Sumário

Introdução

O SED é um editor de fluxo que filtra e transforma texto, fazendo sua alteração/edição. Sua principal função é editar arquivos dinamicamente, o uso de expressão regular dentro do SED o torna uma "linguagem", deixando esse utilitário muito completo, complexo e poderoso.

Você vai usar muito o SED em script, ele vai facilitar sua vida quando você tiver que alterar um determinado dado ou campo em mais de 50 arquivos diferentes e ele vai fazer isso em apenas alguns segundos.

Esse mini tutorial tem como objetivo apresentar algumas funcionalidade do SED, apenas o básico, trazendo para você o que você irá mais precisar para criar bons scripts, com uma linguagem mais automatizada. Lembrando, para que você domine o SED, é necessário ter conhecimento em expressão regular ou ER como chamamos, nesse mini tutorial não vamos abordar isso, mas em certos casos, vou explicar como funciona o comando usando uma expressão regular ou usando um metacaractere.

Opções do SED:

```
-i  Altera o arquivo
-e  Imprime na tela sem alterar o arquivo
```

```
-n  Faz a supressão, mostra só o resultado do comando
 s  Substitui um trecho de texto por outro
 !  Inverte a lógica do comando
 ;  Separador de comandos
 |  Separador de strings
 d  No final deleta
 p  No final imprime
 g  No final (como se usa o d e p) altera todas as ocorrências
 q  Sai do SED , não continua o comando

'\L' - Turn the replacement to lowercase until a '\U' or '\E' is found,
'\l' - Turn the next character to lowercase,
'\U' - Turn the replacement to uppercase until a '\L' or '\E' is found,
'\u' - Turn the next character to uppercase,
'\E' - Stop case conversion started by '\L' or '\U'.
```

Classes do POSIX

```
Classe POSIX    similar            Significado
[:upper:]       [A-Z]              letras maiúsculas
[:lower:]       [a-z]              letras minúsculas
[:alpha:]       [A-Za-z]           maiúsculas/minúsculas
[:alnum:]       [A-Za-z0-9]        letras e números
[:digit:]       [0-9]              números
[:xdigit:]      [0-9A-Fa-f]        números hexadecimais
[:punct:]       [.,!?:...]         sinais de pontuação
[:blank:]       [ \t]              espaço e TAB
[:space:]       [ \t\n\r\f\v]      caracteres brancos
[:cntrl:]       -                  caracteres de controle
[:graph:]       [^ \t\n\r\f\v]     caracteres imprimíveis
[:print:]       [^\t\n\r\f\v]      imprimíveis e o espaço
```

Obs. importante:

Antes de começar, vale ressaltar que o SED não faz a alteração no arquivo, para que isso ocorra, você deve usar o parâmetros `-i`.

Crie o arquivo telefones:

```
Vitor gerald    (011)666-1234
Marcia Dias     (011)555-2112
Enzo            (012)232-3423
Rodrigo Guetes  (022)622-2876
Luiz Rodrigues  (026)555-2124
```

Crie o arquivo telefones_branco:

```
Vitor gerald    (011)666-1234

Marcia Dias     (011)555-2112

Enzo            (012)232-3423

Rodrigo Guetes  (022)622-2876

Luiz Rodrigues  (026)555-2124
```

Inserir dois tabs num range de linhas

```
# Inserindo dois tabs da linha 3 até a linha 5:

linux:~\$ sed '3,5s:^:\t\t:' telefone
Vitor gerald    (011)666-1234
Marcia Dias (011)555-2112
        Enzo        (012)232-3423
        Rodrigo Guetes  (022)622-2876
        Luiz Rodrigues  (026)555-2124
```

Remover espaços duplicados com SED

O *sed* pode fazer o trabalho de alguns comandos, como o `tr -s ' '`, que remove espaços duplicados.

```
# Para isso, vamos precisar usar expressão regular estendidas e avisar ao SED (-
r)

linux:~\$ echo "oi,         tudo bem?" | sed -r 's: +: :g'
oi, tudo bem?

# O mesmo funciona com o 'TR':

echo "oi,         tudo bem?" | tr -s ' '
oi, tudo bem?
```

Então você pergunta, qual é melhor? querendo me dizer que o `tr` pode ser melhor por ter menos caracteres e ser mais leve, mas o `sed` tem e opção `-i` que grava no arquivo, com o `tr` teríamos que jogar essa saída para outro arquivo.

