Linux 操作系统及应用
附录 A — AWK

李亦农　唐晓晟
hoplee@bupt.edu.cn　txs@bupt.edu.cn

Beijing University of Posts and Telecommunications (BUPT)
School of Information and Communication Engineering

内容简介 I

内容简介 II

7. 其他
8. 限制
9. 实例

概述 I

- awk是一种编程语言，她是由 AT&T 贝尔实验室的 Alfred Aho, Peter Weinberger 和 Brian W. Kernighan 开发的，Brian W. Kernighan 目前仍在维护及增强awk。
- awk能非常方便地处理与数据加工和信息检索相关的任务。
- awk还可以和shell程序相互结合，从而增强自己的功能。
- 一个awk程序是由一系列的“模式—动作”语句构成的：

```
pattern {action}
pattern {action}
pattern {action}
......
```

概述 II

- awk程序为每个输入行依次地进行每一个模式的匹配寻找，对每一个匹配上的模式执行相应的动作，接着读取下一行并再次开始匹配，直到所有的输入都处理完毕。
- 在一条语句中可以省略模式或者动作，缺省的模式为匹配所有行，缺省的动作为输出当前行：`print $0`
- 正是因为模式和动作可有可无，动作才要被花括号括起来以便与模式相区别。
- awk的程序可以以两种方法来运行：
  1. 直接命令行方式：
     ```
     awk 'pattern-action statements' input_file_list
     ```
     其中“模式—动作”语句必须括在单引号中。
  2. awk命令程序方式：
     通常将awk程序放在一个单独的文本文件中，然后用-f选项来引用：
     ```
     awk -f awk_file input_file_list
     ```

概述 III

- awk从输入中一次读取一行（一条记录），缺省的RS为\n。
- 然后awk将记录分割为一个个的字段，缺省的FS为Blank。一行中的第一个字段称为$1，第二个字段称为$2，…，整个记录称为$0。
- awk提供了print和printf语句用于显示输出。
  - print为无格式输出语句：
    print expr1,expr2,...,exprN
    print语句显示每个表达式的串值，默认的ORS和OFS分别为\n和Blank。
  - printf为格式化输出语句：
    printf format,expr1,expr2,...,exprN
    format含有要显示的信息以及要转换的规格说明，其语法与 C 语言中的输出函数中的格式说明类似。

概述 IV

- 常用的转换字符如下表所示：

%c	单个字符
%d	十进制数
%e	[-]d.ddddddE[+|-]dd
%f	[-]d.dddddd
%g	e或f中较短的，并去掉无用的 0
%o	无符号八进制数
%s	串
%x	无符号十六进制数
%%	显示一个%

概述 V

- 使用printf时，不会自动输出ORS，必须自己在format中使用\n来显式地产生。

变量 I

- awk的变量分为内部变量和用户定义的变量两种。
- awk的内部变量有：

ARGC	命令行参数的个数
ARGCIND	当前命令行参数下标
ARGV	命令行参数数组
ENVIRON	环境变量数组
FILENAME	当前输入文件名
FNR	当前文件中的记录号
FS	字段分隔符
IGNORECASE	忽略正则表达式和串的大小写
NF	当前记录中的字段数
NR	至今读取的记录数
OFMT	数的输出格式，缺省为%.6g

变量 II

OFS	输出字段分隔符
ORS	输出记录分隔符
RS	输入记录分隔符
RSTART	由match()匹配的第一个字符的索引
RLENGTH	由match()匹配的串的长度
SUBSEP	下标分隔符，缺省为\034

变量 III

- 当前记录的字段可以用`$1`, `$2`, ..., `$NF`来表示。
- 用户定义的变量类似于 Shell 中的情形。

模式 I

- 模式是一种表达式。
- BEGIN和END是两个特殊的模式，BEGIN在第一条记录被读取之前匹配，END在最后一条记录处理完之后匹配。
- 关系表达式：awk有 6 个关系运算符和 2 个正则表达式匹配运算符：

