Useful One-Line Scripts for Perl Dec 03 2013 | version 1.10

-------------------------------- ----------- ------------

Compiled by Peteris Krumins (peter@catonmat.net, @pkrumins on Twitter)

http://www.catonmat.net -- good coders code, great reuse

Latest version of this file is always at:

http://www.catonmat.net/download/perl1line.txt

This file is also available in other languages:

Chinese: https://github.com/vinian/perl1line.txt

Please email me peter@catonmat.net if you wish to translate it.

Perl One-Liners on Github:

https://github.com/pkrumins/perl1line.txt

You can send me pull requests over GitHub! I accept bug fixes,

new one-liners, translations and everything else related.

I have also written "Perl One-Liners Explained" ebook that's based on

this file. It explains all the one-liners here. Get it at:

http://www.catonmat.net/blog/perl-book/

No Starch Press has published "Perl One-Liners" as a real book too:

http://nostarch.com/perloneliners

These one-liners work both on UNIX systems and Windows. Most likely your

UNIX system already has Perl. For Windows get the Strawberry Perl at:

http://www.strawberryperl.com/

Table of contents:

1. File Spacing

2. Line Numbering

3. Calculations

4. String Creation and Array Creation

5. Text Conversion and Substitution

6. Selective Printing and Deleting of Certain Lines

7. Handy Regular Expressions

8. Perl tricks

FILE SPACING

------------

# Double space a file

perl -pe '$\="\n"'

perl -pe 'BEGIN { $\="\n" }'

perl -pe '$\_ .= "\n"'

perl -pe 's/$/\n/'

perl -nE 'say'

# Double space a file, except the blank lines

perl -pe '$\_ .= "\n" unless /^$/'

perl -pe '$\_ .= "\n" if /\S/'

# Triple space a file

perl -pe '$\="\n\n"'

perl -pe '$\_.="\n\n"'

# N-space a file

perl -pe '$\_.="\n"x7'

# Add a blank line before every line

perl -pe 's//\n/'

# Remove all blank lines

perl -ne 'print unless /^$/'

perl -lne 'print if length'

perl -ne 'print if /\S/'

# Remove all consecutive blank lines, leaving just one

perl -00 -pe ''

perl -00pe0

# Compress/expand all blank lines into N consecutive ones

perl -00 -pe '$\_.="\n"x4'

# Fold a file so that every set of 10 lines becomes one tab-separated line

perl -lpe '$\ = $. % 10 ? "\t" : "\n"'

LINE NUMBERING

--------------

# Number all lines in a file

perl -pe '$\_ = "$. $\_"'

# Number only non-empty lines in a file

perl -pe '$\_ = ++$a." $\_" if /./'

# Number and print only non-empty lines in a file (drop empty lines)

perl -ne 'print ++$a." $\_" if /./'

# Number all lines but print line numbers only non-empty lines

perl -pe '$\_ = "$. $\_" if /./'

# Number only lines that match a pattern, print others unmodified

perl -pe '$\_ = ++$a." $\_" if /regex/'

# Number and print only lines that match a pattern

perl -ne 'print ++$a." $\_" if /regex/'

# Number all lines, but print line numbers only for lines that match a pattern

perl -pe '$\_ = "$. $\_" if /regex/'

# Number all lines in a file using a custom format (emulate cat -n)

perl -ne 'printf "%-5d %s", $., $\_'

# Print the total number of lines in a file (emulate wc -l)

perl -lne 'END { print $. }'

perl -le 'print $n=()=<>'

perl -le 'print scalar(()=<>)'

perl -le 'print scalar(@foo=<>)'

perl -ne '}{print $.'

perl -nE '}{say $.'

