Regex é uma expressão regular que define um método de pesquisa através de uma sequência de caracteres que pode representar localizar ou localizar e substituir.

ER=Expressão regular

String é uma sequência de caractere que pode formar por exemplo uma palavra ou uma frase.

Metacaracteres
[ ] incluem um conjunto de caracteres

[0-9] (corresponderá a uma série de caracteres) exemplo:

Dentro do diretório tem arquivos numerados por números agora vamos dar o comando: ls | grep [0-9]
colega1.mnb
colega.3.mnb
colega4.mnb #O grep verifica de 0 a 9 os nomes dos arquivos que contem número . Pode se usar também [a-e] para verificar palavras que tem de A até E =ABCDE

grep "N[oen]n"

Aqui procuramos uma palavra que comece com 'N', termine com 'n' e só possa ter 'o' ou 'e' ou 'n' no meio.

echo x y z FG | grep -i [xyz] Resultado: **x y z** FG . O grep só detectou **xyz**

echo yes | grep "[Yy][Ee][Ss]" Corresponde a *yes* , *Yes* , *YES* , *yEs* e assim por diante.

echo não | grep [n][ã][o] Neste exemplo o grep só detectará a palavra "**não**"

Localizar os dígitos de 0-9 e o alfabeto maiúsculo de A-F , e minusculo a-f

echo L 0 1 | grep -E "[0-9A-Fa-f]" Resultado: L **0 1** Não foi detectado o L já que no alfabeto é bem depois de F.

[^] corresponderá a todos os caracteres, exceto o mencionado entre colchetes exp: [^ a] não localiza a letra a . Enquanto só o ^ corresponde ao início de uma linha**:**

echo axa | grep ^a **Resultado:** **a**xa Localiza a letra inicial na linha que é a

echo axa | grep [^a] **Resultado:** a**x**a Não localiza a letra inicial na linha que é a

Como tirar um espaço vazio no inicio de uma linha usando **sed** e **[^]**

Tem um documento chamado branc e nele existe um espaço vazio de um caractere ' toca'

sed 's/[^H]toca/bolo/' branc Resultado: 'bolo'

Explicação: o [^H] Define que o vazio no inicio da linha não existe portanto quando o sed faz a troca da palavra toca por bolo não aparece o vazio. A letra H está fazendo o papel de dizer que existe um caractere qualquer no inicio da linha portando pode ser usado qualquer letra.

--Quando você não quer localizar algo mesmo sabendo que existe você exclui

***** = Verifica o nome anterior e posterior como por exemplo a extensão de arquivos

Diferença de ***** e **+**
Este comando logo abaixo deve ser executado em um diretório e se for executado no /home/$USER tem que usar aspas.

echo abbbe | grep ab* Detecta tudo depois da repetição da ultima letra Resultado: abbb (Atenção o grep não conseguiu identificar a ultima letra e)
--grep a(b)* repetição da ultima letra = b

echo abbbe | grep 'ab\+e' Detecta tudo depois de ab, mas se tiver qualquer letra diferente de b na frente da primeira ocorrência do mesmo já é considerado outra palavra. Como foi programado dizendo que a ultima letra é (e) o grep já localiza a ultima letra. Resultado: abbbe

Resumo:
***** Tem a vantagem de localizar vários resultados sem depender da letra posterior se for diferente.
+ é mais concentrado na palavra e se tiver qualquer letra diferente da especificada já não é considerado.

$ Cifrão no final de uma ER corresponde ao final de uma linha.

echo sdXXX XXX | sed 's/XXX//' Resulado: sd XXX (ficou um espaço entre sd e XXX), mas para tirar o espaço podemos usar **echo sdXXX XXX | sed 's/XXX$//' Resultado:** sdXXX .Portando $ pode representar o vazio.

Deletar espaços vazios de linhas de um arquivo:
Conteúdo do arquivo branc:
aaa
aaa
‘
‘ Linhas vazias

sed '/^$/d' branc Resultado: ‘aaa Com o comando grep pode ser: cat branc | grep [^$]
aaa’

O operador d do sed serve para: deletar
"^$" corresponde a linhas em branco.

\<….\>

echo the there other | grep the Mostrá todas palavras que estiverem includas com **the**
Resultado: **the the**re o**the**r

echo the there other | grep "\<the\>" Mostrá somente **the**
the there other

\ **(**escape**)** Quando as expressões regulares veem uma barra invertida, elas sabem que devem interpretar o próximo caractere. Ativa ou desativa o metacaractere versus o status literal do caractere à sua frente.

