AWK

João Marcelo Uchôa de Alencar

Universidade Federal do Ceará - Quixadá

24 de Outubro de 2019

O que é AWK?

- O *awk* é um comando que foi criado para englobar as capacidades do *sed* e do *grep*.
- A operação básica é pesquisar um conjunto de linhas de entrada, uma após a outra, procurando as que satisfaçam a um conjunto de **padrões ou condições especificadas** por você.
- Para cada padrão você pode especificar uma **ação**:
- [padrão] [{ ação }]

Exemplo:

```
awk '$1 == "Paula" {print $2, $3}'
```

A instrução mostrada imprime o segundo e terceiro campos de cada linha entrada a partir do teclado cujo primeiro campo seja Paula. Veja a sutil diferença entre o uso de $1, $2 e $3 no *awk* e no *shell*.

Uso do *awk*

Diretamente a partir do teclado:

```
awk '{<padrão e ação>}' [arquivo1] ... [arquivoN]
```

A partir de um arquivo de comando:

```
awk -f <arquivo de programa> <lista de arquivos>
```

Por padrão, o separador é o TAB ou espaço em branco.
A variável IFS não tem efeito no *awk*.
Os **padrões** se encaixam em três categorias:

- Expressões Relacionais (comparação)
- Expressões Regulares
- BEGIN e END

Expressões Relacionais

Operadores	Resultados
==	Igual a
>	Maior que
>=	Maior ou igual
<	Menor que
<=	Menor ou igual
&&	e
||	ou
!	Não

```
$ awk '$1 > "J" { print }' ../emailsordenados.txt
$ awk '$1 > "J" && $2 < "E" { print }' ../emailsordenados.txt
```

Expressões Regulares

Devemos colocar a expressão na parte do padrão, entre //

```
$ awk '/^C/ { print }' ../emailsordenados.txt
$ awk '$2 ~ /^C/ { print }' ../emailsordenados.txt
$ awk '$1 !~ /^C/ { print }' ../emailsordenados.txt
$ awk '$1 ~ /^[AB]/ { print }' ../emailsordenados.txt
$ awk '$1 ~ /^(ANDRE|ANA)/ { print }' ../emailsordenados.txt
```

Padrões BEGIN e END

- Padrão BEGIN: processamento anterior à entrada. Exemplo: cabeçalho.
- Padrão END: processamento posterior à saída. Exemplo: totais e médias.

```
$ awk 'BEGIN { print "Nome Completo" } { print }' ../emailsordenados.txt
$ awk 'END { print "Fim da Lista"} { print }' ../emailsordenados.txt
```

Variáveis

Variável	Significado
ARGC	Número de argumentos
ARGV	Vetor com parâmetros
FILENAME	Nome do arquivo
FMR	Número do registro
FS	Separador
NF	Número de campos
NR	Quantidade de registros
OFMT	Formato para números
OFS	Separador na saída
ORS	Separador de registros
RS	Separador na entrada

Variáveis

```
$ awk 'BEGIN { print "Nome do Alunos" }
> { print }
> END { print "Entradas=", NR }' ../emailsordenados.txt
awk '$1 ~ /^J/ { print; Soma=Soma+1 }
> END { print Soma}' ../emailsordenados.txt
$ awk '
> BEGIN { printf "%15s %5s\n", "IP", "Latência" }
> { printf "%15s %5s\n", $1, $2, LatenciaTotal = LatenciaTotal + $2 }
> END { printf "\n Latência média: %5s\n",
> LatenciaTotal / NR } ' ips_latencia.txt
```

Operadores e Funções Matemáticas

Operadores	**Significado**
+	Soma
-	Diferença
*	Multiplicação
/	Divisão
%	Resto da Divisão
**	Exponenciação
=	Igualdade
Função	**Significado**
sqrt(x)	Raiz quadrada
sin (x)	Seno
log (x)	Logaritmo
rand (x)	Aleatório entre 0 e 1

Funções de Cadeia de Caracteres

Função	**Descrição**
index(c1, c2)	Posição de c1 em c2
length(c1)	Comprimento da cadeia
match(c1, exp)	Posição de c1 onde exp ocorre
split(c1, v [,c2])	Divide c1 baseado em c2
sprintf(fmt, lista)	Retorna lista formatada
sub(exp, c1 [, c2])	Substitui exp em c2 por c1
gsub(exp, c1 [, c2])	Toda exp em c2 é substituída por c1
substr(c1, p, n)	Retorna subcadeia em p de c1 com n caracteres

