Introdução às Expressões Regulares

Mario Luiz Bernardinelli (mariolb@gmail.com)

August 24, 2010

Abstract

Regular Expressions (ERs) are special text strings for describing a search pattern. They are very powerful tool for programmers and system administrators.

Programmers can use regular expressions to check fields in user inputs, like HTML forms.

System administrators can take the poweful of ERs to search for string patterns quickly through all system logs, or they can be used to look for patterns over the network traffic using tools like ngrep.

So, many programmer and administration search tasks can be performed more quickly using regular expressions.

1 O que são e para que servem?

Expressões regulares são utilizadas para especificar padrões de texto. Uma expressão regular é composta por caracteres comuns e também por alguns caracteres com funções especiais que, quando agrupados entre si, formam uma sequência ou expressão.

Essa expressão é interpretada como uma regra e é utilizada no processamento de uma entrada de dados em formato texto, produzindo como saída apenas os dados que casarem com esta regra.

Usando expressões regulares é possível:

- Procurar um texto sem que se saiba exatamente como ele é, mas que se saiba algumas de suas variações.
- Procurar trechos específicos do texto, como o início ou final de linhas.
- Especificar um padrão complexo de busca de sequências de caracteres.

As expressões regulares são montadas em pequenos blocos sequenciais que, se agrupados, permitem o casamento com padrões complexos.

> **DICA:** As expressões regulares (ER) são implementadas de forma diferente em diversos aplicativos. Assim, uma ER que funcione no egrep, pode ter que ser modificada para funcionar no *sed*, *Java Script*, *Oracle*, *PostgreSQL*, *Awk*, *MySQL*, *SQLServer* etc. O objetivo principal aqui é aprender como as ERs funcionam, com alguns exemplos utilizando ferramentas como o egrep. Para cada ferramenta que for utilizar, o administrador (ou programador) deve adaptar a ER de acordo com as características da ferramenta utilizada.

2 Como surgiram?

As expressões regulares têm suas origens nos estudos sobre autômatos e linguagens formais. A teoria dos autômatos é o estudo de máquinas abstratas (os autômatos) e os problemas que

elas são capazes de resolver. Um autômato discreto é um modelo matemático para uma máquina de estado finito (FSM - Finite State Machine). Uma máquina de estado finito é normalmente executada ciclicamente e a cada ciclo é executado apenas um de seus estados internos. Ao final de execução de um estado, a própria máquina de estados pode definir o novo estado a ser executado no próximo ciclo.

Uma linguagem formal é, basicamente, um conjunto de palavras e símbolos que definem as regras de uma linguagem, ou seja, uma linguagem formal define os aspectos sintáticos de uma linguagem.

Em 1950, o matemático Stephen Cole Kleene descreveu os modelos de autômatos usando uma notação matemática chamada conjuntos regulares.

Algum tempo depois, Ken Thompson (um dos pastores do Unix e um dos criadores da linguagem C) utilizou a notação definida por Kleene na construção de uma ferramenta de pesquisas de textos no editor de textos QED. Depois disso, ele adicionou esta mesma capacidade ao editor "ed", iniciando a popularidade das expressões regulares no mundo Unix. Aliás, o comando grep, uma das ferramentas de pesquisa de textos em arquivos mais conhecidas, tem seu nome derivado do formato do comando de pesquisas utilizado no editor ed: g/re/p, onde "re" é um acrônimo para Regular Expression (expressão regular).

3 Como são compostas?

As expressões regulares são compostas por caracteres de uso corrente (letras e números) e por alguns caracteres com significados especiais. Estes caracteres especiais são chamados metacarateres. Os metacaracteres adicionam muito poder às expressões regulares, pois permitem definir padrões de pesquisa sem que se saiba, por exemplo, exatamente o que deve ser procurado.

Há vários tipos de metacaracteres:

- Metacaracteres Representantes
- Metacaracteres Quantificadores
- Metacaracteres Âncoras
- Outros

4 Metacaracteres tipo Representante

Metacaractere	**Nome**	**Função**
.	Ponto	Um caractere qualquer
[...]	Lista	Lista de caracteres permitidos
[^...]	Lista negada	Lista de caracteres não permitidos

4.1 Ponto (.)

O uso do metacaractere ponto (.) numa expressão regular casa com qualquer caractere.

