

Expressões regulares em Perl

Introdução

José Pedro Oliveira
(jpo@di.uminho.pt)

Grupo de Sistemas Distribuídos
Departamento de Informática
Escola de Engenharia
Universidade do Minho

Administração de Sistemas
2006-2007



Conteúdo

- Expressões regulares
 - Meta-caracteres
 - Classes de caracteres
 - Âncoras
 - Quantificadores
 - Variáveis
- Operadores
 - Operador de binding
 - Operador de match
 - Operador de substituição
- Funções
 - Função split
 - Função join



José Pedro Oliveira Expressões regulares em Perl

Expressões regulares

Expressões Regulares

Expressão Regular

Uma expressão regular é uma string que descreve um padrão. Em Perl, os padrões descritos por expressões regulares são utilizados para:

- procurar/validar strings,
- extrair sub-strings,
- realizar operações de pesquisa e substituição.

Nota

Por omissão as expressões regulares são gananciosas (*greedy*), isto é, tentam sempre fazer o maior match possível.



José Pedro Oliveira Expressões regulares em Perl

Expressões regulares

Classes de caracteres

Classes de caracteres

Classes de caracteres

- São definidas através da construção [...]
 - /[aA]bc/
- O caracter '-' é usado para definir sequências de caracteres (ranges)
 - /[a-zA-Z]/ # qualquer letra
 - /[0-9]/ # qualquer dígito
- O caracter '-' por ser incluído numa classe através da sua protecção com uma backslash (\-)
- Se o primeiro caracter da sequência entre parênteses rectos for um '^', criam-se classes de caracteres negadas
 - /[^A-Z]/ # qualquer caracter que não seja uma letra maiúscula



José Pedro Oliveira Expressões regulares em Perl

Expressões regulares

Âncoras

Âncoras

Âncoras

Asserção	Significado
^	Início da string (ou linha se /m)
\$	Fim da string (ou linha se /m)
\b	Fronteira de palavra (entre \w e \W)
\B	Negação de \b
\A	Início da string
\Z	Fim da string ou antes de \n
\z	Fim da string
\G	Continua a partir da posição prévia de m//g



José Pedro Oliveira Expressões regulares em Perl

José Pedro Oliveira Expressões regulares em Perl

Expressões regulares

Meta-caracteres

Meta-caracteres

Meta-caracter	Significado
\	Protege o caracter a seguir (retira significado especial)
.	Faz match com qq caracter excepto newline (a não ser que a opção /s seja usada)
^	Faz match com o início da string (ou da linha se a opção /m for usada)
\$	Faz match com o fim da string (ou da linha se a opção /m for usada)
*	Faz match do elemento precedente 0 ou mais vezes
+	Faz match do elemento precedente 1 ou mais vezes
?	Faz match do elemento precedente 0 ou 1 vez
{...}	Especifica o número de ocorrências
[...]	Faz match com um qualquer caracter especificado dentro dos parênteses rectos
(...)	Permite agrupar expressões regulares
	Faz match com a expressão que o precede ou que lhe sucede



José Pedro Oliveira Expressões regulares em Perl

Expressões regulares

Classes de caracteres

Classes de caracteres

Sequências de escape pré-definidas

Seq. Escape	Significado
\d	[0-9]
\D	[^0-9]
\w	[a-zA-Z0-9_]
\W	[^a-zA-Z0-9_]
\s	[\t\n\r\f]
\S	[^\t\n\r\f]



José Pedro Oliveira Expressões regulares em Perl

Expressões regulares

Quantificadores

Quantificadores

Quantificadores

Máximo	Mínimo	Significado
{n,m}	{n,m}?	Pelo menos n mas menos de m vezes
{n,}	{n,}?	Pelo menos n vezes
{n}	{n}?	Exactamente n vezes
*	*?	0 ou mais vezes (o mesmo que {0,})
+	+	1 ou mais vezes (o mesmo que {1,})
?	??	0 ou 1 vez (o mesmo que {0,1})

Exemplo

```
1 $_ = 'aaaa';
2 print "$&\n" if m/a+/; # match maximo (greedy)
3 print "$&\n" if m/a+?/; # match minimo
```



José Pedro Oliveira Expressões regulares em Perl

Variáveis

Memórias

Os parênteses servem não só para agrupar elementos como também para memorizar os padrões por eles emparelhados. Cada par de parênteses é guardado numa variável read-only especial.