# Print the number of non-empty lines in a file

perl -le 'print scalar(grep{/./}<>)'

perl -le 'print ~~grep{/./}<>'

perl -le 'print~~grep/./,<>'

perl -E 'say~~grep/./,<>'

# Print the number of empty lines in a file

perl -lne '$a++ if /^$/; END {print $a+0}'

perl -le 'print scalar(grep{/^$/}<>)'

perl -le 'print ~~grep{/^$/}<>'

perl -E 'say~~grep{/^$/}<>'

# Print the number of lines in a file that match a pattern (emulate grep -c)

perl -lne '$a++ if /regex/; END {print $a+0}'

perl -nE '$a++ if /regex/; END {say $a+0}'

CALCULATIONS

------------

# Check if a number is a prime

perl -lne '(1x$\_) !~ /^1?$|^(11+?)\1+$/ && print "$\_ is prime"'

# Print the sum of all the fields on a line

perl -MList::Util=sum -alne 'print sum @F'

# Print the sum of all the fields on all lines

perl -MList::Util=sum -alne 'push @S,@F; END { print sum @S }'

perl -MList::Util=sum -alne '$s += sum @F; END { print $s }'

# Shuffle all fields on a line

perl -MList::Util=shuffle -alne 'print "@{[shuffle @F]}"'

perl -MList::Util=shuffle -alne 'print join " ", shuffle @F'

# Find the minimum element on a line

perl -MList::Util=min -alne 'print min @F'

# Find the minimum element over all the lines

perl -MList::Util=min -alne '@M = (@M, @F); END { print min @M }'

perl -MList::Util=min -alne '$min = min @F; $rmin = $min unless defined $rmin && $min > $rmin; END { print $rmin }'

# Find the maximum element on a line

perl -MList::Util=max -alne 'print max @F'

# Find the maximum element over all the lines

perl -MList::Util=max -alne '@M = (@M, @F); END { print max @M }'

# Replace each field with its absolute value

perl -alne 'print "@{[map { abs } @F]}"'

# Find the total number of fields (words) on each line

perl -alne 'print scalar @F'

# Print the total number of fields (words) on each line followed by the line

perl -alne 'print scalar @F, " $\_"'

# Find the total number of fields (words) on all lines

perl -alne '$t += @F; END { print $t}'

# Print the total number of fields that match a pattern

perl -alne 'map { /regex/ && $t++ } @F; END { print $t }'

perl -alne '$t += /regex/ for @F; END { print $t }'

perl -alne '$t += grep /regex/, @F; END { print $t }'

# Print the total number of lines that match a pattern

perl -lne '/regex/ && $t++; END { print $t }'

# Print the number PI to n decimal places

perl -Mbignum=bpi -le 'print bpi(n)'

# Print the number PI to 39 decimal places

perl -Mbignum=PI -le 'print PI'

# Print the number E to n decimal places

perl -Mbignum=bexp -le 'print bexp(1,n+1)'

# Print the number E to 39 decimal places

perl -Mbignum=e -le 'print e'

# Print UNIX time (seconds since Jan 1, 1970, 00:00:00 UTC)

perl -le 'print time'

# Print GMT (Greenwich Mean Time) and local computer time

perl -le 'print scalar gmtime'

perl -le 'print scalar localtime'

# Print local computer time in H:M:S format

perl -le 'print join ":", (localtime)[2,1,0]'

# Print yesterday's date

perl -MPOSIX -le '@now = localtime; $now[3] -= 1; print scalar localtime mktime @now'

# Print date 14 months, 9 days and 7 seconds ago

perl -MPOSIX -le '@now = localtime; $now[0] -= 7; $now[4] -= 14; $now[7] -= 9; print scalar localtime mktime @now'

# Prepend timestamps to stdout (GMT, localtime)

tail -f logfile | perl -ne 'print scalar gmtime," ",$\_'

tail -f logfile | perl -ne 'print scalar localtime," ",$\_'

# Calculate factorial of 5

perl -MMath::BigInt -le 'print Math::BigInt->new(5)->bfac()'

perl -le '$f = 1; $f \*= $\_ for 1..5; print $f'

# Calculate greatest common divisor (GCM)

perl -MMath::BigInt=bgcd -le 'print bgcd(@list\_of\_numbers)'

# Calculate GCM of numbers 20 and 35 using Euclid's algorithm

perl -le '$n = 20; $m = 35; ($m,$n) = ($n,$m%$n) while $n; print $m'

# Calculate least common multiple (LCM) of numbers 35, 20 and 8

perl -MMath::BigInt=blcm -le 'print blcm(35,20,8)'

# Calculate LCM of 20 and 35 using Euclid's formula: n\*m/gcd(n,m)

perl -le '$a = $n = 20; $b = $m = 35; ($m,$n) = ($n,$m%$n) while $n; print $a\*$b/$m'