Imprimir usando um intervalo, usando o seq 10, vamos imprimir do intervalo de 2 até 8

```
linux:~\$ seq 10 | sed -n '/2/,/8/p'
2
3
4
5
```

```
6
7
8

linux:~\$ seq 10 | sed -n '2,8 p'
2
3
4
5
6
7
8
```

O segundo exemplo é mais fácil de entender, ele diz que é para imprimir (por isso o `p`) de 2 até 8.

Já o primeiro exemplo também diz isso, mas ele usa `/x/` que faz uma pesquisa (por `x` nesse caso), vamos ver mais exemplos de pesquisa mais adiante, outro comando muito usado para pesquisa de strings e padrões usando regex é a família *grep*: `grep, egrep, fgrep`.

Fazendo a substituição de strings

```
# Primeiro, veremos quais contatos
# possuem '555' no número:

linux:~\$ sed -n '/555/p' telefone
Marcia Dias (011)555-2112
Luiz Rodrigues  (026)555-2124
```

Temos 2 números que tem 555 como dígitos, vamos substituir esses três cincos por três uns (111)

```
linux:~\$ sed 's/555/111/' telefone
Vitor gerald    (011)666-1234
Marcia Dias     (011)111-2112
Enzo            (012)232-3423
Rodrigo Guetes  (022)622-2876
Luiz Rodrigues  (026)111-2124

# Você pode usar a opção '-n' com a opção de printar
# dessa forma, só o que foi alterado será printado na tela.

linux:~\$ sed -n 's/555/111/p' telefone
Marcia Dias (011)111-2112
Luiz Rodrigues  (026)111-2124
```

Vários comandos do SED sem pipe

Uma coisa muito boa no SED é poder rodar vários comando sem a utilização do pipe, segue um exemplo:

```
# Antes da alteração, vou fazer uma pesquisa em 3
# contatos, apenas para você ver como são os números
# antes de trocarmos eles.

linux:~\$ sed -n '/555/p; /232/p; /449/p' telefone
Marcia Dias     (011)555-2112
Enzo            (012)232-3423
Luiz Rodrigues  (026)555-2124
```

Perceba que logo aqui, já pesquisamos por várias strings sem o uso do pipe.

```
# Depois da alteração:

linux:~\$ sed -n 's/555/000/p; s/232/011/p; s/449/555/p' telefone
Marcia Dias (011)000-2112
Enzo        (012)011-3423
Luiz Rodrigues  (026)000-2124
```

O ponto e vírgula `;` é o separador de comandos, lembrando novamente que para cada comando `s/String1/String2/` eu coloquei o '*p*' no final para facilitar a identificação do que foi alterado (ele exibe só o que foi digitado).

Apagar linhas em branco

Uma coisa bem legal é a possibilidade de se excluir linhas em branco, e como fazer isso?

Simples, usando metacaracteres (lembrando que um conjunto de metacaracteres vira expressão regular).

Agora apagando as linhas em branco:

```
linux:~\$ sed '/^$/d' telefone_branco
Vitor gerald    (011)666-1234
Marcia Dias (011)555-2112
Enzo        (012)232-3423
Rodrigo Guetes  (022)622-2876
Luiz Rodrigues  (026)555-2124
```

O circunflexo `^` representa início de uma linha, já o cifrão `$` representa o fim dela, portanto, uma linha em branco tem o início junto com o final, sendo assim é só deletar a mesma.

Fazer pesquisa

Um modelo já mostrado aqui, é como fazer pesquisa e imprimir o resultado, por default, o *sed* faz a pesquisa mas não exibe na tela.

Para que exiba na tela você tem que forçar (usando o '*p*', caso não seja usado irá retornar um erro).

Deletando linhas

Após fazer a pesquisa, você deve especificar alguma opção, como por exemplo excluir a linha:

```
linux:~\$ sed '/555/d' telefone
Vitor gerald    (011)666-1234
Enzo            (012)232-3423
Rodrigo Guetes  (022)622-2876

# Desse modo, todas as linhas que tiverem a ocorrência 555 serão deletadas.
```

Printando múltiplas strings

Como já mostrado aqui, vamos pesquisar mais de um conteúdo no arquivo e vamos imprimir esse resultado, para isso temos duas formas, uma mais prática e uma menos prática mas que funciona super bem.

