Aula 7

7.1 – Manipulando dados com AWK

O AWK é definida como uma linguagem para criação de rotinas cujo foco principal constituí-se na formatação dos dados. Como toda linguagem interpretada, ela é simples e prática, sendo muito bem aproveitada para situações em ambientes de programação em *shell*.

Basicamente, o que o AWK faz, é reconhecer um padrão, e através desse reconhecimento, aplicar um conjunto de regras ou ações. Analisando de maneira simplista, sua lógica pode ser definida como:

padrão { ação }
padrão { ação }
…

Como é de praxe com qualquer outro aplicativo; para o *awk* também é necessário passar um conjunto de dados de entrada. Por sua vez, o *awk* irá recorrer ou aplicar às instruções de comandos de formatação sobre esses dados de entrada. Ao fim, o awk envia os dados devidamente alterados para a saída padrão.

O uso das instruções *awk* é feita com dados de entrada como no exemplo:

#> awk 'programa'
#> awk 'programa' arquivo1 arquivo2 …

Dessa forma, podemos ter diversas opções para que um programa *awk* trabalhe com seus dados de entrada. As mais simples delas, podem ser vistas a seguir:

```
awk  "BEGIN {print \" Testando \" }"
awk '  { print  " testando"  }'
```

Imprime a palavra “testando” na saída padrão

```
awk ' BEGIN { print "Procurando um  \"padrao\"  "}
>      /padrao/ { ++x }
>    END { print "padrao apareceu", x, "vezes." }' arquivo_entrada
```

Imprime quantas vezes (em linhas) a palavra “padrao” aparece no arquivo de entrada

```
awk 'BEGIN { RS=":" }
 > { print $0 }' /etc/passwd
```

Imprime somente o primeiro campo do arquivo /etc/passwd, cujo o delimitador de campos é ":"

```
cat /etc/passwd | awk -F":" '{print  $1}'
```

Imprime somente o primeiro campo de cada linha do arquivo /etc/passwd. Sendo que neste caso o delimitador de campos é o símbolo ":"

7.2 – Formatando com comandos auxiliares

Existem muitos outros comandos e aplicativos também bastante usados para a formatação e filtragem de dados de entrada em Linux.

A seguir, mostraremos alguns dos mais importantes desses comandos através de exemplos bastante sucintos.

- tr: O comando *translate* serve justamente para traduzir um padrão de um ou mais caracteres em um novo padrão. O exemplo de comandos *tr* pode ilustrar bem finalidades de uso:

```
cat /var/log/messages | tr "br" "sp"
# Imprime o arquivo messages na saída padrão e troca os caracteres 'b' por 's' e 'r' por 'p'

echo "abcdefg" | tr  '[:lower:]' '[:upper:]'
# Imprime a cadeia de caracteres de entrada alterando-a para caracteres em caixa alta

cat passwd | tr --delete '=;:`"<>,./?!@#$%^&(){}[]'
# Imprime o arquivo passwd na saída retirando quaisquer uns dos símbolos listados entre aspas

free | tr -s ' '
# Imprime o resultado comando "free" eliminado todos os espaços existentes.
```

- cut: O comando cut utiliza delimitadores para executar o que seu próprio nome indica. Este comando vai cortando e reproduzindo na saída o resultado desejado a partir de um conjunto de entrada. Exemplos bem simples são vistos a seguir:

```
 cat /etc/passwd | cut -d: -f1,7
# Imprime as colunas 1 e 7 do arquivo passwd, sujo delimitador é ":"
```

w | cut -c1-8
Imprime os caracteres de 1 até 8 da saída do comando “w”

- sed: O comando *sed* pode ser utilizado quando se deseja realizar a alteração ou substituição de um texto de entrada. Essa aplicação é muito utilizada porque pode receber expressões regulares como forma de padrões a serem reconhecidos. Por conta disso, esse comando é muito poderoso e é bastante utilizado por editores de texto e *scripts* que fazem formatação ou substituição de conjunto de textos. Alguns exemplos bastante simples podem ser vistos:

sed 's/palavra_velha/palavra_nova/g' arquivo
Substitui todas ocorrências de “palavra_velha” por “palavra_nova” em arquivo

sed 'linha_inicial,linha_finald' arquivo
Remove desde a “linha_inicial” até a linha “linha_final do arquivo e imprime na saída padrão

sed 'linha_inicial,/palavra_final/d' arquivo
Remove desde a “linha_inicial” até a primeira ocorrência de “palavra_final” no arquivo e imprime na saída padrão.

7.3 Exercícios

1 - Obtenha as suas partições (df -h) e escreva somente os dispositivos, porcentagens de espaço livre e ponto de montagem conforme a saída abaixo:

/dev/hdb2 80% /home

2 – Reescreva o comando anterior, inserindo dois pontos como separador de colunas

3 – Imprima novamente, removendo a linha que contém /dev/shm line e salvando em um arquivo texto em /tmp/

4 – Obtenha a saída do comando *lsusb* e separe por vírgulas os campos existentes, salvando em um arquivo de saída

5 – Com a saída resultante do comando anterior substitua todas as palavras “Device” por dispositivo e salve em um novo arquivo.

6 – Com este último arquivo resultante remova as três primeiras linhas do mesmo e salve em um novo arquivo.

7.4 - Bibliografia

Diversos, The GNU Awk User's Guide. Acessado em 22 de setembro de 2009, http://www.gnu.org/software/gawk/manual/gawk.html

Hamish Whittal. Shell Script (2005). Acessado em 22 de setembro de 2009, http://learnlinux.tsf.org.za/courses/build/shell-scripting/index.html