º Definição de literal: Um literal é um valor que é expresso **literalmente** no meio do seu código.
Ex: vari=”Meu texto”

Literais ou constantes são os valores que escrevemos em uma forma convencional, cujo valor é óbvio. Em contraste com as variáveis, os literais não mudam de valor.

< (sintaxe para limites de início e fim da palavra)

\< corresponde no início de uma palavra e \> corresponde ao final de uma palavra. Com isso programamos que é para detectar do primeiro caractere da palavra até o ultimo.

O parâmetro -w do grep faz a função de selecionar exatamente a string

echo alex alexa | grep alex* Resultado: **alex alex**a

echo alex alexa | grep -w alex Resultado: **alex** alexa

?

É muito comum ao combinar e extrair texto o *?* (ponto de interrogação) metacaractere que denota *opcionalidade*. Este metacaractere permite que você combine zero ou um dos caracteres ou grupos anteriores. Por exemplo, o padrão *ab? C* corresponderá às strings "abc" ou "ac" porque b é considerado opcional.

printf "colour\ncolor\ncolouur\n" | egrep 'colou?r' Resultado: **colour color**

echo abacate abaate | egrep 'abac?ate' Resultado: **abacate abaate**

echo banana banano | grep -E banana?o? Resultado: **banana banano**

egrep (o mesmo que grep -E) "entende/aceita" expressões regulares.

Em um arquivo chamado nome.txt tem as seguintes strings. Observe como a pluralidade da palavra "arquivo" depende do número de arquivos encontrados. Agora vamos escrever um padrão que use o metacaractere de opcionalidade para corresponder apenas às linhas onde um ou mais arquivos foram encontrados.

Corresponde 1 arquivo encontrado?
Corresponde 2 arquivos encontrado?
Corresponde 24 arquivos encontrado?
Pular Nenhum arquivo encontrado.

cat nome.txt | grep -E "1 arquivo|24?" Resultado: **Corresponde 1 arquivo encontrado?**
Corresponde 2 arquivos encontrado?
Corresponde 24 arquivos encontrado?

Pipe= A barra vertical (pipe) é um caractere ASCII 124, simbolizado por uma barra vertical | ou uma barra vertical interrompida ¦

Foi usado pipe com grep já que estamos procurando mais de uma string

/caractere{N}/ Operador de intervalo

Exatamente N ocorrências do caractere

echo xx | gawk --re-interval '/x{3}/' Resultado: Não apareceu stdont

echo xxx | gawk --re-interval '/x{3}/' Resultado: **xxx**

Um operador de intervalo tem a função de detectar quantas vezes um determinado caractere foi repetido.

--re-interval: Habilita o uso de expressões de intervalo em expressões regulares.

echo xxx xx | grep -E x{3} Resultado: **xxx** xx

º Algumas versões do egrep não requerem o escape das chaves.

Um quatificador é um termo matemático tem o significado de verificar a existência tantas vezes de determinada definição.

Eu uma distro Linux digitar awk ou gawk invoca o GAWK até porque awk é um link simbólico para gawk.

O gawk é o superconjunto que contém todos os recursos do awk e nawk originais. O comando --re-interval nasceu do gawk.

/d Representa os dígitos de 0-9 o mesmo que [0-9]

\d \w não trabalha em expressões regulares POSIX , portanto tem que usar [:digit:] como alternativa.

(string|string) Expressão regular de mais de uma string

localizar planeta Terra
localizar planeta Marte

localizar planetas (Terra|Marte) Com isso pode procurar mais de uma string sem precisar usar duas vezes o comando para localizar.

echo Terra Marte Júpiter | grep -E "(Terra|Marte)" Resultado: **Terra Marte** Júpiter

echo arroz feijão wow | grep -E "arroz|feijão" Resultado: **arroz feijão** wow

echo cat | grep -E "([cb]ats*|[dh]ogs?)" Resultado: **cat**

Corresponderia a cat=gatos ou bat=morcegos | dog=cães ou hog=porcos.
[ ] está combinando as inicias

Localizar parâmetro com grep

echo --ca | egrep '--ca|--no' Resultado: grep: opção não reconhecida “--ca|--no”

Para dizer ao grep para não tratar ' **- -**' como parâmetro do próprio grep temos que fazer assim:

echo --ca | egrep -- '--ca|--no' O -- quer dizer para não interpretar como prefixo as strings posteriores.

Método para não fornecer resultado caso verifique a existência das strings especificadas

echo bana | grep -v 'bana\|nono' Resultado: Não apareceu stdont
echo carro | grep -v 'bana\|nono' Resultado: **carro**

Graças ao parâmetro -v que isso é possível já que significa mostrar o contrário do que o grep sem -v mostraria.

