***Apostila de AWK***

**Introdução ao AWK:**

**AWK** é uma Linguagem de Progamação criada para fazer o processamento de textos e análise de dados, utilizada em sistemas Unix/Linux, para manipular arquivos de textos, filtrar dados, modificar saídas e até mesmo gerar relatórios, sendo uma linguagem extramemente poderosa.

A linguagem recebeu esse nome por causa dos seus criadadores, **Alfred V.aho**, **Peter J.Weingerger** e **Brian W.Kernighan**, onde foi usado as iniciais do nome de cada um deles.

Esses três cientistas da computação trabalham na **Bell Labs** mesmo laboratório onde surgiu o unix, dando origem e criação da linguagem em 1977, sendo uma das primeiras linguagens criada para manipulação de texto.

Na época o Unix estava se tornando um sistema popular entre os progamadores e cientistas, e com isso várias ferramentas estavam sendo criadas, com intuito de facilitar a manipulação dos arquivos e textos. Tais ferramentas que podiam ser combinadas e formavam a base do que o Unix podia oferecer.

Antes da AWK ser formada, já existiam outros comandos como:

**ed:**

O primeiro editor de texto do Unix. Funcionava no terminal sem nenhum tipo de Interface Gráfica. Para usá-lo é bem simples…

1. Usamos o comando **“ed”** para iniciar o editor.
2. Digitamos a letra **“a”** na primeira linha, informando que escreveremos um texto.
3. Digitamos o texto nas linhas abaixo.
4. Na penúltima linha digitamos um “.”, informando que acabamos de digitar o texto.
5. Na última linha escrevemos **“w”**, que é usado para salvar o contéudo no arquivo, depois um espaço, e o nome do arquivo que será salvo.
6. Para sair digitamos **“q”**.

Se quiser continuar editando o arquivo, basta…

1. Usarmos o comando **“ed** nome\_do\_arquivo**”**.
2. Na primeira linha será exibído o número de bytes atual do arquivo,
3. Agora basta continuarmos o processo apresentado acima.

**cut:**

Esse comando era usado para recortar colunas ou campos das linhas de um arquivo, muito utilizado em arquivos CSV. Neste comando temos os seguintes parâmetros de passagem.

**-d →** Utilizado para inserir um delimitador.

**-c →** Para fazer a busca de uma posição específica de caracter.

**-b →** Para fazer a busca de uma posição específica de byte.

**-f →** Para buscar uma posição específica da string ao utilizar um delimitador.

**-s →** Usado para não emitir comandos que não contenham um delimitador.

**-z →** Trata a separação de delimitadores como caracteres nulos e não como espaços que é o padrão. Ele usa **“\0”**  ao invés da quebra de linha **“\n”**.

Sendo assim podemos por exemplo usar o seguinte comando em um arquivo de lista:

**cut -d “ “ -f 1**

Com Esse comando estaremos obtendo apenas a primeira coluna de cada linha. As colunas são separadas apartir do delimitador **espaço**.

**sed:**

O comando **sed** é um comando comumente utilizado para manipular arquivos de textos e strings. E muito utilizado principalmente com uso de Expressões Regulares.

Podemos substituir o conteúdo no arquivo:

**sed “s**/texto\_atual**/**novo\_texto**/”** nome\_do\_arquivo

Alterar uma linha inteira:

**sed “3c/**Estamos alterando o texto da linha 3.**”** nome\_do\_arquivo

Inserir uma linha antes de alguma linha existente:

**sed “2i/**Estamos inserindo o texto antes da linha 2.**”** nome\_do\_arquivo

Inserir uma linha depois de alguma linha existente:

**sed “4a/**Estamos inserindo o texto depois da linha 4.**”** nome\_do\_arquivo

Usando o parâmetro **-i** no comando para criar um arquivo de backup antes do arquivo de texto ser salvo. Esse arquivo será salvo com a extensão **“.bak”**, o diferenciando do arquivo original.

**sed -i “s**/texto\_atual**/**novo\_texto**/”** nome\_do\_arquivo

Podemos também utilizar Expressões Regulares de maneira completa:

**sed -E “s/**expressão\_regular**/”** nome\_do\_arquivo

**tr:**

O comando **tr** é utilizado para substituir valores de strings. Assim podemos por exemplo fazer o teste desse comando usando o **echo**, que imprimi um texto na tela, da seguinte forma:

**echo “Banana,Maça,Laranja” | tr “,” “-“**

O resultado será:

**Banana-Maça-Laranja**

**grep:**

O comando **grep** faz a filtragem de resultado de pesquisas buscando aquilo que foi determinado, sendo como uma barra de pesquisa, apartir de um comando ou arquivo aprensentado por exemplo.

Esse comando também contém alguns parâmetros interressantes como...

Filtrar dados específicos em um arquivo de texto:

**grep “**texto**”** nome\_do\_arquivo

Podemos usar o parâmetro **“-i”** para diferenciar minúsculas de maiúsculas:

**grep -i “**texto**”** nome\_do\_arquivo

Podemos usar o parâmetro **“-n”** para fazer a filtragem com número de linhas:

**grep -n “**texto**”** nome\_do\_arquivo

Podemos usar o parâmetro **“-w”** para uma busca precisa:

**grep -w “**texto**”** nome\_do\_arquivo

Podemos usar o parâmetro **“-E”** para uso de Expressões Regulares:

**grep -E “**expressão\_regular**”** nome\_do\_arquivo

Assim todos esses comandos apresentados se mostram muito utéis para manipulação de texto, porém é necessário mesclar o uso deles e de muitos outros. Então por conta disso criaram a Linguagem de Progamação AWK, que facilita o uso e manipulação de texto usando apenas uma biblioteca com apenas 1 comando.