A Quick Guide To PERL Regular Expressions

This is a Quick reference Guide for PERL regular expressions (also known as regexps or regexes).
These tools are used to describe text as “motifs” or “patterns” for matching, quoting, substituting or translitterating. Each programming language (Perl, C, Java, Python...) define its own regular expressions although the syntax might differ from details to extensive changes. In this guide we will concentrate on the Perl regexp syntax, we assume that the reader has some preliminary knowledge of Perl programming.

Perl uses a Traditional Nondeterministic Finite Automata (NFA) match engine. This means that it will compare each element of the motif to the input string, keeping track of the positions. The engine choose the first leftmost match after greedy (i.e., longest possible match) quantifiers have matched.

References
For more information on Perl regexps and other syntaxes you can refer to O’Reilly’s book “Mastering Regular Expressions”.

Examples:
The following sentence will be used in all our examples:

```
The ID sp:UBP5_RAT is similar to the rabit AC tr:Q12345
```

Motif finding: match operator m//

EXPR =~ m/MOTIF/cgimosx
EXPR =~ /MOTIF/cgimosx
EXPR !~ m/MOTIF/cgimosx
EXPR !~ /MOTIF/cgimosx

Examples: match any SwissProt ID for a rat protein

```
if ($ex =~ m/\w{2,5}_RAT/) { print “Rat entry\n”; }
```

will match

```
The ID sp:UBP5_RAT is similar to the rabit AC tr:Q12345
```

and as a result print Rat entry.

Options

`cg`	continue after a failure in `/g`
`g`	global matches (matches all occurrences)
`i`	case insensitive
`m`	multiline, allow “`^`” and “`$`” to match with (\n)
`o`	compile MOTIF only once
`s`	single line, dot “`.`” matches new-line (\n)
`x`	ignore whitespace and allow comments “`#`” in MOTIF

Search&Replace: substitution operator s///

EXPR =~ s/MOTIF/REPLACE/egimosx

Example: correct typo for the word rabbit

```
$ex =~ s/rabit/rabbit/g;
```

Here is the content of $ex:

```
The ID sp:UBP5_RAT is similar to the rabbit AC tr:Q12345
```

Example: find and tag any TrEMBL AC

```
$ex =~ s/tr:/trembl_ac=/g;
```

Here is the content of $ex:

```
The ID sp:UBP5_RAT is similar to the rabit AC trembl_ac=Q12345
```

Options

`e`	evaluate REPLACE as an expression
`g`	global matches (matches all occurrences)
`i`	case insensitive
`m`	multiline, allow “`^`” and “`$`” to match with (\n)
`o`	compile MOTIF only once
`s`	single line, dot “`.`” matches new-line (\n)
`x`	ignore whitespace and allow comments “`#`” in MOTIF

Quoting: quote and compile operator qr//

EXPR =~ qr/MOTIF/imosx

Example: reuse of a precompiled regexp

```
$myregexp = qr/\w{2,5}_\w{2,5}/;
if ($ex =~ m/$myregexp/) { print “SwissProtID\n”; }
```

will match:

```
The ID sp:UBP5_RAT is similar to the rabit AC tr:Q12345
```

and as a result will print SwissProtID.

Options

`i`	case insensitive
`m`	multiline, allow “`^`” and “`$`” to match with (\n)
`o`	compile MOTIF only once
`s`	single line, dot “`.`” matches new-line (\n)
`x`	ignore whitespace and allow comments “`#`” in MOTIF

Character classes

`[...]`	Match any one character of a class
`[^...]`	Match any one character not in the bracket
`.`	Match any character (except newline [^\n]) in non single-line mode (/s)
`\d`	Any digit. Equivalent to `[0..9]` or `[[:digit:]]`
`\D`	Any non-digit.
`\s`	Any whitespace. `[ \t\s\n\r\f\v]` or `[[:space:]]`
`\S`	Any non-whitespace.
`\w`	Any word character. `[a-zA-Z0-9_]` or `[[:alnum:_]]`
`\W`	Any non-word character. Warning `\w != \S`

POSIX Character class

`[[:class:]]` class can be any of:
`alnum alpha ascii blank cntrl digit graph lower print punct space upper xdigit`

Special characters

`\a`	alert (bell)
`\b`	backspace
`\e`	escape
`\f`	form feed
`\n`	newline
`\r`	carriage return
`\t`	horizontal tabulation
`\nnn`	octal *nnn*
`\xnn`	hexadecimal *nn*
`\cX`	control character *X*

Repetitions

`?`	Zero or one occurrence of the previous item.
`*`	Zero or more occurrences of the previous item.
`+`	One or more occurrences of the previous item.
`{n,m}`	Match at least `n` times but no more than `m` times the previous item.
`{n,}`	Match `n` or more times
`{n}`	Match exactly `n` times
`{}?`	Non-greedy match (i.e., match the shortest string)

Anchors

`^ or \A`	Match beginning of the string/line
`$ or \Z`	Match end of the string/line
`\z`	End of string in any match mode
`\b`	Match word boundary
`\B`	Match non-word boundary

Capture & Grouping

`(...)`	Group several characters together for later use or capture as a single unit
`|`	Match either subexpressions (equivalent to “OR”)

Example: match any database code in the list

```
$ex =~ m/(sp:|tr:|rs:)/g;
```

will match:

```
The ID sp:UBP5_RAT is similar to the rabit AC tr:Q12345
```

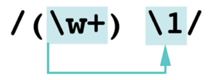
`\n`	Back reference. Match the same as the captured group number *n* that was previously matched in the same MOTIF.
`$n`	Substring of captured group *n*