# Generate 10 random numbers between 5 and 15 (excluding 15)

perl -le '$n=10; $min=5; $max=15; $, = " "; print map { int(rand($max-$min))+$min } 1..$n'

# Find and print all permutations of a list

perl -MAlgorithm::Permute -le '$l = [1,2,3,4,5]; $p = Algorithm::Permute->new($l); print @r while @r = $p->next'

# Generate the power set

perl -MList::PowerSet=powerset -le '@l = (1,2,3,4,5); for (@{powerset(@l)}) { print "@$\_" }'

# Convert an IP address to unsigned integer

perl -le '$i=3; $u += ($\_<<8\*$i--) for "127.0.0.1" =~ /(\d+)/g; print $u'

perl -le '$ip="127.0.0.1"; $ip =~ s/(\d+)\.?/sprintf("%02x", $1)/ge; print hex($ip)'

perl -le 'print unpack("N", 127.0.0.1)'

perl -MSocket -le 'print unpack("N", inet\_aton("127.0.0.1"))'

# Convert an unsigned integer to an IP address

perl -MSocket -le 'print inet\_ntoa(pack("N", 2130706433))'

perl -le '$ip = 2130706433; print join ".", map { (($ip>>8\*($\_))&0xFF) } reverse 0..3'

perl -le '$ip = 2130706433; $, = "."; print map { (($ip>>8\*($\_))&0xFF) } reverse 0..3'

STRING CREATION AND ARRAY CREATION

----------------------------------

# Generate and print the alphabet

perl -le 'print a..z'

perl -le 'print ("a".."z")'

perl -le '$, = ","; print ("a".."z")'

perl -le 'print join ",", ("a".."z")'

# Generate and print all the strings from "a" to "zz"

perl -le 'print ("a".."zz")'

perl -le 'print "aa".."zz"'

# Create a hex lookup table

@hex = (0..9, "a".."f")

# Convert a decimal number to hex using @hex lookup table

perl -le '$num = 255; @hex = (0..9, "a".."f"); while ($num) { $s = $hex[($num%16)&15].$s; $num = int $num/16 } print $s'

perl -le '$hex = sprintf("%x", 255); print $hex'

perl -le '$num = "ff"; print hex $num'

# Generate a random 8 character password

perl -le 'print map { ("a".."z")[rand 26] } 1..8'

perl -le 'print map { ("a".."z", 0..9)[rand 36] } 1..8'

# Create a string of specific length

perl -le 'print "a"x50'

# Create a repeated list of elements

perl -le '@list = (1,2)x20; print "@list"'

# Create an array from a string

@months = split ' ', "Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec"

@months = qw/Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec/

# Create a string from an array

@stuff = ("hello", 0..9, "world"); $string = join '-', @stuff

# Find the numeric values for characters in the string

perl -le 'print join ", ", map { ord } split //, "hello world"'

# Convert a list of numeric ASCII values into a string

perl -le '@ascii = (99, 111, 100, 105, 110, 103); print pack("C\*", @ascii)'

perl -le '@ascii = (99, 111, 100, 105, 110, 103); print map { chr } @ascii'

# Generate an array with odd numbers from 1 to 100

perl -le '@odd = grep {$\_ % 2 == 1} 1..100; print "@odd"'

perl -le '@odd = grep { $\_ & 1 } 1..100; print "@odd"'

# Generate an array with even numbers from 1 to 100

perl -le '@even = grep {$\_ % 2 == 0} 1..100; print "@even"'

# Find the length of the string

perl -le 'print length "one-liners are great"'

# Find the number of elements in an array

perl -le '@array = ("a".."z"); print scalar @array'

perl -le '@array = ("a".."z"); print $#array + 1'

TEXT CONVERSION AND SUBSTITUTION

--------------------------------

# ROT13 a string

'y/A-Za-z/N-ZA-Mn-za-m/'

# ROT 13 a file

perl -lpe 'y/A-Za-z/N-ZA-Mn-za-m/' file

# Base64 encode a string

perl -MMIME::Base64 -e 'print encode\_base64("string")'

perl -MMIME::Base64 -0777 -ne 'print encode\_base64($\_)' file

# Base64 decode a string

perl -MMIME::Base64 -le 'print decode\_base64("base64string")'

perl -MMIME::Base64 -ne 'print decode\_base64($\_)' file

# URL-escape a string

perl -MURI::Escape -le 'print uri\_escape($string)'

