共通的運作方式是：只要符合「樣式」的資料列，就對它執行指定的「動作」。因此，瞭解樣式的基本語法，運用 sed 和 awk 才能得心應手。

何謂正規表示式

正規表示式是一種描述的方法，一種小型的語言，可表示某種樣式，或若干種樣式的組合，它的威力在於僅需幾個簡單的符號，便可代表許多字串共同的樣子。這是固定樣式無法比擬的，例如，樣式 `shell`，只能比對固定的字串，作用不大，但若改成 `sh*`，卻可比對 `she`、`shell`、`short` 等多種字串，涵蓋面較大，因此，能發揮的作用較強。

以下介紹正規表示式的語法：

. 一點代表一個字元

用途 . 代表任意的字元。

例 1：樣式 .T. ，代表三個字元，中間是 T，左右二邊是任意的一個字元。

例 2：...

代表字元長度是 3 的字串。若想比對 . 這個字元本身，需加上跳脫字元（\），寫成\.。例如：樣式 data\.... 代表 data. 後接三個字元，如 data.txt、data.cfg、data.123 等等，都符合這個樣式，但 data1234 就不符合了，因為四個點最左邊的那個點，已經用\跳脫其特殊意義，還原為 . 這個字元本身，因此，\....和 1234 比對不符。

^ 在列首

用途 ^ 代表位置在列的開頭。

例如：樣式 ^Jack，代表 Jack 應出現在列首，才算符合樣式。像 Jack and Marry 123 就符合此一樣式，但 Hi Jack 就不符合，因為 Jack 沒有出現在該列的最前面。

$ 在尾部

用途 $ 代表位置在列的最後面。

例如：樣式 123$，代表在列的最後面是 123。像 Jack and Marry 123 即符合此一樣式。

[...] 字元集合

用途 [...] 代表字元串列中的一個字元（長度為 1 個字元）。

例 1：樣式 [ABc]，代表 A 或 B 或 c 這三個字元中的一個。

例 2：[Ss]ame 代表 Same 或 same。

以下是常見的用法：

[A-Z]	一個大寫字母。
[a-z]	一個小寫字母。
[0-9]	一個數字。
[^A-Z]	除了大寫字母之外的一個字元。
[^a-zA-Z]	一個非英文字母的字元。
[^a-zA-Z0-9]	一個非英文字母、且非數字的字元。

^ 出現在括號裡的第一個位置，代表「非/不是」之意。

* 出現 0 個以上

用途 * 代表前面的（左鄰）字元有 0 個或 0 個以上。

例如：樣式 aA*c，代表 A 這個字元可能出現 0 個或 0 個以上。例如：ac、aAc、aAAc、aAAAc 都符合此一樣式。

\{...\} 指定符合的個數

用途 指定前面的（左鄰）字元的個數。

例如：`\{3,5\}` 表示前面的字元有 3 到 5 個。`[a-z]\{3,5\}`代表以小寫字母組成的字串，長度是 3~5。

\(...\) 把比對符合的字串暫時保存起來

例如：`H\(...\)y` 表示要保存 H 和 y 之間的三個字元。

若欲取用保存的字串，可用位置參數，`\1` 代表第一個保存的字串，`\2` 代表第二個，其他依此類推。

13-2 sed 的用法

sed 是一種非交談式的串流編輯器，可動態編輯檔案。所謂非交談式是說，sed 和傳統的文字編輯器不同，並非和使用者直接互動，sed 處理的對象是檔案的資料流（稱為 stream/串流）。sed 的工作模式是，比對每一資料列，若符合樣式，就執行指定的動作。

Tip

本節介紹的 sed 是指 GNU 版的 sed。執行 sed --version 可查看版本資訊。

sed 的語法如下：

```
sed '樣式命令' 檔案
```

它的意思是說：如果檔案中某一列符合「樣式」，就執行指定的 sed 命令，例如刪除（d）或取代（s）。

這裡的「樣式」使用一對//含括，表示尋找之意；也可以指定資料列的範圍，例如：1,6 表作用範圍是由第 1 列到第 6 列；/AAA/,/DDD/表作用範圍是從含有 AAA 的資料列，到含有 DDD 的資料列。

請特別注意：sed 並不會更改檔案內容。sed 的工作方式是讀取檔案內容，經串流編輯之後，把結果顯示到標準輸出。因此，如果想要儲存 sed 的處理結果，得自行運用轉向輸出將結果存成其他檔案。

