Chapter

13

sed 和 awk 入门

设计 Script 时，有时候需修改脚本，例如：删除或置换某些关键词。像这种在 Script 执行过程动态修改文件的做法，称为流编辑。具有流编辑能力的工具，称为流编辑器（stream editer）。sed 是这方面的佼佼者，可补 Bash 的不足。另外，Script 执行时可能要制作报表，呈现各字段信息。传统上，能和 Bash 完美搭配的，非 awk 莫属。

本章将介绍这两个强大工具的基本用法，熟悉这两者，可让你的 Script 如虎添翼，威力大增。

13.1 正则表达式

在介绍 sed 和 awk 之前，要先具备正则表达式的基本知识。

正则表达式是组成“样式”的基本语法，而“样式”是运用 sed 和 awk 必备的能力。sed 和 awk 相同的运行方式是：只要符合“样式”的数据行，就对它执行指定的“操作”。因此，了解样式的基本语法，运用 sed 和 awk 才能得心应手。

何谓正则表达式

正则表达式是一种描述的方法，一种小型的语言，可表示某种样式或若干种样式的组合，它的威力在于仅需几个简单的符号，便可代表许多字符串共同的样子。这是固定样式无法比拟的，例如，样式 Shell，只能比对固定的字符串，作用不大，但若改成 sh*，却可比对 she、Shell、short 等多种字符串，涵盖面较大，因此，发挥的作用较强。

以下介绍正则表达式的语法：

. 一点代表一个字符

用途 . 代表任意的字符。

例 1：样式 .T. ，代表 3 个字符，中间是 T，左右两边是任意的一个字符。

例 2：...

代表字符长度是 3 的字符串。若想比对 . 这个字符本身，需加上转义字符（\），写成\.。例如：样式 data\....代表 data.后接 3 个字符，如 data.txt、data.cfg、data.123 等，都符合这个样式，但 data1234 就不符合了，因为 4 个点最左边的那个点，已经用\转义其特殊意义，还原为.这个字符本身，因此，\....和 1234 比对不符。

^ 在行首

用途 ^ 代表位置在行的开头。

例如：样式 ^Jack，代表 Jack 应出现在行首，才算符合样式。像 Jack and Marry 123 就符合此样式，但 Hi Jack 就不符合，因为 Jack 没有出现在该行的最前面。

$在尾部

用途 $ 代表位置在行的最后面。

例如：样式 123$，代表在行的最后面是 123。像 Jack and Marry 123 即符合此样式。

[...] 字符集合

用途 [...] 代表字符串行中的一个字符（长度为 1 个字符）。

例 1：样式 [ABc]，代表 A 或 B 或 c 这 3 个字符中的一个。

例 2：[Ss]ame 代表 Same 或 same。

以下是常见的用法：

[A-Z]	一个大写字母
[a-z]	一个小写字母
[0-9]	一个数字
[^A-Z]	除了大写字母之外的一个字符
[^a-zA-Z]	一个非英文字母的字符
[^a-zA-Z0-9]	一个非英文字母、且非数字的字符

^ 出现在括号里的第一个位置，代表“非/不是”之意。

* 出现 0 个以上

用途 * 代表前面的（左邻）字符有 0 个或 0 个以上。

例如：样式 aA*c，代表 A 这个字符可能出现 0 个或 0 个以上。例如：ac、aAc、aAAc、aAAAc 都符合此样式。

\{...\} 指定符合的个数

用途 指定前面的（左邻）字符的个数。

例如：\{3,5\} 表示前面的字符有 3～5 个。[a-z]\{3,5\}代表以小写字母组成的字符串，长度是 3～5。

\(...\) 把比对符合的字符串暂时保存起来

例如：H\(...\)y 表示要保存 H 和 y 之间的 3 个字符。

若要提取保存的字符串，可用位置参数，\1 代表第一个保存的字符串，\2 代表第二个，其他类推。

13.2 sed 的用法

sed 是一种非交互式的流编辑器，可动态编辑文件。所谓非交互式是说，sed 和传统的文本编辑器不同，并非和使用者直接互动，sed 处理的对象是文件的数据流（称为 stream/流）。sed 的工作模式是，比对每一数据行，若符合样式，就执行指定的操作。

Tip

本节介绍的 sed 是指 GNU 版的 sed。执行 sed --version 可查看版本信息。

sed 的语法如下：

```
sed '样式命令' 文件
```

它的意思是说：如果文件中某一行符合“样式”，就执行指定的 sed 命令，如删除（d）或取代（s）。

这里的“样式”使用一对//含括，表示寻找之意；也可以指定数据行的范围，例如：1,6 表示作用范围是由第 1 行到第 6 行；/AAA/,/DDD/表示作用范围是从含有 AAA 的数据行，到含有 DDD 的数据行。

请特别注意：sed 并不会更改文件内容。sed 的工作方式是读取文件内容，经流编辑之后，把结果显示到标准输出。因此，如果想要存储 sed 的处理结果，得自行运用转向输出将结果存成其他文件。