# URL-unescape a string

perl -MURI::Escape -le 'print uri\_unescape($string)'

# HTML-encode a string

perl -MHTML::Entities -le 'print encode\_entities($string)'

# HTML-decode a string

perl -MHTML::Entities -le 'print decode\_entities($string)'

# Convert all text to uppercase

perl -nle 'print uc'

perl -ple '$\_=uc'

perl -nle 'print "\U$\_"'

# Convert all text to lowercase

perl -nle 'print lc'

perl -ple '$\_=lc'

perl -nle 'print "\L$\_"'

# Uppercase only the first word of each line

perl -nle 'print ucfirst lc'

perl -nle 'print "\u\L$\_"'

# Invert the letter case

perl -ple 'y/A-Za-z/a-zA-Z/'

# Camel case each line

perl -ple 's/(\w+)/\u$1/g'

perl -ple 's/(?<!['])(\w+)/\u\1/g'

# Strip leading whitespace (spaces, tabs) from the beginning of each line

perl -ple 's/^[ \t]+//'

perl -ple 's/^\s+//'

# Strip trailing whitespace (space, tabs) from the end of each line

perl -ple 's/[ \t]+$//'

# Strip whitespace from the beginning and end of each line

perl -ple 's/^[ \t]+|[ \t]+$//g'

# Convert UNIX newlines to DOS/Windows newlines

perl -pe 's|\n|\r\n|'

# Convert DOS/Windows newlines to UNIX newlines

perl -pe 's|\r\n|\n|'

# Convert UNIX newlines to Mac newlines

perl -pe 's|\n|\r|'

# Substitute (find and replace) "foo" with "bar" on each line

perl -pe 's/foo/bar/'

# Substitute (find and replace) all "foo"s with "bar" on each line

perl -pe 's/foo/bar/g'

# Substitute (find and replace) "foo" with "bar" on lines that match "baz"

perl -pe '/baz/ && s/foo/bar/'

# Binary patch a file (find and replace a given array of bytes as hex numbers)

perl -pi -e 's/\x89\xD8\x48\x8B/\x90\x90\x48\x8B/g' file

SELECTIVE PRINTING AND DELETING OF CERTAIN LINES

------------------------------------------------

# Print the first line of a file (emulate head -1)

perl -ne 'print; exit'

# Print the first 10 lines of a file (emulate head -10)

perl -ne 'print if $. <= 10'

perl -ne '$. <= 10 && print'

perl -ne 'print if 1..10'

# Print the last line of a file (emulate tail -1)

perl -ne '$last = $\_; END { print $last }'

perl -ne 'print if eof'

# Print the last 10 lines of a file (emulate tail -10)

perl -ne 'push @a, $\_; @a = @a[@a-10..$#a]; END { print @a }'

# Print only lines that match a regular expression

perl -ne '/regex/ && print'

# Print only lines that do not match a regular expression

perl -ne '!/regex/ && print'

# Print the line before a line that matches a regular expression

perl -ne '/regex/ && $last && print $last; $last = $\_'

# Print the line after a line that matches a regular expression

perl -ne 'if ($p) { print; $p = 0 } $p++ if /regex/'

# Print lines that match regex AAA and regex BBB in any order

perl -ne '/AAA/ && /BBB/ && print'

# Print lines that don't match match regexes AAA and BBB

perl -ne '!/AAA/ && !/BBB/ && print'

# Print lines that match regex AAA followed by regex BBB followed by CCC

perl -ne '/AAA.\*BBB.\*CCC/ && print'

# Print lines that are 80 chars or longer

perl -ne 'print if length >= 80'

# Print lines that are less than 80 chars in length

perl -ne 'print if length < 80'

# Print only line 13

perl -ne '$. == 13 && print && exit'

# Print all lines except line 27

perl -ne '$. != 27 && print'

perl -ne 'print if $. != 27'

# Print only lines 13, 19 and 67

perl -ne 'print if $. == 13 || $. == 19 || $. == 67'

perl -ne 'print if int($.) ~~ (13, 19, 67)'

# Print all lines between two regexes (including lines that match regex)

perl -ne 'print if /regex1/../regex2/'