Funções de Cadeia de Caractere

```
$ awk 'BEGIN { print index("LINUX-UNIX", "X")}'
$ awk 'BEGIN { print length("LINUX-UNIX")}'
$ awk 'BEGIN { print match("LINUX-UNIX", /[UI]X/)}'
$ awk 'BEGIN { print match("LINUX-UNIX", /[UI]X$/)}'
$ awk 'BEGIN { OS="LINUX-UNIX" sub(/LINUX/, "BSD", OS) print OS}'
$ $ awk 'BEGIN {
> OS="LINUX-UNIX"
> sub (/LINUX/, "BSD", OS)
> print OS
}'
```

Exemplo

- Dado o arquivo *emailsordenados.txt*, você conseguiria, usando o *awk*, dizer quantos alunos tem o e-mail começando com a mesma letra do primeiro nome? E com a primeira letra do segundo nome? Considere maiúsculas e minúsculas iguais.
- Lembrando a saída do comando *ps aux*, faça agora um *script* usando o *awk* para contar a quantidade de memória virtual usada por determinado programa. Será que ficou mais fácil que usar *grep* e *sed*?

O comando *if*

```
if (expressao)
   {
      comando1
      comando2
      ...
      comandoN
   }
else
   {
      comando1
      comando2
      ...
      comandoN
   }
```

O comando *while*

```
while (expressao)
    {
        comando1
        comando2
        ...
        comandoN
    }
```

O comando *for*

```
for (expressao1; expressao2; expressao3)
    {
        comando1
        comando2
        ...
        comandoN
    }
```

Saída dos Laços

- **break**: desvia para primeira instrução após o laço
- **continue**: avança para a primeira instrução da próxima interação do laço
- **next**: pula para o próximo registro
- **exit**: considera que o arquivo terminou, executa apenas ação END

O comando *for*

```
$ awk '{ Registro [NR] = $0 }
> END { print Registro [2] }' ../emailsordenados.txt
```

Os vetores também pode ser indexados por *strings*.

Saída com *printf*

printf formato, expressao1, expressao2, ... , expressaoN

Letra	Impressão
c	Caractere
d	Decimal
e	Exponencial
f	Ponto flutuante
g	Ponto flutuante compacto
o	Octal
s	*String*
x	Hexadecimal
%	O símbolo

Redirecionando a Saída

```
$2 > 7.0 { print $1, $2 > "aprovados.txt" }
$2 > 5.0 { print $1, $2 > ("/tmp/" ARQUIVO ) }

{
   PrimeiraNota[$1] = $2
}
END {
   for (aluno in PrimeiraNota)
      print aluno, "\t" , PrimeiraNota[aluno] | "sort"
}
```

Acessando Parâmetros

```
'BEGIN {
for (i = 1; i < ARGC; i++)
   print ARGV [i], "\n"
}' a b c
```

Exemplo

Faça um *script* marajas.awk que dado um arquivo de salários no formato abaixo imprima os professores que mais ganham por curso.

```
$ cat professores.txt
NOME       CURSO       SALARIO
Jeandro    Redes         10000
Elvis      Engenharia    11000
Hélder     Engenharia    15000
João       Redes          1000
Michel     Redes           800
Jefferson  Sistemas       5000
Márcio     Sistemas       6000
Marcos     Redes         11000
$ awk -f marajas.awk professores.txt
Engenharia: Elvis,  11000
Redes:      Marcos, 11000
Sistemas:   Márcio,  6000
```

Atividade

Coloque a resposta em *atividades/atividade10*. Faça um *script* chamado *contaPalavras.awk* que ao ser invocado com o nome de um arquivo de texto diga quantas vezes cada palavra aparece no texto.

```
$ cat arquivo.txt
a casa que vivo é boa.
boa casa é.
$ awk -f contaPalavras.awk arquivo.txt
Relatório de palavras.
casa: 2
boa:  2
é:    2
a:    1
que:  1
vivo: 1
Total de Palavras analisadas: 9.
```

Atividade

Considere o seguinte arquivo *notas.txt*

```
Aluno:Nota1:Nota2:Nota3
João Marcelo:10.0:10.0:10.0
Alisson Barbosa:1.0:2.0:3.0
Jeandro Bezerra:7.0:8.0:9.0
Marcos Dantas:5.0:4.0:6.0
Michel Sales:3.0:4.0:2.0
```

Coloque na pasta *atividades/atividade10* um *script* chamado *disciplina.awk* que realize a seguinte ação:

```
$ awk -F: -f disciplina.awk notas.txt
Aluno:Situação:Média
João Marcelo:Aprovado:10.0
Alisson Barbosa:Reprovado:2.0
Jeandro Bezerra:Aprovado:8.0
Marcos Dantas:Final:5.0
Michel Sales:Reprovado:3.0
Média das Provas: 5.2 5.6 6.0
```