以下介紹 sed 的各種用法：

■ sed 的用法 1：刪除某一段範圍的資料列。

```
sed '1,4d' dataf1
```

用途 把第 1 到第 4 列資料刪除，剩下的顯示出來。d 是 sed 的刪除命令。

■ sed 的用法 2：把含有「樣式」的資料列刪除。

```
sed '/La/d' dataf3
```

用途 把含有 La 的列刪除，剩下的顯示出來。其中，/ / 代表搜尋之意。

```
sed '/[0-9]\{3\}/d' dataf3
```

用途 把含有「3 位數」的列刪除，剩下的顯示出來。

在樣式[0-9]\{3\}中，\{3\} 表/ /要尋找的是 3 個數字組成的字串。

```
sed '/^$/d' dataf5
```

刪除 dataf5 的空白列。^ 表開頭，$ 表尾部，這兩者之間沒有任何字元，代表該列是一空白列。

■ sed 的用法 3：把不含有「樣式」的資料列刪除。

```
sed '/La/!d' dataf3
```

用途 把不含有 La 的列刪除，剩下的顯示出來。

這裡的!是否定的意思，表示不符合樣式者。

■ sed 的用法 4：把含有「樣式」的資料列顯示出來。

```
sed '/La/p' dataf3
```

用途 把含有 La 的列秀出來。其中，p 是 sed 的命令，它會把目前的資料顯示出來，但因為 sed 預設也會顯示不符合的資料列，所以，應改用以下指令：

```
sed -n '/La/p' dataf3
```

選項 -n 會抑制 sed 秀出其他資料列的預設動作，只顯示符合樣式的資料列。

■ sed 的用法 5：取代。

```
sed -n 's/La/Oo/p' dataf3
```

這裡的 s 是取代的意思，第一對 //中含括的字串（La）是搜尋的目標，第二對 //含括的是置換的字串（Oo）。它會把資料列中的字串 La 換成 Oo。

請注意：上面這個指令，只會更換第一個出現的 La 而已，欲全部置換，應再加上全域的命令 g，如下所示：

```
sed -n 's/La/Oo/gp' dataf3
```

這樣就會把所有找到的 La 全換成 Oo 了。

取代的用法，還有以下幾個：

用例1

```
sed -n 's/La//p' dataf3
```

把每一列第一個出現的 La 刪除。（把 La 置換成空字串，就是刪除）

用例2

```
sed 's/^...//' dataf3
```

把每一列開頭的 3 個字元刪除。

用例3

```
sed 's/...$//' dataf3
```

把每一列末尾 3 個字元刪除。

■ sed 的用法 6：取用符合樣式的字串。

```
sed -n 's/\(La\)/\1Oo/p' dataf3
```

把找到的 La 存起來，用\1 取回來再使用。

這個指令作用的結果：若資料列含有 La 字串，則第一個出現的 La 會置換成 LaOo，然後，再顯示這些含有 La 的資料列。

■ sed 的用法 7：找到符合樣式的資料列後，再進行取代的動作。

用例1

```
sed -n '/AAA/s/234/567/p' dataf3
```

用途 找到含有 AAA 的那一列之後，將 234 換成 567。

用例2

```
sed -n '/AAA/,/DDD/s/B/567/p' dataf3
```

用途 將含有 AAA 到含有 DDD 的那幾列，皆將其中的 B 換成 567。

用例3

```
sed -n '2,4s/B/567/p' dataf3
```

用途 由第 2 列到第 4 列，皆將其中的 B 換成 567。

由以上的說明可知：sed 動態編輯的威力是相當強大的，它補足了 bash 在修改檔案方面能力的不足。

實例應用

設計 script 時，我們經常會利用 sed 置換系統設定檔裡的關鍵字，以開啟或關閉某個設定選項。

若用 bash 來做，可能要大費周章，但用 sed 來做，通常只要一列指令，就可搞定。

如範例 13-2-1，這裡想要開啟 vsftpd 匿名登入的功能：

範例 13-2-1：anonyftp1.sh

```
01.     #! /bin/bash
02.
03.     # 修改 vsftpd 的設定檔，開放匿名 FTP 服務
04.
05.     VSFTPD_conf='/etc/vsftpd.conf'
06.     TMP_file="/tmp/tmp.$$"
07.
08.    # 將 anonymous_enable 選項，設成 YES，這樣，vsftpd 就會開啟匿名 FTP 登入的功能。
09.     sed s/^.*anonymous_enable=.*/anonymous_enable=YES/ $VSFTPD_conf >
        $TMP_file
10.     mv -f $TMP_file $VSFTPD_conf
```