Exemplos: considere o arquivo `ponto.txt` com o seguinte conteúdo:

```
1 Não
2 Nao
3 ovo
4 oco
5 toco
6 Toco
7 o organograma...
```

Usando o `grep`, pesquise pela palavra "**Não**", mesmo que escrita errada (sem o til):

```
grep --color "o.o" ponto.txt
Nao
Não
```

Agora pesquise por qualquer sequência que possua duas vogais "**o**" separadas por um caractere qualquer:

```
grep --color "o.o" ponto.txt
ovo
oco
toco
Toco
o organograma...
```

4.2 Usando o valor literal de um metacaractere

Eis uma pergunta frequente: como procurar pelo valor literal de um metacaractere numa ER se ele representa um metacaractere (e não o seu valor literal)?

A resposta é simples assim: para utilizar o valor literal de um metacaractere numa ER, é preciso utilizar uma sequência de escape que, neste caso, consiste em preceder o caractere por uma contrabarra (\).

Por exemplo, para construir uma ER que case com qualquer letra minúscula, número ou o caractere ponto(.), poderíamos escrevê-la assim:

```
[a-z0-9\.]
```

4.3 Lista ([...])

Uma lista armazena os caracteres permitidos. Assim, por exemplo, a lista [aeiou] casa apenas com as vogais minúsculas.

Exemplo: Considere o seguinte conteúdo do arquivo lista.txt:

```
aeiou
abedefgh
Bernardo
Bruno
bcdfg
```

A expressão [**aeiou**] casa com os seguintes caracteres:

```
grep --color [aeiou] lista.txt
aeiou
abedefgh
Bernardo
Bruno
```

Lembre-se que o grep irá mostrar qualquer linha que possua pelo menos um elemento da lista.

> **DICA:** Para procurar sequências de letras maiúsculas e minúsculas, não use a lista [**A-z**], pois esta faixa inclui outros caracteres que não letras (consulte uma tabela ASCII para maiores detalhes).

4.4 Lista negada ([^...])

Ao contrário da lista, a lista negada armazena os caracteres não permitidos.

Exemplo: considerando o exemplo anterior, a pesquisa pela lista [^**aeiou**] resultaria:

```
grep --color [^aeiou] lista.txt
abedefgh
Bernardo
Bruno
bcdfg
```

5 Metacaracteres tipo Quantificador

Os metacaracteres quantificadores servem para indicar o número de repetições permitidas para a entidade imediatamente anterior a eles.

Os metacaracteres quantificadores são os seguintes:

Metacaractere	**Nome**	**Função**
?	Opcional	Zero ou um
*	Asterisco	Zero, um ou mais
+	Mais	Um ou mais
{n,m}	Chaves	De n até m

5.1 Opcional (?)

O metacaractere opcional (?) casa nenhuma ou uma vez a entidade imediatamente anterior a ele.

Exemplo: considere a ER "**ondas?**". Esta ER casa com a palavra "**onda**" e também com a palavra "**ondas**".

> **DICA:** Os metacaracteres quantificadores relacionam-se apenas com a entidade imediatamente anterior a eles.

Para utilizarmos os quantificadores com mais de um caractere, basta inseri-los numa lista. Por exemplo, considere a sequência de palavras:

```
fala fala! falar falam falaram
```

Vejamos o que casa com as expressões regulares a seguir:

ER	Casa com
`fala[!r]`	`fala fala! falar falam falaram`
`fala[!r]?`	`fala fala! falar falam falaram`

DICA: Para usar o grep com ER estendidas, utilize o comando `egrep`, pois ele permite o uso de metacaracteres sem a necessidade de escapes (\+, por exemplo).

5.2 Qualquer coisa (*)

O metacaractere asterisco (*) é utilizado quando uma entidade da ER pode ou não existir. Assim:

Expressão	Casa com
7*0	0, 70, 770, 7770 ...
bi*p	bp, bip, biip, biiip ...
b[ip]*	b, bi, bip, biiip, bippp, bp ...

Exemplo: considerando a palavra "**batata**", o que casará com a ER "**ba[ta]***"?