<	小于
<=	小于等于
==	等于
!=	不等于
>=	大于等于
>	大于
~	匹配
!~	不匹配

模式 II

- 在比较表达式中，若两个操作数都是数值，则进行数值比较，否则进行串比较。
- 类型强制转换：num，""，string+0
- 正则表达式：awk把在“~”和“!~”右边的任一串或变量都解释为一个正则表达式。
- 当用引号括起来的字面字符串用作一个正则表达式时，如果字符串中含有元字符，需要再加一层反斜线，以保护正则表达式中的元字符。
- awk里引入了一个新的概念：字符类。这个概念来自于 **POSIX** 标准。
- 字符类是一种特殊的表达式，用于描述具有某种特定属性的字符集合。她具体能表示什么样的字符集合是和应用程序的地区特性有关的。

模式 III

- 字符类只能在正则表达式中的一对方括号中出现，用于表示特定的字符集合。
- 常用的字符类有：

[:alnum:]	字母或数字字符
[:alpha:]	字母字符
[:blank:]	空格或制表符
[:cntrl:]	控制字符
[:digit:]	数字字符
[:graph:]	既能看见又能打印的字符
[:lower:]	小写字母字符
[:print:]	可打印字符（非控制字符）
[:punct:]	标点符号字符
[:space:]	空白字符，包括空格、制表、换页等
[:upper:]	大写字符
[:xdigit:]	十六进制数字字符

模式 IV

- 模式组合：用圆括号和逻辑运算符||、&&和! 可以把简单的模式组合成复合模式。优先级从左到右依次增高。
- 模式范围由逗号分隔的两个模式组成：

  `pattern1,pattern2 {action}`

  表示对于在`pattern1`和`pattern2`出现之间的每一条记录都要执行动作。包括`pattern1`和`pattern2`。

动作 I

- 动作决定对模式选中的记录进行什么操作。
- awk的运算包括算数运算和串运算。
- awk可以使用传统的算数表达式来计算数值，算数运算在内部以浮点形式完成。常用的算数运算符有：
  +,-,*,/,++,--,%,^,+=,-=,*=,/=,%=,^=和=
  其含义和 C 语言里一样。
- awk还提供了一些内部算数函数：atan1(y,x), cos(x), exp(x), int(x), log(x), rand(), sin(x), sqrt(x)和srand(x)。其中x,y是任意表达式。rand()返回 (0,1) 范围内的随机数，srand(x)用于设置rand()的种子。
- awk提供的串运算符只有一个：space，进行串的串接。

动作 II

- awk提供的串函数有：
  其中，r代表一个正则表达式，s和t代表串表达式，n和p代表整数，a代表数组。

gsub(r,s)	将当前记录中的r替换为s，全局，返回替换数
gsub(r,s,t)	在串t中全局用s替换r，返回替换数
index(s,t)	返回s中串t的位置，不出现时为 0
length(s)	返回串s的长度
match(s,r)	返回r在s中出现的位置，不出现时为 0
split(s,a)	利用FS把s分裂成数组a，返回字段数
split(s,a,r)	利用r把s分裂成数组a，返回字段数
sprintf(fmt,exprs)	根据格式串fmt，返回经过格式编排的expr_list
sub(r,s)	在当前记录中把第一个r替换成s之后的部分，返回替换的个数

动作 III

sub(r,s,t)	在t中把第一个r替换成s之后的部分
substr(s,p)	返回从位置p开始的s之后的部分
substr(s,p,n)	返回从位置p开始，长度为n的s的子串
tolower(s)	将串s中的大写字母改为小写
toupper(s)	将串s中的小写字母改为大写

动作 IV

- 流控：awk提供和 C 语言相似的流控语句:

```
1  if (condition) statement [ else statement ]
2  while (condition) statement
3  do statement while (condition)
4  for (expr1; expr2; expr3) statement
5  for (var in array) statement
6  break
7  continue
8  delete array[index]
9  delete array
10 exit [ expression ]
11 { statements }
```