- Dentro de um padrão cada elemento delimitado por parênteses guarda o seu match numa variável numerada a partir de 1. Estes valores podem ser re-utilizados através de `\1`, `\2`, ...
- Fora de um padrão, as variáveis de match podem ser re-utilizadas através de `$1`, `$2`, ...

Nota

A criação de memórias pode ser evitada recorrendo à seguinte construção: `(?: ...)`, que pode ser interpretada como "agrupa mas não captures".

Conteúdo

- 1 Expressões regulares
 - Meta-caracteres
 - Classes de caracteres
 - Âncoras
 - Quantificadores
 - Variáveis
- 2 Operadores
 - Operador de binding
 - Operador de match
 - Operador de substituição
- 3 Funções
 - Função split
 - Função join

Operador de binding

Exemplo 1

```
1 $_ = '123';
2
3 if (m^[0-9]+$/) {
4     print "Apenas digitos\n";
5 } else {
6     print "Nao sao so digitos\n";
7 }
```

Exemplo 2

```
1 my $input = '123';
2
3 print "Apenas digitos\n" if ($input =~ m/^\d+$/);
4
5 print "Nao sao so digitos\n" if $input !~ m/^\d+$/;
```

Operador de match

Exemplo

```
1 $source = "1/10/2006"; # dia/mes/ano
2
3 if ($source =~ m#(\d{1,2})/(\d{1,2})/(\d{4})#) {
4
5     $dia = $1;
6     $mes = $2;
7     $ano = $3;
8
9     print "$ano-$mes-$dia\n";
10 }
```

Variáveis

Outras variáveis

<code>\$+</code>	Retorna o match do último par de parênteses
<code>\$&</code>	Retorna a string de match
<code>\$'</code>	Retorna tudo antes da string de match (backtick)
<code>\$`</code>	Retorna tudo depois da string de match

Operador de binding

Operador `=~`

O operador `=~` realiza a pesquisa regex contra a string alvo especificada à esquerda em vez de a realizar contra `$..`

Exemplo

```
$resposta =~ m/~sim$/i
```

Operador `!~`

Inverte o sentido da operação de match.

Operador de match

Synopsis

```
m/PADRÃO/opções
/PADRÃO/opções
```

Retorna

Depende do contexto; num contexto escalar retorna true/false.

Opções

- `c` - Não faz reset em caso de falha (`/g`)
- `g` - Procura todas as ocorrências
- `i` - Detecta padrões de uma forma case-insensitive
- `m` - Trata a string como múltiplas linhas
- `o` - Compila o padrão uma única vez
- `s` - Trata a string como uma linha única
- `x` - Utiliza expressões regulares extendidas

Operador de match

Exemplo 2

```
1 #!/usr/bin/perl -w
2 use strict;
3
4 my ($dia, $mes, $ano);
5 my $source = "1/10/2006"; # dia/mes/ano
6
7 ($dia, $mes, $ano) =
8     $source =~ m#(\d{1,2})/(\d{1,2})/(\d{4})#;
9
10 print "$ano-$mes-$dia\n";
```

Operador de match

Exemplo 3

```

1 my $str = 'abc def ghi';
2
3 if ($str =~ m/def/) {
4     print "String de match: '$&'\n";      # 'def'
5
6     print "Tudo antes do padrao: '$'\n";  # 'abc '
7
8     print "Tudo depois do padrao: '$'\n"; # ' ghi'
9
10 }
11

```



Operador de substituição

Synopsis

s/PADRÃO/SUBSTITUIÇÃO/opções

Opções

- e - Avalia o termo de substituição como uma expressão Perl
- g - Procura todas as ocorrências
- i - Detecta padrões de uma forma case-insensitive
- m - Trata a string como múltiplas linhas
- o - Compila o padrão uma única vez
- s - Trata a string como uma linha única
- x - Utiliza expressões regulares extendidas



Operador de substituição

Exemplo de conversão de formato de datas

```

1 my $str = '... 1/1/2000 ...';
2 my $regex = '(\d{1,2})/(\d{1,2})/(\d{4})';
3
4 $str =~ s#$regex#sprintf "%04d-%02d-%02d", $3, $2, $1#e;
5
6 print "$str\n";      # '... 2000-01-01 ...'

```

Exercício

Converter para o formato ISO a data da seguinte string
Oct 17 11:49:07 hostname bla bla bla.

Formato ISO para datas: YYYY-MM-DD
(Fonte: <http://www.w3.org/TR/NOTE-datetime>).