說明

- 列 5，`VSFTPD_conf` 存放 vsftpd 的設定檔。
- 列 6，`TMP_file` 是暫存檔，用來儲存 sed 編輯後的結果。`$$`是 script 的行程編號，利用`$$`組成暫存檔名，這樣，可避免不同的 script 開啟重複的暫存檔。
- 列 9，將 `anonymous_enable` 的選項設成 YES。作法如下：

由於在`/etc/vsftpd.conf` 中，這一列的預設值可能是：

```
anonymous_enable=NO
```

或者被 # 註解掉了：

```
#anonymous_enable=NO
```

因此，筆者決定採用樣式`^.*anonymous_enable=.*` 來做比對，因為`^`表示由列首開始比對，緊接著 0 個以上的字元，如此，這就能包含 #可能出現的情況，接著再用 `anonymous_enable=.*` 比對 `anonymous_enable=NO` 或 `anonymous_enable=YES` 即可。

找到符合的資料列之後，進行取代，把指定型態的字串，換成 `anonymous_enable=YES`，再將修改結果轉向儲存至暫存檔中。

- 列 10，將暫存檔覆蓋原本的設定檔，如此，即可更新設定檔，開啟匿名登入的功能。

範例 13-2-1，只能開啟匿名登入，其實，做法還可以更有彈性一點，例如：執行時，加上選項 on 或 off，即可切換：開啟或關閉匿名登入。

範例 13-2-2：anonyftp.sh

```
 01.     #! /bin/bash
 02.
 03.     # 修改 vsftpd 的設定檔，切換：'開啟/關閉' 匿名 FTP 服務
 04.
 05.     if [ $# -ne 1 ]; then
 06.        echo "Usage: $0 on 或 $0 off"
 07.        exit 1
 08.     fi
 09.
 10.     OPT=$1
 11.     case "$OPT" in
 12.        [Oo][Nn]) CMD='YES';;
 13.        [Oo][Ff][Ff]) CMD='NO';;
 14.        *)
 15.          echo '選項格式錯誤！ 請用 on 或 off 來切換匿名登入的開關。'
 16.          exit 1
 17.          ;;
 18.     esac
 19.
 20.     VSFTPD_conf='/etc/vsftpd.conf'
 21.     TMP_file="/tmp/tmp.$$"
 22.
 23.     sed s/^.*anonymous_enable=.*/anonymous_enable=$CMD/ $VSFTPD_conf >
$TMP_file
 24.     mv -f $TMP_file $VSFTPD_conf
```

說明

- 列 5~8，檢查使用者提供的選項個數是否剛好。
- 列 10，變數 OPT 存放選項值。
- 列 11，使用 case 語法，判斷選項是 on 或 off。
- 列 12~13，利用字元集合，讓選項不分大小寫，均可接受。若選項是 on，CMD 變數值就設為 YES；若是 off，就設為 NO。CMD 變數，會放在列 23，做為 anonymous_enable 的設定值。
- 列 14~17，若選項不是 on 或 off，就顯示選項錯誤的訊息，然後 exit 離開 script。

■ 列 20~24，同範例 13-2-1 的說明。

使用方法：

■ 欲關閉匿名登入，請執行：

```
./anonyftp.sh off
```

■ 欲開啟匿名登入，請執行：

```
./anonyftp.sh on
```

當然，要讓設定生效，得重新啟動 vsftpd 才行。在 B2D Server 中，做法很簡單，如下所示：

```
service vsftpd restart
```

或

```
/etc/init.d/vsftpd restart
```

13-3 awk 的用法

awk 是一種可以處理資料、產生格式化報表的語言，功能相當強大。awk 的工作方式是讀取資料檔，將每一列資料視為一筆記錄（record），每筆記錄以欄位分隔符號分成若干欄位，然後，輸出各個欄位的值。

以下是執行 `ps auxw` 的輸出片段：

```
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.0    164    92 ?        Ss   Apr09   0:01 init [5]
```

像這種固定結構的資料，用 awk 來處理，特別有威力，通常只要短短數列程式碼，就可以完成工作。

例如，僅用以下單一指令，就可取得所有行程的 PID：

```
ps auxw | awk '{print $2}'
```

那麼，awk 是如何處理每一筆記錄的呢？

awk 對每一筆記錄，都會套用一個「樣式{動作}」，如果該列符合樣式，就執行指定的動作。樣式或動作之一，可以省略。如果只有樣式，表示要顯示符合樣式的資料列；如果只有動作，表示對每一資料列都執行該項動作。

