Shell Script do zero

Participe do processo de seleção para passarmos as aulas para vídeo, será remunerado.

Seja você iniciante ou não, o importante é saber explicar e seguir o conteúdo do material. para quem está fazendo o curso ainda dá tempo de concorrer.

Caso esteja interessado grave uma aula teste explicando os operadores de comparação com conectores usando o comando if.

Serão avaliados:

Se você explica bem e os recursos usados (quadro, imagens, exemplos etc.)

Upa o teste de preferência no media fire e manda o link no meu e-mail luizsouza99@gmail.com

Aula 12 – Comandos sed, cut, pipeline, tr e grep

Os comandos apresentados aqui deixam o script mais "inteligente" ajudando-o a "ler", escrever, filtrar, colocar mais de um comando na mesma linha etc.

O texto abaixo será usado de exemplo nas próximas ferramentas mostradas aqui

L1 - Contribuir com o software livre é contribuir consigo mesmo,
L2 - porque hoje você usa a tecnologia criada por alguém no passado
L3 - e amanhã outros poderão desfrutar da sua contribuição iniciada hoje.

Sed

O comando sed tem várias funcionalidades relacionadas a filtragem, como exibir linhas com determinada palavra, extrair um trecho do texto, trocar linhas de lugar etc. Como o material do mesmo é vasto vamos aprender apenas a filtrar textos especificando por linha, o que já nos ajudará nas filtragens.

Para mostrarmos somente determinada linha damos o comando:

```
sed -n '2p'
```
Linha 2

Mas isto não é o suficiente, devemos executar algum comando para que ele filtre sua saída, seja o ls, cat, ps etc. Abaixo eu usei o comando cat (exibe o conteúdo de arquivos-texto) no nosso texto exemplo e filtrei com o sed.

```
cat /home/luiz/texto | sed -n '2p'
```

Agora sim, a saída do comando cat são 3 linhas, o sed pega este resultado faz a filtragem e mostra apenas a linha 2. Veremos a barra antes do sed nesta aula, por enquanto basta saber que faz o sed processar o resultado do primeiro comando.

Com o comando acima a saída no texto de exemplo seria esta:

L2 - porque hoje você usa a tecnologia criada por alguém no passado

Para “eliminarmos” determinada linha:

Usamos a sintaxe abaixo indicando a linha a ser ocultada

```
cat /home/luiz/texto | sed 1d
```

Saindo assim o nosso texto exemplo:

L2 - porque hoje você usa a tecnologia criada por alguém no passado
L3 - e amanhã outros poderão desfrutar da sua contribuição iniciada hoje.

Ou então posso eliminar mais de uma linha:

```
cat /home/luiz/texto | sed -e 1d -e 3d
```

Como são 3 linhas poderíamos usar o comando de incluir e chegaríamos no mesmo resultado, mas é só exemplo.

Saindo assim:

L2 - porque hoje você usa a tecnologia criada por alguém no passado

Como somos usuários iniciantes no shell apresentei apenas o básico e na sua caminhada você poderá conhecer os vários recursos que este comando tem.

Caso queira se aprofundar no sed acesse o link:
http://thobias.org/doc/sosed.html

Cut

Com o cut limitamos a saída dos comandos em campos, este texto que escrevo podemos considerar que seus campos são determinados pelo carácter “espaço”, então se eu pedir ao cut que me mostre o conteúdo do campo 3 ele me retornará a palavra “cut”, porque as palavras são os campos e os espaços mostram ao shell onde é a delimitação dos mesmos, nada te impede de usar a letra “a” por exemplo como delimitador, sairá um resultado muito louco com isto quero dizer que é você quem determinar o que será o delimitador e consequentemente os campos.

No exemplo anterior considerei o texto não tendo nenhum enter, ou seja se temos um arquivo com várias linhas que são exibidas separadamente, o cut mostrará o campo X de cada linha.

O texto abaixo por exemplo, podemos usar o delimitador “dois pontos e três linhas”

```
mouse:azul:novo
carro:branco:usado
radio:preto:novo
```

Então usando o comando cut para mostrar o campo 2, seria exibido isto:

```
azul
branco
preto
```

Se quisermos mostrar somente a palavra “preto” é só usar o cut e sed		
cat texto |	“cut” pedindo o 2º campo	| sed -n '3p' mostrou a linha 3
mouse:azul:novo carro:branco:usado radio:preto:novo	azul branco preto	preto

O importante é sempre procurar padrões que lhes sirvam de base para chegar no conteúdo que alimente corretamente o script. No comando ps podemos filtrá-lo de uma forma que cheguemos somente no pid, o problema é que o pid passando de certa numeração ganha mais um carácter numérico diminuindo um carácter espaço, ou seja o seu comando que funcionava perfeitamente, simplesmente não vai servir mais (tem maneiras de contornar isto, mas não abordarei aqui).

Então preste atenção em todas as variações possíveis para evitar erros no script que nem sempre são fáceis de enxergar.

Sintaxe

Comando	-f mais número do campo a ser mostrado	Aqui mostro o carácter que determina o campo “espaço”.
cut	-f1	-d' '

Da mesma forma que o sed, o cut também precisa de usar o resultado de outro comando:

```
cat /home/luiz/texto | cut -f5 -d' '
```

Resultado do comando acima dado no texto exemplo (somente na primeira linha e sem o L1):

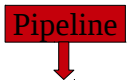
```
luiz@linuxmint ~ $ cat /home/luiz/texto | cut -f5 -d' '
livre
```

É isto que tínhamos sobre o cut e nunca esqueça que um dia você precisará de uma informação tão especifica para o seu script e o cut + sed serão a solução.