Usando separador de string:

```
linux:~\$ sed -n '/Enzo\|Vitor\|Luiz/p' telefone
Vitor gerald    (011)666-1234
Enzo            (012)232-3423
Luiz Rodrigues  (026)555-2124
```

Aqui usamos o `\|` (contra barra + pipe) e assim podemos colocar múltiplas strings para serem pesquisadas e exibidas na tela, isso poupa muito tempo quando você precisa pegar mais de um dado.

Usando separador comandos:

```
linux:~\$ sed -n '/Enzo/p; /Vitor/p; /Luiz/p' telefone
Vitor gerald    (011)666-1234
Enzo            (012)232-3423
Luiz Rodrigues  (026)555-2124
```

Dessa forma, temos o mesmo resultado, mas ao invés de pesquisarmos tudo de uma vez, estamos pesquisando aos poucos, primeiro roda um comando que pesquisa por Enzo, depois roda outro comando que pesquisa por Vitor e por último, um outro comando que pesquisa por Luiz.

Vê como é melhor rodar tudo em apenas um comando? isso economiza tempo de digitação e deixa o código mais bonito, além de diminuir a chance de erro por rodar vários comandos.

Você deve pensar sempre além disso, aqui o resultado só foi impresso, mas você poderia deletar, substituir, inserir dados, as opções são inúmeras, basta você adequar a suas necessidades.

Inserir dados no início e fim

No *sed*, nós podemos pesquisar por um dado e depois inserir uma informação antes ou depois do que foi pesquisado, como assim?

Pense que você precisa adicionar uma tag depois de um endereço de telefone, essa tag informa que esse número está errado e precisa ser acertado, assim, quem ver esse telefone vai saber que o número está errado, e como fazemos isso?

Para fazer isso vamos usar o `&` (ec comercial), ele meio que faz um *append*, usando o *sed* como substituição.

```
linux:~\$ sed 's/2876$/& - desatualizado/' telefone
Vitor gerald    (011)666-1234
Marcia Dias     (011)555-2112
Enzo            (012)232-3423
Rodrigo Guetes  (022)622-2876 - desatualizado
Luiz Rodrigues  (026)555-2124
```

Observe que como colocamos o dado após o ec comercial `&`, o dado foi adicionado depois do número pesquisado, mas e se colocarmos antes, então o dado será adicionado antes, veja abaixo:

```
linux:~\$ sed 's/2876$/ desatualizado - &/' telefone
Vitor gerald    (011)666-1234
Marcia Dias     (011)555-2112
Enzo            (012)232-3423
Rodrigo Guetes  (022)622- desatualizado - 2876
Luiz Rodrigues  (026)555-2124
```

Coloquei a tag com espaço no início porque se não, o desatualizado estaria grudado com o traço `-` do número telefônico e coloquei no final para ver que a tag ficou antes do número pesquisado.

Colocar todas as linhas numa só

```
linux:~\$ sed ':a;$!N;s/\n//;ta;' telefone
Vitor gerald    (011)666-1234Marcia Dias    (011)555-2112Enzo       (012)232-
3423Rodrigo Guetes (022)622-2876Luiz Rodrigues (026)555-2124
```

Pesquisando e imprimindo ocorrências

Por padrão o *sed* imprime a linha inteira da ocorrência:

```
linux:~\$ sed -n '/Luiz\|Enzo/p' telefone
Enzo            (012)232-3423
Luiz Rodrigues  (026)555-2124
```

Imprimindo apenas uma linha

```
linux:~\$ sed -n '4p' telefone
Rodrigo Guetes  (022)622-2876
```

Imprimindo um range de linhas

```
linux:~\$ sed -n '3,5p' telefone
Enzo            (012)232-3423
Rodrigo Guetes  (022)622-2876
Luiz Rodrigues  (026)555-2124
```