Example: match several instances with back reference

```
$ex =~ m/(the).+\1/i;
```

will match:

```
The ID sp:UBP5_RAT is similar to the rabit AC tr:Q12345
```

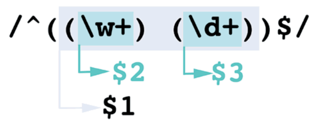


Example: rename any tr:AC to trembl_AC= using a capture

```
$ex =~ s/tr:([[:alnum:]]{6})/trembl_AC=$1/gi;
```

will match:

```
The ID sp:UBP5_RAT is similar to the rabit AC trembl_AC=Q12345
```



Text-span modifiers

`\Q`	Quote following metacharacters until `\E` or end of motif (allow the use of scalars in regexp)
`\u`	Force next character to uppercase
`\l`	Force next character to lowecase
`\U`	Force all following characters to uppercase
`\L`	Force all following characters to lowercase
`\E`	End a span started with `\Q`, `\U` or `\L`

Extended Regexp

`(?#...)`	Substring "..." is a comment
`(?=...)`	Positive lookahead. Match if exists next match (e.g., allow overlapping matches in global mode)
`(?!...)`	Negative lookahead. Match if no next match
`(?<=...)`	Positive lookbehind. Fixed length only.
`(?<!...)`	Negative lookbehind. Fixed length only.
`(?imsx)`	Modify matching options

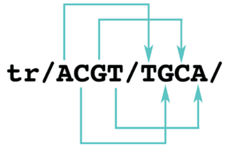
Transliteration: translate operator tr///

EXPR =~ tr/SEARCHLIST/REPLACELIST/cds

Transliteration is not - and does not use - a regular expression, but it is frequently associated with the regexp in PERL. Thus we decided to include it in this guide.

Example: reverse and complement a DNA sequence

```
$DNA = AAATATTTCATCGTACAT;
$revcom = reverse $DNA;
$revcom =~ tr/ACGTacgt/TGCAtgca/;
```



```
The transliteration will produce the following:
print($DNA);
               AAATATTTCATCGTACAT
print($revcom);
               ATGTACGATGAAATATTT
```

Options

`c`	complement REPLACELIST
`d`	delete non-replaced characters
`s`	single replace of duplicated characters

UniCode matches

Perl 5.8 supports UniCode 3.2. However it would be too long to describe all the properties in details here. For more information see "Mastering Regular Expressions".

`\p{PROP}`	Matches a UniCode property
`\P{PROP}`	Matches anything but a UniCode property

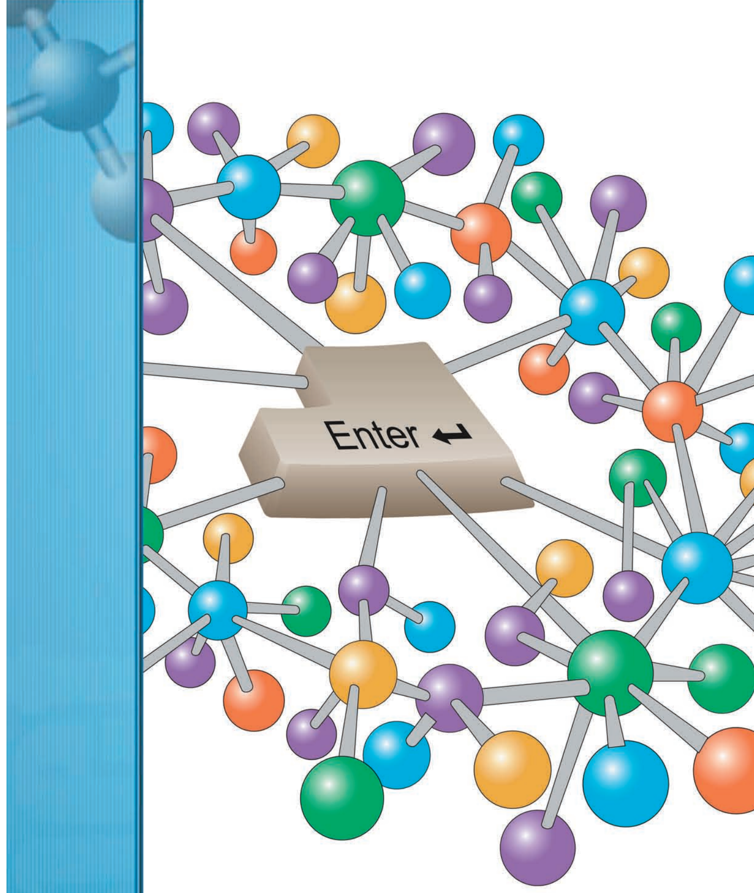
This document was written and designed by Laurent Falquet and Vassilios Ioannidis from the Swiss EMBnet node and being distributed by P&PR Publications Committee of EMBnet.

EMBnet - European Molecular Biology Network - is a bioinformatics support network of bioinformatics support centers situated primarily in Europe. Most countries have a national node which can provide training courses and other forms of help for users of bioinformatics software.

You can find information about your national node from the EMBnet site:

http://www.embnet.org/

A Quick Guide To PERL Regular Expressions
Second edition © 2006



EMBnet

A Quick Guide

RegExp

PERL Regular Expressions