Função split

Synopsis

```

split /PADRAO/, EXPR, LIMIT
split /PADRAO/, EXPR
split /PADRAO/
split

```

Descrição

Subdivide a string EXPR utilizando a expressão regular PADRAO como separador das sub-strings



Função join

Synopsis

```
join EXPR, LISTA
```

Descrição

Concatena as strings da lista usando EXPR como separador (EXPR não é uma expressão regular)

Exemplo

```

$resultado = join ':', $str1, $str2, $str3;
$resultado = join ':', @campos;

```



Conteúdo

- 1 Expressões regulares
 - Meta-caracteres
 - Classes de caracteres
 - Âncoras
 - Quantificadores
 - Variáveis
- 2 Operadores
 - Operador de binding
 - Operador de match
 - Operador de substituição
- 3 Funções
 - Função split
 - Função join



Função split

Exemplos

```

1 $source = "root:0:/root:/bin/shell";
2
3 @campos = split /\:/, $source;
4
5 ($login, $uid, $home, $sh) = split /\:/, $source;
6
7 ($login, $uid, @resto) = split /\:/, $source;
8
9 ($login, $uid) = (split /\:/, $source)[0,1];

```



Conteúdo

- 4 Exemplos
 - Alterar terminador de registo
 - Validar endereços ethernet
 - Gerar documento HTML



Exemplo: conversão de formato DOS para UNIX

```

1 while(<>) {
2     s/\r//;
3     print;
4 }

```

Exemplo: conversão de formato DOS para UNIX

```
$ perl -i.bak -pe "s/\r//;" ficheiro ...
```



```

1 #!/usr/bin/perl -w
2 use strict;
3 my ($login, $uid, $home, $sh);
4
5 print <<..HTML..;
6 <DOCTYPE HTML PUBLIC
7     "-//W3C//DTD HTML 4.01//EN"
8     "http://www.w3.org/TR/html4/strict.dtd">
9 <html>
10 <head>
11     <title>Utilizadores </title>
12 </head>
13 <body>
14 <table border="1">
15 <tr><th>login </th><th>uid </th>
16 <th>home</th><th>shell </th></tr>
17 ..HTML..

```

Comando killuserproc

Implementar um script perl que permita matar todos os processos de um dado utilizador. O nome do utilizador deverá ser especificado na linha de comando.

Sugestões

- capturar o output do comando `ps aux`

```
$ perldoc perlopentut
$ perldoc -f open
```
- utilizar a função `kill`

```
$ perldoc -f kill
$ kill -l # lista de sinais
```



Referências

```

1 #!/usr/bin/perl -w
2 use strict;
3
4 my @macs = (
5     '00:01:02:03:04:05',
6     '00-01-02-03-04-aA',
7     '00:01-02-03-04-aa', # separadores inconsistentes
8     '00-01-02-03-04-AG' # simbolo hexadecimal invalido
9 );
10 my $hex = '[0-9a-fA-F]{2}'; # octeto
11
12 for (@macs) {
13     if (m/$hex([ -:])$hex\$hex\$hex\$hex\$hex/) {
14         print "[$_] OK. Separador: '$1'.\n";
15     } else {
16         print "[$_] NOK.\n";
17     }
18 }

```



```

18
19 open (my $f, '<', '/etc/passwd') or die "$!";
20 while (<$f>) {
21     chomp;
22     ($login, $uid, $home, $sh) = (split /:/)[0,2,-2,-1];
23     $sh =~ s#\bbash\b#<b>bash</b>#;
24     print "<tr><td>$login </td><td>$uid </td> .
25         <td>$home </td><td>$sh </td></tr>\n";
26 }
27 close $f;
28
29 print <<..HTML..;
30 </table>
31 </body>
32 </html>
33 ..HTML..

```



Extrair entradas de ficheiros de logs por datas

Implementar um script perl que permita extrair entradas do ficheiro de log `/var/log/messages` correspondentes a um dado período de tempo. O intervalo de tempo desejado deverá ser especificado na linha de comando.

Sugestões

- converter as datas para formato ISO
- ficheiros de log:
 - Linux: `/var/log/messages`
 - Mac OS X: `/private/var/log/system.log`



Referências

- **Mastering Regular Expressions, Third Edition**
Jeffrey E. F. Friedl
<http://www.oreilly.com/catalog/regex3/>
- `perlre` - Perl regular expressions
`$ perldoc perlre`
- `perlrequick` - Perl regular expressions quick start
`$ perldoc perlrequick`
- `perlretut` - Perl regular expressions tutorial
`$ perldoc perlretut`
- `perlop` - Perl operators and precedence
`$ perldoc perlop`