Adicionando uma cerquilha no início de todas as linhas

```
linux:~\$ sed 's/^/#/' telefone
#Vitor gerald   (011)666-1234
#Marcia Dias    (011)555-2112
#Enzo       (012)232-3423
#Rodrigo Guetes (022)622-2876
#Luiz Rodrigues (026)555-2124
```

Adicionando uma cerquilha no fim de todas as linhas

```
linux:~\$ sed 's/$/#/' telefone
Vitor gerald    (011)666-1234#
Marcia Dias     (011)555-2112#
Enzo            (012)232-3423#
Rodrigo Guetes  (022)622-2876#
Luiz Rodrigues  (026)555-2124#
```

Adicionando uma cerquilha da linha 2 até a linha 5 (no início)

```
linux:~\$ sed '3,5s/^/#/' telefone
Vitor gerald    (011)666-1234
Marcia Dias     (011)555-2112
#Enzo           (012)232-3423
#Rodrigo Guetes (022)622-2876
#Luiz Rodrigues (026)555-2124
```

Substituir 2 por ' Mudou ' nas linhas que tiverem 011

```
linux:~\$ sed -n '/011/ s/2/ Mudou /p' telefone
Vitor gerald    (011)666-1 Mudou 34
Marcia Dias     (011)555- Mudou 112
```

Contando linhas

```
linux:~\$ sed -n '$=' telefone
5
```

Adicionando uma nova linha depois do match, usando o ‘a’

```
linux:~\$ sed '/011/a ola' telefone
Vitor gerald    (011)666-1234
ola
Marcia Dias     (011)555-2112
ola
Enzo            (012)232-3423
Rodrigo Guetes  (022)622-2876
Luiz Rodrigues  (026)555-2124
```

Dessa forma, toda vez que o *sed* achar o 011, ele vai adicionar uma nova linha e colocar ‘ola’ nessa linha, essa linha é adicionada abaixo do resultado.

Adicionando uma nova linha antes do match, usando o ‘i’

```
linux:~\$ sed '/011/i ola' telefone
ola
Vitor gerald    (011)666-1234
ola
Marcia Dias     (011)555-2112
Enzo            (012)232-3423
Rodrigo Guetes  (022)622-2876
Luiz Rodrigues  (026)555-2124
```

Dessa forma, toda vez que o *sed* achar o 011, ele vai adicionar uma nova linha e colocar ‘ola’ nessa linha, essa linha é adicionada em cima do resultado.

Modificando uma linha com o ‘c’

```
linux:~\$ sed '/011/c ola' telefone
ola
ola
Enzo            (012)232-3423
Rodrigo Guetes  (022)622-2876
Luiz Rodrigues  (026)555-2124
```

Dessa forma, toda vez que o sed achar o 011, ele vai pegar toda a linha existente e vai substituir pela string que você colocar lá.

Convertendo para uppercase (maiúscula)

```
linux:~\$ echo "esse texto esta em maiusculo" | sed -r 's/(.*)/\U\1/g'
ESSE TEXTO ESTA EM MAIUSCULO
```

Aqui estamos criando um grupo `()` e nesse grupo estamos pegando todos os caracteres `.*`, depois fazemos a substituição de tudo o que está no grupo `\u` chamando esse grupo `\1`, o contra barra 1 é um retrovisor, eles servem para fazer chamadas em grupos criados.

Convertendo para lowercase (minúscula)

```
linux:~\$ echo "ESSE TEXTO ESTA EM MINUSCULO" | sed -r 's/(.*)/\L\1/g'
esse texto esta em minusculo
```

Aqui estamos criando um grupo `()` e nesse grupo estamos pegando todos os caracteres `.*`, depois fazemos a substituição de tudo o que está no grupo `\L` chamando esse grupo `\1`, o contra barra 1 é um retrovisor, eles servem para fazer chamadas em grupos criados.

Exibindo número da linha do match

```
linux:~\$ sed -n '/011/=' telefone
1
2
```

Essa opção é semelhante ao `grep -n`, e só exibe o número da linha, mas caso queira exibir também a linha, pode usar uma cadeia de comando, e mandar printar `sed -n '/021/{=;p}' sed_ex.txt`.

```
linux:~\$ sed -n '/011/{=;p}' telefone
1
Vitor gerald    (011)666-1234
2
Marcia Dias (011)555-2112
```