AWK用法1

awk是一个文本处理工具，可以对数据进行处理。awk可以进行正则表达式匹配、样式装入、流控制、数学运算符、进程控制语句甚至于内置的变量和函数。简单的说AWK是一种用于处理文本的程序语言。其名称来自于它的创始人 Alfred Aho, Peter Weinberger, 和 Brian Kernighan 名字的首字母。gawk是awk的GUN版本。

awk 处理文本过程

awk 从文件或者标准输入读取内容 按行 对文本 循环 进行处理，每一行循环一次。每一行又被指定的分隔符号(默认是空格符号)分割成一系列的域或字段，此外，awk 针对每一行进行模式匹配，符合模式的行进行后续的处理操作，默认情况下将处理后的结果输出。

awk 语法格式：

```
1 | awk [option] 'BEGIN{}/pattern/{action}END{'  
   | FILENAME
```

给出 awk 的语法格式，我们可以对 awk 处理过程更进一步的进行说明。awk 从文件或者标准输入读取内容之前，首先进行 BEGIN{} 所定义的操作，例如可以在 BEGIN{} 定义变量，格式化输出表头等操作。随后，awk 解释器对读入的内容进行模式匹配，模式匹配可以是表示式，也可以是正则表达式匹配，随后被 pattern 匹配到的行被送到后续的操作步骤 action，action 中可以使用表达式、控制语句、输入输出语句、循环、甚至使用 awk 的内置函数，切记上述过程除 BEGIN只是在开始工作之前执行一次外，action 循环对每一行都进行处理，直到内容结束。处理完所有的行之后，awk 执行 END 操作。接下来，我们通过示例对awk的用法进行详细的说明。

awk用法

常用选项

awk 的常用选项有：

`-F "CHAR"` 指定输入分割符号。默认空格作为输入分割符

`-v VARNAME=VAR` 自定义变量

```
1 # 示例1 取出 /etc/passwd 文件中的用户名和UID
2 $ awk -F ":" '{print $1,$3}' /etc/passwd
3 root 0
4 bin 1
5 daemon 2
6 adm 3
7 lp 4
8 sync 5
9 ...
```

awk 内置变量

`$N` N是一个数字，awk 按指定或者默认的分割符号对行进行分割，\$N 表示第几段

`NF` Number of Field, \$NF行的最后一个字段。NF 表示字段的总个数

`FS` 输入字段分割符，可以对默认的空格分割符进行重新赋值

`RS` 输入行分割符

`OFS` 输出字段分割符，默认是空格

`ORS` 输出行分割符

`NR` The Number of input records, 总共处理的行记录个数。如果有多个文件，则一起计数

`FNR` 对单个文件的行分别进行计数

```
1 # 示例2 取出 /etc/passwd 文件以冒号分割的最后一个字段
2 $ awk -F ":" '{print $NF}' /etc/passwd
```

```
1 # 示例3 取出 /etc/passwd 的用户名，每个用户名和UID，字段间
   用冒号分割，处理后的行以空格隔开
2 $ awk -v FS=":" -v ORS=" " -v OFS=":" '{print $1,$3}'
   /etc/passwd
3 root:0 bin:1 daemon:2 adm:3 lp:4 sync:5 shutdown:6
   halt:7 mail:8 operator:11 games:12 ftp:14 nobody:99
   systemd-bus-proxy:999 systemd-network:192 dbus:81
   polkitd:998 tss:59 postfix:89 sshd:74 ntp:38
   tcpdump:72 apache:48 mysql:27 xuekaixin:1000
```

```
1 # 示例4 对文件的行进行计数，显示行号
2 $ awk '{print NR,$0}' /etc/passwd /etc/fstab
3 1 root:x:0:0:root:/root:/bin/bash
4 2 bin:x:1:1:bin:/bin:/sbin/nologin
5 ...
6 25 xuekaixin:x:1000:1000::/home/xuekaixin:/bin/bash
7 26
8 27 #
9 28 # /etc/fstab
10 32 # See man pages fstab(5), findfs(8), mount(8)
    and/or blkid(8) for more info
11 ....
12 34 UUID=a7f36c4e-a6e8-4b09-9bd9-99f8308db592 /
    ext4 defaults 1 1
13 35 UUID=d20d68ed-a2cd-4878-84fb-73808b91662e /boot
    ext4 defaults 1 2
```

awk操作符及条件判断

awk 支持的操作符有算术运算操作符、赋值操作符、布尔型、比较操作符、逻辑操作符。

算术操作符: `**` 平方运算 `*` 乘法 `%` 取模运算 `/` 除法

赋值操作符: `=` `+=` `-+` `*=` `/=` `^=` `**=` `++` `--`

- %S 显示字符串
- %u 显示无符号整数
- %% 显示百分号本身

修饰符:

- N 指定显示宽度
- 左对齐
- + 显示数值符号

```

1 # 示例8 显示系统普通用户和系统，并进行格式化输出
2 $ awk -F : 'BEGIN{printf "%-17s
   %s\n%s\n", "UserName", "Identity", "=====
   ====="}{if ($3==0) {printf "%-17s
   %s\n", $1, "root user"} else {printf "%-17s
   %s\n", $1, "common user"}}' /etc/passwd
3 UserName          Identity
4 =====
5 root              root user
6 bin               common user
7 daemon            common user
8 adm                common user
9 lp                 common user
10 sync              common user
11 .....

```

```

1 # 示例9 统计UID大于500的用户个数
2 $ awk -F : -v sum=0 '{if ($3>=500) {sum++}}END{printf
   "%s %d\n", "Number of users:", sum}' /etc/passwd
3 Number of users: 3

```

awk 循环

同大多数编程语言一样，awk 也支持循环。在 awk 由于对文件内容默认情况下是按行进行循环处理，所以其本身就内置对行的循环。此处的循环指的是对行中分割后的字段进行循环处理。同时，awk 也支持数组的使用。

awk 中常用的循环及语法格式：

```
while (condition) {statement1; statment2; ...}
```

```
do {statement1, statement2, ...} while (condition)
```

```
for ( variable assignment; condition; iteration
```

```
process) { statement1, statement2, ...}
```

```
1 # 示例10 遍历数组循环打印字符串
2 $ awk 'BEGIN{a[0]="xue";a[1]="jin";a[2]="wei";for
   (i=0;i<=2;i++) {print a[i]} }'
3 xue
4 jin
5 wei
```

```
1 # 示例11 统计当前用户使用shell的个数
2 $ awk -F: '$NF!~/^$/{BASH[$NF]++}END{for(A in BASH)
   {printf "%15s:%i\n",A,BASH[A]}}' /etc/passwd
3     /bin/sync:1
4     /bin/bash:2
5     /sbin/nologin:20
6     /sbin/halt:1
7     /sbin/shutdown:1
```

说明： `$NF!~/^$` 表示对行中的最后一个字段不为空的字段匹配；
`{BASH[$NF]++}`，定义一个BASH的关联数组，以行中最后一个字段的内容作为下表，每一次循环，匹配到下标为相同的内容时对BASH数组的值加1，当处理完成后，BASH数组中保存的数值就是不同类型bash的值，当对每一行处理完成后，循环遍历BASH的下标，然后进行打印输出。

awk 应用

解决DOS 攻击生产案例：根据web日志或者或者网络连接数，监控当某个IP 并发连接数或者短小时内 PV 达到100，即调用防火墙命令，添加防火墙规则。并且定时进行监控，并将监控的情况记录到日志。

```

1 # 监控脚本
2 #!/bin/bash
3 IP=$(netstat -tnp | awk '/^tcp.*\
  <TIME_WAIT\>/{split($5,array,":");status[array[1]]++
  }END{for(i in status) {printf "%-13s
  %d\n",i,status[i]}}' | awk '{if ($2>100) {print
  $1}}'
4 if [ -z "$IP" ]; then
5     echo "No DOS attack." >> /root/DOS.log
6 fi
7 for a in `echo "$IP"`; do
8     iptables -A INPUT -s "$a" -j REJECT
9     echo "$a Add to the firewall!" >> /root/DOS.log
10 done
11 # 添加 crontab 计划任务
12 * * * * * for i in {1..2}; do /works/scripts/DOS-
  NO.sh >> /root/DOS.log; sleep 30;done

```

说明: `/^tcp.*\<TIME_WAIT\>/` 匹配tcp连接且连接状态为TIME_WAIT的连接记录。 `{split($5,array,":");` 将第五个字段分割，只取IP地址内容，并将分割后的内容保存至array数组中，`status[array[1]]++` 对不同IP地址进行计数统计。 `END{for(i in status) {printf "%-13s %d\n",i,status[i]}}` 遍历关联数组status，并进行输出。 `awk '{if ($2>100) {print $1}}'` 如果同一个IP连接数大于100，则进行输出，并保存至IP变量中。

随后对IP进行判断，如果IP不为空，则进行添加防火墙规则。cron进程每30秒执行一次脚本。

更多参考资料

[GUN awk手册](#)

返校两天，时间来不及了，下一节总结awk的case语句，循环中使用break和continue以及next关键字的使用和awk内置函数自定义函数。