(G)AWK

O Que é?

- Um acrónimo dos últimos nomes dos 3 autores;
- Linguagem e processador de padrões;
- Mais avançado que o sed, menos complicado que o C, menos críptico que o Perl;
- Gawk, nawk.

awk syntax

- awk [-Ffield_sep] ‘cmd’ infile(s)
- awk [-Ffield_sep] –f cmd_file infile(s)
- <infile> pode ser a saída de um *pipe: |*
- O espaço é o field_sep por omissão

Como funciona

- [pattern] [{action} …]
- Os ficheiros de entrada são processados linha a linha;
- Se nenhum padrão for especificado todas as linhas são processadas;
- As linhas são separadas em campos atendendo ao valor de field_sep;
- Print é a ação por omissão;
- Os ficheiros de entrada não são afetados pelo processamento.

Nomes dos campos

- As linhas são partidas em campos pelo valor de field_sep;
- $0 representa a linha inteira;
- $1, $2, …, $n representam os vários campos;
- Os nomes dos campos podem ser usados como variáveis em expressões, testes, etc.

Variáveis internas

Variável	Descrição
FS	Separador de campo; ‘ ‘ como valor por omissão
NR	Número de linhas processadas até ao momento
NF	Número de campos da linha corrente
FILENAME	Nome do ficheiro de input corrente
OFMT	Formato por omissão para output de números
OFS	Separador de campo na saída; ‘ ‘ como valor por omissão
ORS	Separador de registo na saída; ‘\n’ como valor por omissão
RS	Separador de registo na entrada; ‘\n’ como valor por omissão
FNR	Igual a NR só que é colocado a 0 sempre que o ficheiro de entrada muda
RSTART, RLENGTH	Variáveis atualizadas pela função match() e que indicam a posição onde começou o match e qual o seu comprimento em caracteres
SUBSEP	Separador subscript, usado em arrays multi-dimensionais

Operadores

Operador	Descrição
+, -, *, /	Adição, subtração, multiplicação e divisão
%	Resto da divisão inteira
++	Incremento
--	Decremento
^ ou **	Exponenciação
+=, -=, *=, /=, %=	Atribuição precedida da operação indicada
Sem operador	Concatenação de strings: nova = “teste” $3 “de” $5
?:	Operador ternário condicional: expr1? expr2 : expr3

Operadores relacionais

Operador	Descrição
==	Igualdade
!=	Desigualdade
<	Menor
<=	Menor ou igual
>	Maior
>=	Maior ou igual
~	Contem expressão regular
!~	Não contem expressão regular

Padrões

- Podem fazer match com linhas individuais ou grupos de linhas;
- Padrões especificados com expressões regulares;
- Padrões especificados com expressões relacionais;
- Padrões especiais: BEGIN e END.

Expressões Regulares

Meta carácter	Descrição
.	Qualquer carácter exceto o '\n'
*	0 ou mais ocorrências do carácter precedente
^	Início de linha: ^A – linhas iniciadas por A
$	Fim de linha: :$ - linhas terminadas em :
\	Carácter de escape: \., *, \[, \\, etc
[]	Classes de caracteres: [aeiou], [a-z], [a-zA-Z], [0-9], [a-z?,!]
[^]	Qualquer carácter excepto os especificados: [^13579] – digito par
\<. \>	Caracteres no início ou fim de palavras

Expressões regulares (extensão)

Meta carácter	Descrição
|	Alternativa: ho(use|me)
+	1 ou mais ocorrências do carácter precedente
?	0 ou 1 ocorrência do carácter precedente
{n}	n repetições do carácter precedente ou grupo
{ n, m }	n a m repetições
(…)	Agrupamento

Expressões regulares (exemplos)

Exemplo	Descrição
.{10,}	10 ou mais caracteres
[0-9]{4}\-[0-9]{3}	Código postal português
[0-9]{3}[ ]*[0-9]{3}	Código postal na Índia
[0-9]{9}	Número de telefone nacional
[a-zA-Z][a-zA-Z0-9_.]*	Identificador numa linguagem de programação

Padrões com expressões regulares

Exemplo	Descrição
awk ‘/pat1/’ infile	Igual a: grep ‘pat1’ infile
awk ‘/pat1/,/pat2/’ infile	Imprime as linhas entre pat1 e pat2 ciclicamente
awk ‘/pat1/|/pat2/’ infile	Imprime as linhas que contêm pat1 ou pat2
awk ‘/pat1/.*/pat2/’ infile	Imprime as linhas que têm pat1 seguido de pat2 podendo ter ou não outros caracteres entre eles