Localizar duas strings com grep sem precisar de ativar suporte a expressão regular , e localizar e substituir com o sed mais de uma string

echo carro pow palma | grep "carro\|pow" Resultado: **carro pow** palma
\| = Barra invertida com pipe é o mesmo que | sozinho. Execute no bash \| e confira.

echo abate kota | sed -e 's/abate/vegano/' -e 's/kota/queijo/'

echo abacate.koia joca.txt sed 's/\(.koia\|.txt\)//g' Resultado: **abacate joca**

Parâmetro s serve como substituição e g para localizar mais de uma string

echo banana.txt aleluia.pow | sed -e "s/.txt//" -e "s/.pow//" Resultado: **banana aleluia**

-e este parâmetro é herdado do unix. É usado para executar mais vezes o próprio subcomando. Oque significa que argumentos para um comando específico devem ser especificados usando -e após o próprio nome do comando e depois do primeiro comando.

Caractere opcional

echo reed | egrep "re(a|e)d" Resultado: **reed**

echo read | egrep rea?e?d Resultado: **read** Mesma função.

Sed Terminal

Substituir palavra por outra usando parâmetro s

Atenção: este método só substitui a primeira palavra por linha

explicação de man sed : substitua a parte correspondente à substituição.

echo meu nome É alexandre | sed 's/alexandre/abacate/'

Substituir todas palavras por outra sem exceção usando o parâmetro g

echo Macarrão é Macarrão | sed 's/Macarrão/batata/g'

Enquanto com s puro só substitui uma palavra por linha o g substitui todas.

1-2 Substituir nomes com barras por exemplo: /usr/bin/ com parâmetro s

echo o diretório /usr/bin/ é usado para executáveis | sed 's/\/usr\/bin/\/etc\/passwd/'

/\ = separa \/= /

como podemos ver é \/usr\/bin\/etc/\passwd/
o parâmetro s do sed precisa de uma barrinha ‘s/’ por isso tem usar \/ ate porque // é igual a //, mas \/ = / . Para entender melhor digite //a no bash agora \/a

Resultado: o diretório /etc/passwd/ é usado para executáveis.

Ler a linha especificada através do número da linha

em filename.txt tem várias linhas vamos ler linha 4
Filename.txt:
Joca
Conha
Jujuba
balão
pera

cat filename.txt | sed -n 4p Resultado: balão

man sed
  -n, --quiet, --silent. O parâmetro n silencia os resultados que não tem a ver com o comando posterior

p se a substituição foi feita, imprime a string. Aparece a string original com a duplicada.

Colocar um espaço entre uma linha e outra

echo -e banana "\npiu" | sed G

Em um arquivo tem várias linhas tem que substituir a string test e mostrár exatamente somente a linha substituída

cat filename.txt | sed -n ‘s/test/testando/p’

-n para imprimir apenas as linhas modificadas.

Como fazer o sed processar uma linha especifica

Em um arquivo filename.txt tem as seguintes linhas:

kanjica
kanjica
kanjica com milho

cat filename.txt | sed “2s/kanjica/banana/” Resultado: kanjica
banana

Substitui a string da segunda linha

Pode ser especificado também de um linha até outra por exemplo: sed ‘2,3s/…../…./’ Segunda linha até terceira.

Deletar uma linha especifica

Em um arquivo chamado filename.txt:

csh
ll
bash
likegeeks

sed ‘2d’ filename.txt Resultado: csh O sed não modifica o arquivo, mas aparece o stdout
bash

d = operador que faz a função de deletar

Deletar de uma linha a outra

sed ‘2,3d’ Deletar da segunda linha até a terceira

Deletar de x linha até a ultima

sed ‘3,$d’

Deletar uma string

Em filename.txt
lolo
lele
lulu
sed ‘/lele/d’ filename.txt Resultado: lolo lulu

echo -e arroto "\nvomito" | sed '/vomito/d' Resultado: arroto Deletou a string vomito com isso só ficou arroto

A string que vai ser deletada tem que está em outra linha já que desta forma procura a string por linha e a deleta por completo a linha.

O operador d tem a função de deletar

Deletar mais de uma string

Em um arquivo filename.txt
lolo
lele
lápis

cat filename.txt | sed ‘/lolo/,/lele/d’ Resultado: lápis

A virgula separa a primeira string da string posterior

Substituir números sequencialmente

echo 123 | sed ‘y/123/456/’

Substitui sequencialmente.