1 ba

2 bata

3 batata

4 n.d.a.

Resposta: opção 3. Na verdade, as três primeiras alternativas casam com a ER apresentada, porém, o asterisco tentará sempre casar com o máximo de repetições possível. Portanto, ao usar o asterisco, preste atenção a esta detalhe.

5.3 Um ou mais (+)

O metacaractere mais (+) é praticamente idêntico ao asterisco, exceto pelo fato de que o mais (+) não é opcional, isto é, a entidade anterior deve casar pelo menos uma vez. Assim:

Expressão	Casa com
7+0	70, 770, 7770, ...
bi+p	bip, biip, biiip, ...
b[ip]+	bi, bip, biiip, bippp, bp ...

Teste as ERs acima com o comando:

```
1 echo "b bip  biip  bpiii biiip bppppppppp" | egrep  --color "b[ip]+"
```

Troque as ERs conforme o exemplo.

5.4 Chaves ({})

Este metacaractere define o número de vezes que a entidade anterior deve repetir-se.

Assim, por exemplo:

Metacaractere	Repetições	Comentário
{1,3}	De 1 a 3	
{3,}	3 ou mais	
{0,3}	0 a 3 (ou, até 3)	
{3}	Exatamente 3	
{1}	Exatamente 1	
{0,1}	Zero ou 1	Igual ao opcional
{0,}	Zero ou mais	Igual ao asterisco
{1,}	Um ou mais	Igual ao mais

Considere o seguinte comando:

```
echo "12345 1223334444" | egrep --color "expressão"
```

Onde o termo **expressão** deve ser substituído pela ER do exemplo, teremos:

ER	Resultado	Comentário
1{1}	12345 1223334444	
1{2,3}		Número de repetições insuficiente no texto
2{1,1}	12345 122333444 4	A ER sempre casa o máximo possível
3{2,3}	12345 1223334444	Duas até 3 repetições
3{2,}	12345 1223334444	Dois ou mais repetições
4{2,3}	12345 12233344 44	Dois ou mais repetições

6 Metacaracteres tipo Âncora

Os metacaracteres tipo âncora servem para marcar uma posição específica na linha.

Os metacaracteres do tipo âncora são os seguintes:

Metacaractere	Nome	Função
^	Circunflexo	Início da linha
$	Cifrão	Fim da linha
\b	Borda	Início ou fim de palavra

DICA: Os metacaracteres tipo âncora não podem ser quantificados, portanto, os metacaracteres mais, asterisco e as chaves não têm influência sobre âncoras.

6.1 Início (^)

O metacaractere de início (^) marca o início de uma linha.

Exemplo: a ER "**^123**" só casa com as sequências "**123**" que estiverem no início da linha.

Observe que o circunflexo é também utilizado para marcar listas negadas. Neste caso, o circunflexo dentro da lista (como primeiro caractere da lista) indica uma lista negada. Agora, se o circunflexo estiver fora da lista, então ele indicará que a lista deve estar no início da linha. Portanto:

ER	Significado
^[0-9]	A partir do início da linha, procure por um número.
^[^0-9]	Procuramos por linhas que não começam com número.

6.2 Fim de linha ($)

O metacaractere cifrão ($) marca o final da linha e só é válido no final de uma ER.

Assim:

ER	Significado
[0-9]$	Indica que a linha deve terminar com um número.
^$	Indica uma linha vazia.

6.3 Borda (\b)

O metacaractere borda marca os limites de uma palavra e é útil para casar palavras exatas.

Exemplos:

ER	Casa com
dia	dia, diafragma, melodia, radial, bom-dia
\bdia	dia, diafragma, bom-dia
dia\b	dia, melodia, bom-dia
\bdia\b	dia, bom-dia

Teste os exemplos apresentados com os comandos:

```
1 echo "dia diafragma melodia radial bom-dia" | egrep --color "dia"
2 echo "dia diafragma melodia radial bom-dia" | egrep --color "\bdia"
3 echo "dia diafragma melodia radial bom-dia" | egrep --color "dia\b"
4 echo "dia diafragma melodia radial bom-dia" | egrep --color "\bdia\b"
```