以下介绍 sed 的各种用法：

- sed 的用法 1：删除某一段范围的数据行。

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

用途 把第 1 到第 4 行数据删除，剩下的显示出来。d 是 sed 的删除命令。

- sed 的用法 2：把含有“样式”的数据行删除。

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

用途 把含有 La 的行删除，剩下的显示出来。其中，// 代表搜索之意。

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

用途 把含有“3 位数”的行删除，剩下的显示出来。

在样式[0-9]\{3\}中，\{3\} 表//要寻找的是 3 个数字组成的字符串。

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

删除 dataf5 的空白行。^ 表开头，$ 表尾部，这两者之间没有任何字符，代表该行是一空白行。

- sed 的用法 3：把不含有“样式”的数据行删除。

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

用途 把不含有 La 的行删除，剩下的显示出来。

这里的!是否定的意思，表示不符合样式者。

- sed 的用法 4：把含有“样式”的数据行显示出来。

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

用途 把含有 La 的行显示出来。其中，p 是 sed 的命令，它会把目前的数据显示出来，但因为 sed 默认也会显示不符合的数据行，所以，应改用以下指令：

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

选项-n 会抑制 sed 显示出其他资料行的默认操作，只显示符合样式的数据行。

- sed 的用法 5：取代。

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

这里的 s 是取代的意思，第一对//中含括的字符串（La）是搜索的目标，第二对//含括的是置换的字符串（Oo）。它会把数据行中的字符串 La 换成 Oo。

请注意：上面这个指令，只会更换第一个出现的 La 而已，如要全部置换，应再加上全局的命令 g，如下所示：

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

这样就会把所有找到的 La 全换成 Oo 了。

取代的用法，还有以下几个：

用例1

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

把每一行第一个出现的 La 删除（把 La 置换成空字符串，就是删除）。

用例2

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

把每一行开头的 3 个字符删除。

用例3

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

把每一行末尾 3 个字符删除。

- sed 的用法 6：取用符合样式的字符串。

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

把找到的 La 存起来，用\1 取回来再使用。

这个指令作用的结果：若数据行含有 La 字符串，则第一个出现的 La 会置换成 LaOo，然后再显示这些含有 La 的数据行。

- sed 的用法 7：找到符合样式的数据行后，再进行取代的操作。

用例1

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

用途 找到含有 AAA 的那一行之后，将 234 换成 567。

用例2

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

用途 将含有 AAA 到含有 DDD 的那几行，皆将其中的 B 换成 567。

用例3

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

用途 由第 2 行到第 4 行，皆将其中的 B 换成 567。

由以上的说明可知：sed 动态编辑的威力是相当强大的，它补足了 Bash 在修改文件方面能力的不足。

实例应用

设计 Script 时，我们经常会利用 sed 置换系统配置文件里的关键词，以开启或关闭某个设置选项。

若用 Bash 来做，可能要花费很大力气，但用 sed 来做，通常只要一行指令就可搞定。

如范例 13-2-1，这里想要开启 vsftpd 匿名登录的功能：

范例 13-2-1：anonyftp1.sh

```
01.    #! /bin/Bash
02.
03.    # 修改 vsftpd 的配置文件，开放匿名 FTP 服务
04.
05.    VSFTPD_conf='/etc/vsftpd.conf'
06.    TMP_file="/tmp/tmp.$$"
07.
08.    # 将 anonymous_enable 选项，设成 YES，这样，vsftpd 就会开启匿名 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
```

说明

- 行 5，VSFTPD_conf 存放 vsftpd 的配置文件。
- 行 6，TMP_file 是临时文件，用来存储 sed 编辑后的结果。$$是 Script 的行程编号，利用$$组成临时文件名，这样，可避免不同的 Script 开启重复的临时文件。
- 行 9，将 anonymous_enable 的选项设成 YES。做法如下：

由于在/etc/vsftpd.conf 中，这一行的默认值可能是：

```
anonymous_enable=NO
```

或者被#批注掉了：

```
#anonymous_enable=NO
```

因此，笔者决定采用样式^.*anonymous_enable=.* 来做比对，因为^表示由行首开始比对，紧接着 0 个以上的字符，如此，这就能包含 #可能出现的情况，接着再用 anonymous_enable=.* 比对 anonymous_enable=NO 或 anonymous_enable=YES 即可。

找到符合的数据行之后，进而取代，然后把指定形态的字符串换成 anonymous_enable=YES，再将修改结果转向存储至临时盘中。

- 行 10，将临时盘覆盖原本的配置文件，如此即可更新配置文件，开启匿名登录的功能。

范例 13-2-1，只能开启匿名登录，其实，做法还可以更有弹性一点，例如：执行时，加上选项 on 或 off，即可切换：开启或关闭匿名登录。

范例 13-2-2：anonyftp.sh

```
01.    #! /bin/Bash
02.
03.    # 修改 vsftpd 的配置文件，切换：'开启/关闭' 匿名 FTP 服务
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 '选项格式错误！请用 on 或 off 来切换匿名登录的开关。'
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
```