Verificar quantidade de linha

echo 1op | sed "=" Resultado= 1 1op

Em um arquivo filename.txt:

a

b

c

Cat filename.txt | sed “=” Resultado: 1 a 2 b 3 c

Verificar quantas vezes uma string é repetida

echo olá | sed “/olá/=” Resultado: 1 olá

Em um arquivo filename.txt:
pow
pow
pow
pow
cat filename.txt | sed “/pow/=” Resultado 1 pow 2 pow 3 pow 4 pow

Se a string tiver espaçamentos o sed não vai conseguir detectar a quantidade de vezes por exemplo:

pow

pow

pow

cat filename.txt | sed “/pow/=” Resultado: 1 pow 3 pow 5 pow 7 pow Para resolver isso temos que fazer assim:
cat filename.txt | grep [^$] | sed “/pow/=” Resultado: 1 pow 2 pow 3 pow 4 pow

Usando o regex [^$] para não mostrar nenhuma linha vazia

Caso queira mostrar somente os números sem a string tem que usar o operador -n do sed

-n serve para mostar somente oque foi modificado. A string não foi modificado porque antes de usar o sed no arquivo já existia a string portanto oque foi modificado no stdout foram os números.

Substituir string com barras invertidas

echo /bin/bash | sed 's!/bin/bash!/bin/down!' Resultado: /bin/down

O sinal de ! está fazendo a função de separar strings com barras invertidas. Tem uma alternativa a este tipo de comando do sed:

echo /etc/passwd/ | sed "s/\etc\/passwd/\juka\/ept/" Resultado: /juka/ept/

Este exemplo foi especificado anteriormente na seção do sed 1-2

Substituir uma string por outro conteúdo de um arquivo

Em um arquivo .txt com o nome de arquivo1 tem as seguintes linhas

arquivo 1
ABCDE
DATA

E em arquivo2

"arquivo2
Trocado por DATA"

```
sed '/DATA/ {

r arquivo2

d}' arquivo1  Resultado:  arquivo 1
                           ABCDE
                          "arquivo2
                          Trocado por DATA"
```

r tem a função de ler
d tem a função de deletar
{ } serve para agrupar vários comandos juntos.

Ler conteúdo de um arquivo ou comando

echo olá | sed “/^/r” Resultado olá

sed “^/r” filename.txt Resultado: carro

O operador r tem a função de ler.

O ^ em regex significa inicio de linha. Foi usado nestes exemplos já que qualquer conteúdo escrito começa com um inicio de linha.

Determinar até que linha mostrar de um arquivo para somar as linhas de outro arquivo

Em um arquivo .txt chamado file

file linha 1
file linha 2

Em um arquivo .txt chamado filex

linha de filex
linha de filex
linha de filex

sed “2r file” filex Resultado:

linha de filex
linha de filex
file linha 1
file linha 2
linha de filex

Dividi até a segunda linha de filex e o restante depois do conteúdo de file.

r é um operador para imprimir na janela do terminal a string

Deletar mais de uma string por linha

Em um arquivo filename.txt

tui
taa
pkg

cat filename.txt | sed “/tui/,/taa/d” Resultado pkg

echo -e "olá\nbanana\npaçoca" | sed "/olá/,/banana/d" Resultado: paçoca

, tem a função de separar mais de uma string

Pode se fazer assim como alternativa: cat filename.txt | sed -e /tui/d -e /taa/d Resultado pkg
-e serve para o sed conseguir executar mais de um comando sem precisar especificar outra vez o sed

Colocar os comandos do sed em um arquivo e depois fazer o sed utilizar estes comandos

Em um arquivo file.txt:

s/pão/feijão/
s/açucar/tubarão/

echo açucar | sed -f file.txt Resultado tubarão

Incorporar um arquivo para ser modificado na janela do terminal

O arquivo tem o nome de filename.txt:

paçoca
açúcar

sed -f file.txt filename.txt Resultado paçoca tubarão

*O sed chega a substituir a string no stdout, mas não modifica o arquivo

Modificar o arquivo

Para o sed modificar o arquivo tem que se usar o parâmetro -i

Substituir a string depois que foi repetida tantas vezes

Em um arquivo chamado file.txt

olho olho
olho olho

sed "s/olho/doce/2" file.txt Resultado olho doce
olho doce

*Só Substitui quantas vezes a ocorrência da string que tiver na mesma linha

Filtrar a linha especificada usando q e d

Em um arquivo filename.txt

paçoca
pkg
koja

sed “2q;d” filename.txt Resultado: pkg

O parâmetro q imprime as linhas anteriores até a que você especificou. Para imprimir somente a linha especificada tem que usar o operador d para deletar as linhas não especificadas.