Padrões com expressões relacionais

Exemplo	Descrição
awk ‘$1==“USA”’ infile	Imprime as linhas que têm “USA” no 1º campo
awk ‘$2!=“xyz”’ infile	Imprime as linhas que não contêm “xyz” no 2º campo
awk ‘$2 < $3’ infile	Imprime as linhas em que o 3º campo é maior que o 2º
awk ‘$5 ~ /USA/’ infile	Imprime as linhas em que o 5º campo contem “USA”
awk ‘$5 !~ /USA/’ infile	Faz o inverso do anterior
awk ‘NF == 5’ infile	Imprime as linhas com 5 campos
awk ‘NR == 5, NR == 10’ infile	Imprime as linhas da 5 à 10
awk ‘NR%5 == 0’ infile	Imprime as linhas de 5 em 5
awk ‘NR%5’ infile	Faz o inverso do anterior
awk ‘NF ~ /pat1/’ infile	Imprime a linha se o último campo contiver pat1

Padrões compostos

- Os padrões compostos são formados com operações booleanas e de domínio;
- pat1 && pat2 (AND – composto)
- pat1 || pat2 (OR – composto)
- !pat1 (Negação)
- pat1, pat2 (domínio)

Padrões compostos: exemplos

Exemplo	Descrição
awk ‘/pat1/ && $1==“str1”’ infile	?
awk ‘/pat1/ || $2>=10’ infile	
awk ‘!/pat1/’ infile	
awk ‘NF >= 3 && NF <= 6’ infile	?
awk ‘/pat1/ || /pat2/’ infile	
awk ‘/pat1/ , /pat2/’ infile	
awk ‘!/pat1|pat2/’ infile	?
awk ‘NR > 30 && $1 ~ /pat1|pat2/’ infile	

Padrões compostos: exemplos

Exemplo	Descrição
awk ‘/pat1/ && /pat2/’ infile	?
awk ‘/pat1.*pat2/’ infile	
awk ‘NR < 10 || NR > 20’ infile	
awk ‘!(NR >= 10 && NR <= 20’ infile	?
awk ‘NR == 10 , NR == 20’ infile	

Padrões BEGIN e END

- O padrão BEGIN permite especificar ações para serem executadas antes do processamento da primeira linha;
- O END permite especificar ações para serem executadas após todas as linhas terem sido executadas;
- São os dois opcionais.

BEGIN

- Deve ser usado para:
  - Inicializar variáveis;
  - Imprimir cabeçalhos;
  - Mudar o valor do seprador de campos:
    - awk ‘BEGIN {FS = “:”; print “File name: “, FILENAME} infile

END

- Deve ser usado para:
  - Realizar cálculos finais;
  - Imprimir rodapés;
  - Fazer o que seja preciso após as linhas terem sido processadas:
    - awk ‘END {print NR}’ infile

Ações

- Uma ação consiste numa ou mais instruções separadas por ‘;’, ‘\n’ ou ‘}’;
- Tipos de instruções:
  - Atribuição;
  - Control;
  - Escrita.

Instruções de Controlo

Instrução	Descrição
if (cond) { statlist1 } [else { statlist2 }]	
while (cond) { statlist }	
for (int_exp; cond_exp; ctrl_exp) { statlist }	
break	Sai do ciclo corrente e continua
continue	Passa à iteração seguinte do ciclo corrente
next	Não aplica os padrões seguintes à linha corrente
exit	Esquece o resto do input e passa para o END se este existir
print [exp_list] [> filename]	
Print f format [exp_list] [> filename]	Como o printf em C

Variáveis

- O nome pode ser formado com letras, digitos e o carácter '_';
- Podem ser do tipo string ou numérico;
- Não é necessário declará-las nem inicializá-las;
- O tipo é inferido na primeira atribuição;
- Os nomes dos campos ($1, $2, …, $n) podem ser usados como variáveis.

Arrays

- Arrays de uma dimensão: array_name[index];
- O índice pode ser numérico ou string;
- O índice começa em 1 se for numérico;
- Não é necessário qualquer declaração, basta começar a atribuir valores;
- Não têm limite. Estão limitados à memória disponível;
- Exemplos:
  - meses[“Janeiro”], grelha[1], cursos[“LEI”], …

Arrays multi-dimensionais

- Em awk, os arrays têm apenas uma dimensão;
- Array_nome[1,2] – não é suportado;
- Pode-se concatenar os índices numa string e usá-la como índice:
  - Array_nome[1 ”,” 2] – o espaço é o operador de concatenação, neste caso, o índice é “1,2”;
- Pode-se usar a variável SUBSEP para eliminar a necessidade das aspas à volta da vírgula.