说明

- 行 5～8，检查用户提供的选项个数是否刚好。
- 行 10，变量 OPT 存放选项值。
- 行 11，使用 case 语法，判断选项是 on 或 off。
- 行 12~13，利用字符集合，让选项不分大小写，均可接受。若选项是 on，CMD 变量值就设为 YES；若是 off，就设为 NO。CMD 变量，会放在行 23，作为 anonymous_enable 的设定值。
- 行 14～17，如果选项不是 on 或 off，就显示选项错误的信息，然后 exit 离开 Script。
- 行 20～24，同范例 13-2-1 的说明。

使用方法：

- 要想关闭匿名登录，请执行：

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

- 如想开启匿名登录，请执行：

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

当然，要让设定生效，得重新启动 vsftpd 才行。在 B2D Server 中，做法很简单，如下所示：

```
service vsftpd restart
```

或

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

13.3 awk 的用法

awk 是一种可以处理数据、产生格式化报表的语言，功能相当强大。awk 的工作方式是读取数据文件，将每一行数据视为一条记录（record），每笔记录以字段分隔符分成若干字段，然后输出各个字段的值。

以下是执行 `ps auxw` 的输出片段：

```
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]
```

像这种固定结构的数据，用 awk 来处理，特别有威力，通常只要短短几行程序代码就可以完成工作。

例如，仅用以下单一指令，就可取得所有行程的 PID：

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

那么，awk 是如何处理每一笔记录的呢？

awk 对每一条记录，都会套用一个“样式{操作}”，如果该行符合样式，就执行指定的操作。样式或操作之一，可以省略。如果只有样式，表示要显示符合样式的数据行；如果只有操作，表示对每一数据行都执行该项操作。

以下是 awk 常用的作用格式：

- awk “样式” 文件：把符合样式的数据行显示出来。
- awk '{操作}' 文件：对每一行都执行{}中的操作。

- awk '样式{操作}' 文件：对符合样式的数据行，执行{}中的操作。

用例

- awk 的用法 1：

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

显示含 La 的数据行。

- awk 的用法 2：

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

显示 dataf3 每一行的第 1 和第 2 个字段。

$1 代表第 1 个字段，$2 代表第二字段，其他类推。

- awk 的用法 3：

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

将含有 La 关键词的数据行的第 1 及第 2 个字段显示出来。

- awk 的用法 4：

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

使用选项-F，指定:为分隔字符，账号 ols3 的 uid（第 3 字段）及 gid（第 4 字段）显示出来。

- awk 的用法 5：

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

以:为分隔字符，+++为输出字段分隔符，将账号 ols3 的第 1~5 栏显示出来。

▶执行结果：

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

本例中，BEGIN{}区域指示 awk 一开始先做初始化的操作，即设定 OFS="+++"。变量 OFS 的作用是存储输出字段的分隔符。接着，寻找 ols3 的账号行，找到后，使用 print 印出第 1 ~ 第 5 个字段，且彼此用 +++ 隔开。

实例应用

- 取得网卡的 IP：

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

- 取得网络设备名称：

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

在本例中，-F:把分隔字符设为:，而且，采用多选一的样式 /eth.:|ppp.:|wlan.:/。这个样式的意思是：设备名称可以是 eth0:、ppp1:、wlan1: 这 3 个其中之一。一旦找到符合样式的字符串后，去掉:，取其中的第一个域值，因此，可能的答案是 eth0 或 ppp1 或 wlan1。

- 取得系统内存大小：

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

/proc/meminfo 记载主机内存相关数据，其中 MemTotal 为内存大小，其样本值如下：

```
MemTotal:       223128 kB
```

因此，在 awk 的样式语法中，利用/MemTotal/ 找到这一行，再印出第二个字段，即可得到内存的大小。

- 修改 CSV 文件各字段的顺序：

以下是数据文件 dataf6.csv，想要把第 2 个字段和第 4 字段调换：

```
所在乡镇,学校名称,学校网址,校长姓名,学校电话,VOIP 前三码,学校地址
新营市,南新国中,http://www.ns12jh.tnc.edu.tw,ABC,06-656313012,1021,新营市民治路 6675 号
佳里镇,佳里国中,http://www.jl41jh.tnc.edu.tw,NOP,06-722224432,1146,佳里镇安南路 5523 号
新营市,新营国小,http://www.sy53es.tnc.edu.tw,DEF,06-632213642,1482,新营市中正路 3248 号
```

做法如下：

范例 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
```

说明

- 行 3，设定临时文件名。
- 行 4，将数据文件的内容通过管道交给 awk 处理。awk 的字段分隔字符和输出分隔字符，皆设为,。在{}的操作中，调换$2 和$4 的顺序，再把结果转向存储在临时文件中。
- 行 5，将临时文件覆盖原文件。