O comando AWK com if, else e outras opções

Link: https://blog.ironlinux.com.br/o-comando-awk/

Assim como o SED , o AWK é uma ferramenta para manipulação de texto. No entanto, o AWK também é considerado uma linguagem de programação. Com ele é possível pesquisar palavras num arquivo, identificar padrões, realizar substituições e muito mais! Além disso, o AWK suporta expressões regulares, o que permite realizar matches de padrões complexos.

Output de exemplo

Antes de tudo, para realizarmos as operações/exemplos com o AWK, vamos utilizar a saída do comando **ps u**:

```
ps u
```

Output do comando ps

1 | Utilizações básicas

1.1 | Printar a primeira coluna

Para apresentar apena a primeira coluna é possível utilizar o comando abaixo. A primeira coluna é representada por **$1**:

```
ps u | awk '{print $1}'
```

Primeira coluna com AWK

1.2 | Printar múltiplas colunas

É possível trazer múltiplas colunas utilizando o comando abaixo. OBS: A vírgula neste exemplo representará um espaço comum na saída final:

```
ps u | awk '{print $1,$2,$3}'
```

Múltiplas colunas com AWK

1.3 | Printar múltiplas colunas separadas por Tab

Utilizando **"\t"** é possível separar as colunas com Tab:

```
ps u | awk '{print $1 "\t" $2 "\t" $3}'
```

Múltiplas colunas separadas por tab com AWK

1.4 | Printar o último elemento

Utilizando **$NF** é possível trazer o último elemento (neste caso é a coluna COMMAND):

```
ps u | awk '{print $NF}'
```

Último elemento/coluna com AWK

1.5 | Ignorar a primeira linha

É comum precisar remover a primeira linha de um arquivo para depois trabalhar com os dados. Para fazer isso, basta utilizar o comando abaixo:

```
ps u | awk 'NR!=1'
```

Ignorar primeira linha com AWK

1.6 | Substituir texto

Para substituir um texto, podemos utilizar a função **sub()**, conforme o exemplo abaixo, que substitui a string "**kali**" por "**outro-usuario**":

```
ps u | awk -e 'sub(/kali/, "outro-usuario")'
```

Substituir texto com AWK

OBS: A função **sub()** substitui apenas a primeira ocorrência, uma vez por linha. Caso queira substituir mais de uma ocorrência, utilize a função **gsub()**.

2 | Utilizando um outro delimitador

Por padrão, o delimitador do AWK é o espaço (ou tab). No entanto, em alguns casos, você precisará indicar um outro delimitador (como por exemplo **vírgula** ou **ponto e vírgula**). Desta forma, no exemplo abaixo, estamos utilizando **;** como delimitador e printando o segundo elemento:

```
echo 'oi;tudo;certo' | awk -F ';' '{print $2}'
```

Alterar delimitador AWK

3 | Condicionais

3.1 | AWK com if

Para exemplificar o uso de condicionais (if) vamos utilizar o arquivo **notas.txt** que possui o seguinte conteúdo:

IF no AWK

Por exemplo, para printar a linha inteira se a primeira coluna for a string **Iron**:

```
awk '{ if ($1 == "Iron") print $0 }' notas.txt
```

Primeira coluna com if no AWK

Por exemplo, para printar a nota do aluno **Iron** em uma frase:

```
awk '{ if ($1 == "Iron") print "A nota do Aluno", $1, "foi", $2}' notas.txt
```

Manipulando output com if no AWK

3.2 | AWK com if/else

No exemplo abaixo estamos utilizando if/else para determinar quais alunos reprovaram ou passaram (com nota maior que 5). Também estamos utilizando **NR!=1** para ignorar a primeira linha:

```
awk 'NR!=1 {if ($2 >=5 ) print $0,"=>","Passou!"; else print $0,"=>","Reprovou!"}' notas.txt
```