I/O 语句 I

- awk的输入、输出语句如下所示：

close(file)	关闭文件（或管道）
getline	从当前输入、文件或管道中读取下一个输入记录，并进行通常的字段分裂处理，同时设置NF, NR和FNR
getline < file	从文件中getline
getline var	读取下一条记录并将其赋给var，同时设置NR和FNR
getline var<file	从文件中读取下一条记录并将其赋给var，同时设置NR和FNR
next	立即从第一个模式开始处理下一条记录
nextfile	停止处理当前输入文件，立即处理下一输入文件。FILENAME和ARGIND被更新，FNR被置为 1

I/O 语句 II

`system(cmd-line)`	执行 Shell 命令`cmd-line`并返回命令的退出状态
`fflush([file])`	刷新输出文件的缓冲区

I/O 语句 III

- 其他的 I/O 重定向也可在awk中使用。对于print和printf语句，>>file会将输出追加到file文件中，|会将输出送到管道中。同样的command | getline将会使得getline函数从管道中读取数据。
- 当遇到文件结束时，getline将返回 0，任何错误将导致getline返回-1。

函数

- 用户可以自己定义函数，语法为：

```
1 function function_name(arg_list){
2     statements
3 }
```

- 数组参数可通过引用传递，标量参数将用值传递。
- 在函数内部的形式参数是局部变量，其他变量都是全局变量。
- return语句可有可无。
- 函数在调用时在函数名与实参表的左括号之间不得留有空格。因为空格是字符串串接运算符。

其他 I

- awk提供一维数组。数组和数组元素无须声明，通过使用它来表明它的存在。
- 数组下标可以是一个数或串：arr[x]和arr["x"]
- 可以使用for语句对数组所有定义了的下标进行循环：
  for (i in arr) statement，此时下标是随机选取的。
- 可以用delete语句删除数组元素：
  delete array_name[subscripts]
- 时间函数：

systime()	返回从Epoch到现在的秒数
strftime([format[, timestamp]])	返回用format格式化后的timestamp指定的时间（缺省为当前时间）

其他 II

- 任何以#开头的行都是注释。
- 几个语句可以用分号分隔以便出现在同一行上。
- 续行符为\。

限制

- 100 个字段
- 每个输入记录 2500 个字符
- 每个输出记录 2500 个字符
- 每个单个字段 1024 个字符
- 每个`printf`串 1024 个字符
- 括起来的串 400 个字符
- 字符类 400 个字符
- 15 个打开的文件
- 1 个管道
- 数值的大小由本地机器限制

实例 I

1. 此程序会显示所有输入行之中字段的最大个数。

```
1 awk '{if (NF > max) max = NF}
2 END {print max}'
```

2. 此程序会显示出超过 80 个字符的每一行。此处只有模式被列出，动作是采用缺省值显示整个记录。

```
1 awk 'length($0) > 80'
```

实例 II

3. 显示拥有至少一个字段的所有行。这是一个简单的方法，将一个文件里的所有空白行删除。

```
1 awk 'NF > 0'
```

4. 此程序会显示出范围是 0 到 100 之间的 7 个随机数。

```
1 awk 'BEGIN {for (i = 1; i <= 7; i++)
2 print int(101 * rand())}'
```

实例 III

5. 此程序会显示出所有指定的文件的总字节数。

```
1 ls -l files | awk '{x += $5};\
2   END {print "total bytes: " x}'
```

6. 此程序会将指定文件里最长一行的长度显示出来。expand会将TAB改成SPACE，所以是用实际的右边界来做长度的比较。

```
1 expand file | awk '{if (x < length()) x = length()}
2 END {print "maximum line length is " x}'
```

实例 IV

7. 将所有用户的登录名称，依照字母的顺序显示出来。

```
1 awk 'BEGIN {FS = ":"}
2 {print $1 | "sort"}' /etc/passwd
```

8. 此程序会将一个文件的总行数显示出来。

```
1 awk '{nlines++}
2 END {print nlines}'
```

实例 V

9. 此程序也会将一个文件的总行数显示出来，但是计算行数的工作由awk来做。

```
1 awk 'END {print NR}'
```

10. 此程序显示出文件的内容时，会在每行的最前面显示出行号，它的功能与cat -n和nl命令类似。

```
1 awk '{print NR, $0 }'
```

The End

The End of Appendix A.