Funções

Função	Descrição
cos(awk_exp)	Cosseno
exp(awk_exp)	e^{awk_exp}
index(str1, str2)	Posição de str2 em str1
length(str)	Comprimento de str
log(awk_exp)	$\log_e awk\_exp$
sin(awk_exp)	Seno
sprintf(frmt, awk_exp)	Valor de awk_exp formatado de acordo com frmt
sqrt(awk_exp)	Raíz quadrada
split(str, array, [field_sep])	Parte um string e armazena-a no array
substr(str, start, length)	Substring
toupper( ), tolower()	Case

Funções

Função	Descrição
sub(pat1, “pat2”, [string])	Substitui a primeira ocorrência de pat1 por pat2 na string; por omissão string é a linha inteira
gsub(pat1, “pat2”, [string])	Igual à anterior só que substitui as ocorrências todas
match(string, pat1)	Procura a expreg pat1 e inicializa as variáveis RSTART e RLENGTH que indicam onde a expressão regular começa e termina
systime( )	Retorna o número de segundos desde 1 de Janeiro de 1970

Match “with case insensitive”

- awk ‘BEGIN { ignorecase=1 } /PAT1/’
- awk ‘tolower($0) ~ /pat1/ …’

Gawk: funções def. pelo utilizador

```
#!/usr/bin/gawk -f
{
  if(NF != 4)
    { erro("Eram esperados 4 campos"); }
  else
    { print; }
}

function erro (mensagem)
{
  if(FILENAME != "-")
    { printf("%s: ", FILENAME); }
  printf("linha n%d, %s, linha: %s\n", NR, mensagem, $0);
}
```

Exemplos de 1 linha

Exemplo	Descrição
awk ‘{print $NF}’ infile	?
awk ‘{print $(NF-1)}’ infile	E se os () forem retirados? E se a linha tiver apenas 1 campo
awk ‘NF’ infile	
awk ‘{print length, $0}’ infile	
awk ‘BEGIN {while(++x<11) print x}’	
awk ‘BEGIN {for(i=10;i<=50;i+=4) print i}’	
awk ‘{print; print “”}’ infile	
awk ‘{print; if(NF>0) print “”}’ infile	
awk ‘NF!=0 {++conta} END {print conta}’ infile	

Exemplos de 1 linha

Exemplo	Descrição
ls –l | awk 'NR>1 {s+=$5} END {print "TM: s/NR-1"}	?
awk '/pat1/?/pat2/:/pat3/' infile	
awk 'NF<10?/pat2/:/pat3/' infile	
awk 'ORS=NR%3?" ":"\n"' infile	
awk 'ORS=NR%3?"\t":"\n" {print $1}' infile	
awk 'FNR<11' f1, f2, f3	
awk 'length < 81'	
awk '/pat1/, 0'	A condição 0 é falsa
awk 'NR==10, 0'	
awk '{sub(/[\t]+$/, ""); print }'	
awk '{gsub(/^[\t]+|[\t]+$/, ""); print }'	

awk '{sub(/[\t]+$/, ""); print }'

Exemplos de 1 linha

Exemplo	Descrição
awk ‘/pat1/ {gsub(/pat2/, str); print }’	
awk ‘{$NF = “”; print }’	
awk '{ print } { print "" }'	
awk 'BEGIN { ORS="\n\n" }; 1'	

Exercícios

1. Colocar uma linha de espaço entre as linhas da entrada (não duplicar linhas em branco);
2. Numera as linhas de vários ficheiros independentemente;
3. Numera as linhas de vários ficheiros consecutivamente;
4. Conta as linhas dum ficheiro;
5. Imprime a soma de todos os campos de todas as linhas;
6. Quantos “Alfredos” existem referenciados no ficheiro?
7. Qual o maior ficheiro na diretoria corrente?

Mais exercícios

8. Elimina o espaço extra entre campos;
9. Adiciona 5 espaços no início de cada linha;
10. Mudar os nomes “Zeferino Pereira” para “Zeferino Martins”;
11. Inverte a ordem das linhas da entrada;
12. Produz uma lista ordenada alfabeticamente dos nomes dos inquiridos em processos.txt;
13. Imprime os campos 2 e 3 em ordem inversa;
14. Remove linhas duplicadas consecutivas;
15. Remove linhas duplicadas não consecutivas;

Mais exercícios(2)

16. Concatena grupos de 5 linhas e separa-as por vírgula;
17. Imprime a primeira linha da entrada (head -1);
18. Imprime a linha anterior à que faz match com /regex/;
19. Imprime a linha seguinte à que faz match com /regex/;
20. Apaga as linhas brancas da entrada;
21. No documento dos processos, quero saber quantas vezes se repete cada nome ($5);