AWK if e else

3.3 | Cheatsheet de condicionais

Condicionais	Descrição
if ($5 >= 10)	Se a quinta coluna for maior ou igual a 10
if ($3 == 10)	Se a terceira coluna for igual a 10
if ($1 == “Linux”)	Se a primeira coluna for igual a string **Linux**
if ($1 == “Linux”	
if ($1 ==“Linux” && $2 > 5)	Se a primeira coluna for igual a string **Linux e** a segunda coluna for **maior** que **5**

4 | Utilizando REGEX

4.1 | Exemplos com REGEX

Na regex abaixo, estamos printando a linha inteira caso a segunda coluna se inicie com o número 1:

```
ps u | awk -e '$2 ~ /^1/ {print $0}'
```

Regex com AWK - exemplo 1

Na regex abaixo estamos printando todas as linhas cuja coluna 2 **não** comecem com o número 1:

```
ps u | awk -e '$2 !~ /^1/ {print $0}'
```

Regex com AWK - exemplo 2

4.2 | Cheatsheet de REGEX

Regex	Descrição
[mr]	Letras** m** ou **r**
[a-z]	Qualquer letra de **a** à **z**
[a-zA-Z]	Qualquer letra de **A** à **Z** (maiúsculo ou minúsculo)
[A-Z0-9]{5}	5 caracteres, podendo ser qualquer letra de A à Z ou números de 0 a 9

5 | Alguns outros usos interessantes

5.1 | Pegar linhas entre dois padrões

Vamos utilizar o arquivo **padrao.txt** abaixo para realizar as operações:

Coletar linhas entre padrões com AWK
Caso você queira printar, todas as linhas entre “**padrao1**” e “**padrao2**”:

```
awk '/padrao1/{flag=1;next}/padrao2/{flag=0}flag' padrao.txt
```

Linhas entre padrões com AWK
Caso queira que “**padrao1**” e “**padrao2**” também seja printado:

```
awk '/padrao1/{a=1}/padrao2/{print;a=0}a' padrao.txt
```

Conteúdo entre padrões com AWK

5.2 | Adicionar um prefixo nas linhas

Para adicionar um prefixo nas linhas pode-se utilizar a função **gensub()**, veja o exemplo abaixo, onde adicionamos a palavra “Prefixos” em todas as linhas que comecem com caracteres alfanuméricos:

```
awk -e ' { print gensub(/^[a-zA-Z0-9]*/, "Prefixos &",1) }' notas.txt
```

gensub no AWK

Por fim, agradecemos a leitura e esperamos que este post tenha te ajudado de alguma maneira!

Caso tenha alguma dúvida, entre em contato conosco pelo Telegram , Facebook ou Instagram !

Veja mais posts no IronLinux !

Tags:

- Awk
- Comandos linux
- Linux
- Manipulação de texto
- Sed

Posts relacionados

Redirecionar a saída padrão e de erros

Redirecionar a saída padrão e de erros

- Vinicius Souza
- Linux

Quando é executado um comando ou algum script no Linux é possível redirecionar a saída padrão e de erros para não ser printado em tela ou que seja direcionado à algum lugar especifico.

Ler post completo

O comando SED no Linux

O comando SED no Linux

- Vinicius Souza
- Linux

O comando SED é uma ótima ferramenta de edição de arquivos ou de formatação de resultados de comandos, com ele você pode pesquisar, localizar e substituir, inserir ou excluir palavras, números e etc.

Ler post completo
Estressando MEM, DISCO e CPU com Stress-ng [Debian9]

Estressando MEM, DISCO e CPU com Stress-ng [Debian9]

- Vinicius Souza
- Linux

O STRESS-NG Com a ferramenta Stress-ng podemos literalmente realizar o Stress de alguns recursos do seu servidor, sendo eles: Cpu, memória e disco.

Ler post completo

Revision #1
Created 2024-07-28 12:56:39 UTC by Administrador
Updated 2025-05-16 11:49:54 UTC by Administrador