Pipeline

Muitas vezes precisamos pegar a saída do comando A para usarmos no comando B e chegar no resultado final com o comando C, D etc. Além disto podemos diminuir o tamanho do script juntando vários comandos na mesma linha, é exatamente isto que o pipeline faz, ele é representado pelo símbolo “|” que está ao lado da letra “z” bastando digitá-lo com o shift.

Exemplo simples:




```
echo 1 > /prog/.LOOP1.txt | echo 1 > /prog/.LOOP2.txt | echo 1 > /prog/.LOOP3.txt | echo 1 > /prog/.LOOP4.txt
```

No exemplo acima eu precisava zerar os arquivos-texto responsáveis pelo loop e não fazia muito sentido usar várias linhas para isto, então coloquei todos concatenados na mesma linha usando o pipeline.

Exemplo mais avançado:

```
ting=$(ping -w 2 192.168.0.160 | cut -f1 -d' ' | sed -n '4p' )
```

Acima dou 3 pings (“-w 2” sai 3 mesmo), limito a saída para o campo 1 (determinado por espaço) e o resultado disto eu ainda pego e mostro apenas a linha 4 usando o sed.

Para usar o pipeline basta usar a criatividade, seja para chegar num resultado que seria mais difícil encontrar sem o pipe ou apenas para compactar o script, o importante é sempre testar, porque nem tudo que parece funcionar vai funcionar.

Tr

Com este comando é possível substituir caracteres de um texto por outros, é claro que podemos incluir como texto um script ou um arquivo de configuração por exemplo.

Sintaxe

Comando	Palavra ou carácter a ser substituído	O que vai entrar no lugar
tr	['a,e']	['4,3']

É importante não deixar espaços dentro dos colchetes para funcionar

Executando o comando abaixo em nosso texto exemplo, ele substituirá a letra “a” pelo número “4” e a letra “e” pelo número 3.

```
cat /home/luiz/texto | tr ['a,e'] ['4,3']
```

Ficando assim:

Contribuir com o softw4r3 livr3 é contribuir consigo m3smo,
porqu3 hoj3 você us4 4 t3cnologi4 cri4d4 por 4lguém no p4ss4do
3 4m4nhã outros pod3rão d3sfrut4r d4 su4 contribuição inici4d4 hoj3.

Lembrando que este foi um resultado mostrado na tela e se precisarmos de uma ação definitiva necessitaremos direcionar a saída para um arquivo texto (pode ser o mesmo)

Uma função muito útil no comando tr é a de transformar caracteres minúsculos em maiúsculos e vice-versa, assim se pedirmos ao usuário que digite algo e que este valor seja obrigatório em maiúsculo ou minúsculo podemos transformar este valor com o tr garantindo assim o funcionamento do script.

Transforma tudo em maiúscula	Tudo minúscula
cat texto | tr a-z A-Z	cat texto | tr A-Z a-z
Elimina o que tem entre aspas do texto (aqui elimina espaços)	
cat texto | tr -d " "	

O código abaixo é um exemplo de como podemos pedir um valor ao usuário e transformá-lo do jeito que quisermos, aqui passo tudo que esta maiúsculo para minusculo. Se o usuário digitar “CASA” então o valor da senha passará a ser “casa”, se ele digitar “CAsa” então teremos “casa” e assim por diante.

```
read senha

senha=$(echo $senha | tr A-Z a-z)
```

Grep

Com o comando grep também filtramos saídas, mas aqui é por palavra e não por número de linha, este comando mostra todas as linhas onde haja a incidência da palavra especificada, ele é muito usado para filtrar saídas extensas no terminal.

Sintaxe

Comando a ser filtrado e pipeline para concatenar	Comando grep	Palavra a ser filtrada
`ls |`	`grep`	`vlc`

No comando acima eu listo os diretórios e peço o grep para mostrar todos que tenham a palavra vlc.

Vamos supor que vou dar o comando "ls -la" no "/home" o resultado será bem grande, mas eu desejo localizar os diretórios contendo o nome "gnome", então …

Sem grep

```
luiz@linuxmint ~ $ ls -la
total 7720
drwxr-xr-x 48 luiz luiz    4096 Nov 23 21:45 .
drwxr-xr-x  4 root root    4096 Nov 15 12:34 ..
-rw-r--r--  1 luiz luiz 1084440 Set 25 12:04 .0gbas.so_u
-rw-r--r--  1 luiz luiz 1084440 Set 25 12:06 .1gbas.so_u
drwx------  3 luiz luiz    4096 Set 19 01:07 .adobe
drwxr-xr-x  2 luiz luiz    4096 Nov 22 13:54 Área de Trabalho
drwxr-xr-x  4 luiz luiz    4096 Nov 11 12:04 .audacity-data
-rw-r--r--  1 luiz luiz     101 Out 19 12:37 aviso
drwxr-xr-x 13 luiz luiz    4096 Nov 23 21:50 .azureus
```

E muito mais resultado para baixo

Com grep

```
luiz@linuxmint ~ $ ls -la | grep gnome
drwx------  3 luiz luiz    4096 Out 19 19:37 .gnome2
drwx------  2 luiz luiz    4096 Set 18 13:48 .gnome2_private
```

Mostrando apenas as linhas onde constam a palavra gnome

Se houver qualquer dificuldade com este comando basta treiná-lo no terminal filtrando as saídas dos comandos que você está acostumado usar.

Exercício 8

Usando a saída do comando "ls -la" no /home

Faça um comando que exiba:

Somente o mês de novembro, apenas uma vez e mais nada.

A resposta será somente → Nov

Pode ser o mês que desejar, este não vai ter resposta porque de um jeito ou de outro você conseguirá chegar no resultado esperado.