以下是 awk 常用的作用格式：

- awk「樣式」檔案：把符合樣式的資料列顯示出來。
- awk '{動作}' 檔案：對每一列都執行{}中的動作。
- awk '樣式{動作}' 檔案：對符合樣式的資料列，執行{}中的動作。

用例

- awk 的用法 1：

```
awk '/La/' dataf3
```

顯示含 La 的資料列。

- awk 的用法 2：

```
awk '{ print $1, $2 }' dataf3
```

顯示 dataf3 每一列的第 1 和第 2 個欄位。

$1 代表第 1 個欄位，$2 代表第二欄位，其他依此類推。

- awk 的用法 3：

```
awk '/La/{ print $1, $2 }' dataf3
```

將含有 La 關鍵字的資料列的第 1 及第 2 個欄位顯示出來。

- awk 的用法 4：

```
awk -F: '/^ols3/{ print $3, $4 }' /etc/passwd
```

使用選項 -F，指定:為分隔字元，帳號 ols3 的 uid（第 3 欄位）及 gid（第 4 欄位）顯示出來。

- awk 的用法 5：

```
awk -F: 'BEGIN{OFS="+++"}/^ols3/{ print $1, $2, $3, $4, $5 }' /etc/passwd
```

以:為分隔字元，+++為輸出欄位分隔符號，將帳號 ols3 的第 1~5 欄顯示出來。

執行結果：

```
ols3+++x+++1002+++1002+++
```

本例中，BEGIN{}區塊指示 awk 一開始先做初始化的動作，即設定 OFS="+++"。變數 OFS 的作用是儲存輸出欄位的分隔符號。接著，尋找 ols3 的帳號列，找到後，使用 print 印出第 1～第 5 個欄位，且彼此用 +++ 隔開。

實例應用

■ 取得網卡的 IP：

```
ifconfig | grep 'inet addr:' | grep Bcast | awk '{print $2}' | awk -F: '{print $2}'
```

■ 取得網路設備名稱：

```
cat /proc/net/dev | awk -F: '/eth.:|ppp.:|wlan.:/{print $1}'
```

在本例中，-F: 把分隔字元設為：，而且，採用多選一的樣式 /eth.:|ppp.:|wlan.:/。這個樣式的意思是：設備名稱可以是 eth0:、ppp1:、wlan1: 這三個其中之一。一旦找到符合樣式的字串後，去掉:，取其中的第一個欄位值，因此，可能的答案是 eth0 或 ppp1 或 wlan1。

■ 取得系統記憶體大小：

```
cat /proc/meminfo | awk '/MemTotal/{print $2}'
```

/proc/meminfo 記載主機記憶體相關資料，其中 MemTotal 為記憶體大小，其樣本值如下：

```
MemTotal:       223128 kB
```

因此，在 awk 的樣式語法中，利用/MemTotal/ 找到這一列，再印出第二個欄位，即可得到記憶體的大小。

■ 修改 CSV 檔各欄位的順序：

以下是資料檔 dataf6.csv，想要把第 2 個欄位和第 4 欄位調換：

```
所在鄉鎮,學校名稱,學校網址,校長姓名,學校電話,VOIP 前三碼,學校地址
新營市,南新國中,http://www.ns12jh.tnc.edu.tw,ABC,06-656313012,1021,新營市民治路 6675 號
佳里鎮,佳里國中,http://www.jl41jh.tnc.edu.tw,NOP,06-722224432,1146,佳里鎮安南路 5523 號
新營市,新營國小,http://www.sy53es.tnc.edu.tw,DEF,06-632213642,1482,新營市中正路 3248 號
```

做法如下：

範例 13-3-1：chcsv24.sh

```
 01.     #! /bin/bash
 02.
 03.     TMPF='/tmp/tmp.$$'
 04.    cat dataf6.csv | awk -F, 'BEGIN{OFS=","}{print $1,$4,$3,$2,$5,$6,$7}'
> $TMPF
 05.     mv -f $TMPF dataf6.csv
```

說明

- 列 3，設定暫存檔名。
- 列 4，將資料檔的內容透過管線餵給 awk 處理。awk 的欄位分隔字元和輸出分隔字元，皆設為,。在{}的動作中，調換 $2 和 $4 的順序，再把結果轉向儲存在暫存檔。
- 列 5，將暫存檔覆蓋原檔。