# Print all lines from line 17 to line 30

perl -ne 'print if $. >= 17 && $. <= 30'

perl -ne 'print if int($.) ~~ (17..30)'

perl -ne 'print if grep { $\_ == $. } 17..30'

# Print the longest line

perl -ne '$l = $\_ if length($\_) > length($l); END { print $l }'

# Print the shortest line

perl -ne '$s = $\_ if $. == 1; $s = $\_ if length($\_) < length($s); END { print $s }'

# Print all lines that contain a number

perl -ne 'print if /\d/'

# Find all lines that contain only a number

perl -ne 'print if /^\d+$/'

# Print all lines that contain only characters

perl -ne 'print if /^[[:alpha:]]+$/

# Print every second line

perl -ne 'print if $. % 2'

# Print every second line, starting the second line

perl -ne 'print if $. % 2 == 0'

# Print all lines that repeat

perl -ne 'print if ++$a{$\_} == 2'

# Print all unique lines

perl -ne 'print unless $a{$\_}++'

# Print the first field (word) of every line (emulate cut -f 1 -d ' ')

perl -alne 'print $F[0]'

HANDY REGULAR EXPRESSIONS

-------------------------

# Match something that looks like an IP address

/^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$/

/^(\d{1,3}\.){3}\d{1,3}$/

# Test if a number is in range 0-255

/^([0-9]|[0-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])$/

# Match an IP address

my $ip\_part = qr|([0-9]|[0-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])|;

if ($ip =~ /^($ip\_part\.){3}$ip\_part$/) {

say "valid ip";

}

# Check if the string looks like an email address

/\S+@\S+\.\S+/

# Check if the string is a decimal number

/^\d+$/

/^[+-]?\d+$/

/^[+-]?\d+\.?\d\*$/

# Check if the string is a hexadecimal number

/^0x[0-9a-f]+$/i

# Check if the string is an octal number

/^0[0-7]+$/

# Check if the string is binary

/^[01]+$/

# Check if a word appears twice in the string

/(word).\*\1/

# Increase all numbers by one in the string

$str =~ s/(\d+)/$1+1/ge

# Extract HTTP User-Agent string from the HTTP headers

/^User-Agent: (.+)$/

# Match printable ASCII characters

/[ -~]/

# Match unprintable ASCII characters

/[^ -~]/

# Match text between two HTML tags

m|<strong>([^<]\*)</strong>|

m|<strong>(.\*?)</strong>|

# Replace all <b> tags with <strong>

$html =~ s|<(/)?b>|<$1strong>|g

# Extract all matches from a regular expression

my @matches = $text =~ /regex/g;

PERL TRICKS

-----------

# Print the version of a Perl module

perl -MModule -le 'print $Module::VERSION'

perl -MLWP::UserAgent -le 'print $LWP::UserAgent::VERSION'

PERL ONE-LINERS EXPLAINED E-BOOK

--------------------------------

I have written an ebook based on the one-liners in this file. If you wish to

support my work and learn more about these one-liners, you can get a copy

of my ebook at:

http://www.catonmat.net/blog/perl-book/

The ebook is based on the 7-part article series that I wrote on my blog.

In the ebook I reviewed all the one-liners, improved explanations, added

new ones, and added two new chapters - introduction to Perl one-liners

and summary of commonly used special variables.

You can read the original article series here:

http://www.catonmat.net/blog/perl-one-liners-explained-part-one/

http://www.catonmat.net/blog/perl-one-liners-explained-part-two/

http://www.catonmat.net/blog/perl-one-liners-explained-part-three/

http://www.catonmat.net/blog/perl-one-liners-explained-part-four/

http://www.catonmat.net/blog/perl-one-liners-explained-part-five/

http://www.catonmat.net/blog/perl-one-liners-explained-part-six/

http://www.catonmat.net/blog/perl-one-liners-explained-part-seven/

CREDITS

-------

Andy Lester http://www.petdance.com

Shlomi Fish http://www.shlomifish.org

Madars Virza http://www.madars.org

caffecaldo https://github.com/caffecaldo

Kirk Kimmel https://github.com/kimmel

avar https://github.com/avar

rent0n

FOUND A BUG? HAVE ANOTHER ONE-LINER?

------------------------------------

Email bugs and new one-liners to me at peter@catonmat.net!

HAVE FUN

--------

I hope you found these one-liners useful. Have fun!

#---end of file---