

Collected hints

Exercises

Regular Expressions

- Do not hardcode file name in scripts - use ARGV argument array instead: `open FL, $ARGV[0]` or `die`.
- Do not hardcode dimensions, either!
- Nice convention: use all-capital letters for file handles.
- Indent your programs!
- The date issue 2007-10-10 vs 10 Oct 2007 was *related* to the environment! Remember and protect yourself!
- For large lists push @arr, @list may be better than `@arr=(@arr, @list)`

Regular Expressions

Exercises

Regular
Expressions

- Very powerful method for search, and search-and-replace.
- Compactly express the search algorithms
- As many powerful things, may seem frightening in the beginning.

THERE IS NOTHING TO BE AFRAID OF.

Metacharacters

Exercises

Regular
Expressions

- \ – quoting.
- — – alternative.
- (and) – grouping.
- [– character set.
- { – specify number of repetitions.
- ^ – match *beginning*.
- \$ – match *end*.
- * – repeat 0 or more times.
- + – repeat 1 or more times.
- ? – repeat 0 or once (or non-greedy, if following * or +).
- . – arbitrary character.

Metasymbols

Exercises

Regular
Expressions

Only selected ones. See Table 5-7 for a complete list.

- `\b` matches a word *boundary*.
- `\d` matches any digit character `[0-9]`.
- `\D` matches any non-digit character `[^0-9]`.
- `\s` matches any space character `[\r\t\n]`.
- `\S` matches any non-space character `[^ \r\t\n]`.
- `\w` matches any “word” character (alphanumeric and “_”).
- `\W` matches any non-“word” character.

Example

Exercises

Regular
Expressions

Given following input file:

Low energy: 0.01 determined with error 6e-3

High Energy: +24e4

High Energy2: -4.1E4

Found following estimation: +24D-7

Write regex that matches all the above-mentioned forms of numbers (/...\.d.../. Work in groups of three.

Example

Exercises

Regular
Expressions

Given following input file:

Low energy: 0.01 determined with error 6e-3

High Energy: +24e4

High Energy2: -4.1E4

Found following estimation: +24D-7

Write regex that matches all the above-mentioned forms of numbers (/...\.d.../. Work in groups of three. Possible solution: `m/[-+]?[d+].[?d]*([eEdD]?[-+]?[d]*)?/`

Greedy vs Non-greedy Example

Exercises

Regular
Expressions

```
"exasperate" =~ /e(.*)e/; # xasperat
```

```
"exasperate" =~ /e(.*)e/; # xasp and not rat!
```

Regular expressions are interpreted left-to-right, and
“backtracked” if needed.

Use for Searching

Exercises

Regular
Expressions

`m//;`

- Returns undef/1 in scalar context.

```
if(/Energy/) { ...}
```

- Returns the matched strings in the list context if the expression selects some `@a = ($_ =~ /a(.{3})/);`
- Play with it:

```
perl -n -e '@a = ($_ =~ /a(.{3})/g); print @a, "\n";'
```

Modifiers:

- `g` for *global*.
- `i` for *ignore case*.
- `s` for *single line*, `.` matches `\n`.
- `m` for *multi line*, `^` and `$` match around `\n`.

Special Variables

Exercises

Regular
Expressions

Perl has special variables that store the text before the match, the match itself, and the text following the match:

```
'hot cross buns' =~ /cross/;
print "Matched <$'> $& <$'>\n";
```

```
Matched <hot > cross < buns>
```

Nested Captures vs Clustering

Exercises

Regular
Expressions

Parentheses are used for grouping alternatives *and* for marking strings for extraction:

```
# marking strings for extraction:  
($full, $first, $last) =~ /((\w+)\s*(\w+))/;
```

Grouping alternatives for processing lists:

```
# Johnny hamster Hamlet  
# Amelia cat Furry  
($owner, undef, $animalName)  
    =~ m/(\w+)\s*(cat|dog|hamster)\s*(\w+)/;  
# or explicitly...  
($owner, $animalName) =~ m/(\w+)\s*(?:cat|dog|hamster)\s*(\w+)/;
```

- Regexp can check syntax automatically.

Character Translation

Exercises

Regular
Expressions

Replace characters from one sequence with ones taken from a matching position in another sequence.

```
tr/[A-M] [N-Z] [a-m] [n-z]/[N-Z] [A-M] [n-z] [a-m]/; # ROT13
```

Returns the number of times translation was applied.

Use in Search&Replace

Exercises

Regular
Expressions

Apply search&replace to a string.
Example: remove trailing spaces

```
$a =~ s/\s+$//gm;
```

Ending Comments

Exercises

Regular
Expressions

Reading:

- Chapter 5 (feel free to skip Unicode discussion). *Little Engine that...* explains the details.
- Scan through Chapter 24, up to “Fluent Perl” (pp. 585-607).

Hand in for the next session:

- Practice simple regex search and replace operations.
- Practice more complex regex matching. Think of the characters that delimit the fields to write the regular expression matching entire line.