Cheatography

Perl Reference Card Cheat Sheet

by Nikolay Mishin (mishin) via cheatography.com/1008/cs/399/

About

This is version 2 of the perl reference card.
(cl) 2008 Michael Goerz <goerz@physik.fu-berlin.de>.
http://www.physik.fu-berlin.de/~goerz/
Information taken liberally from the perl documentation and various other sources.
You may freely distribute this document.

1 Variable Types

1.1 Scalars and Strings

chomp($str);	discard trailing \n
$v = chop($str);	$v becomes trailing char
eq, ne, lt, gt, le, ge, cmp	string comparison
$str = "0" x 4;	$str is now "0000"
$v = index($str, $x);	find index of $x in $str,
$v = rindex($str, $x);	starting from left or right
$v = substr($str, $strt, $len);	extract substring
$cnt = $sky =~ tr/0-9//;	count the digits in $sky
$str =~ tr/a-zA-Z/ /cs;	change non-alphas to space
$v = sprintf("%10s %08d",$s,$n);	format string
Format String:	%[flags][0][width][.precision][mod]type
types:	
c	character
d(i)	signed decimal int

1.1 Scalars and Strings (cont)

e(E)	scientific notation
f	decimal floating point
g, G	shorter %e or %f / %E or %f
o	signed octal
s	string of chars
u, x, X	unsigned decimal int / hex int / hex int in caps
p	address pointer
n	nothing printed
modifiers: h,l,L	arg is short int / long int, double/ long double
More:	
chr, crypt, hex, lc, lcfirst, length, oct, ord, pack	q/STRING/, qq/STRING/, reverse, uc, ucfirst

1.2 Arrays and Lists

@a = (1..5);	array initialization
$i = @a;	number of elements in @a
($a, $b) = ($b, $a);	swap $a and $b
$x = $a[1];	access to index 1
$i = $#a;	last index in @a
push(@a, $s);	appends $s to @a
$a = pop(@a);	removes last element
chop(@a);	remove last char (per el.)
$a = shift(@a);	removes first element

1.2 Arrays and Lists (cont)

reverse(@a);	reverse @a
@a = sort{$ela <=> $elb}(@a);	sort numerically
@a = split(/-/,$s);	split string into @a
$s = join(", " @c);	join @a elements into string
@a2 = @a[1,2,6..9];	array slice
@a2 = grep(!/^#/, @a);	remove comments from @a

Perl image

1.3 Hashes

%h=(k1 => "val1",k2 => 3);	hash initialization
$val = $map{k1};	recall value
@a = %h;	array of keys and values
%h = @a;	create hash from array
foreach $k (keys(%h)){..}	iterate over list of keys
foreach $v (vals(%h)){..}	iterate over list of values
while (($k,$v)=each %h){..}	iterate over key-value-pairs
delete $h{k1};	delete key
exists $h{k1}	does key exist?
defined $h{k1}	is key defined?

3 References and Data Structures

$aref = \@a;	reference to array
$aref = [1,"foo",undef,13];	anonymous array
$el = $aref->[0]; $el = @{$aref}[0];	access element of array
$aref2 = [@{$aref1}];	copy array
$href = \%h;	reference to hash
$href ={APR => 4,AUG => 8};	anonymous hash
$el = $href->{APR}; $el = %{$href}{APR};	access element of hash
$href2 = {%{$href1}};	copy hash
if (ref($r) eq "HASH") {}	checks if $r points to hash
@a = ([1, 2],[3, 4]);	2-dim array
$i = $a[0][1];	access 2-dim array
%HoA=(fs=>["f","b"], sp=>["h","m"]);	hash of arrays
$name = $HoA{sp}[1];	access to hash of arrays
$fh = *STDIN	globref
$coderef = \&fnc;	code ref (e.g. callback)
$coderef =sub{print "bla"};	anon subroutine
&$coderef();	calling anon subroutine

3 References and Data Structures (cont)

sub createcnt{ my $c=shift; return sub { print "$c++"; }; }	closure, $c persists
*foo{THING}	foo-syntax for creating refs

Link to perl cheat

perlcheat

> http://www.cheatography.com/mishin/cheat-sheets/perlcheat/

perl-reference-card

> http://www.cheatography.com/mishin/cheat-sheets/perl-reference-card/

20-killer-perl-programming-tips

> http://www.cheatography.com/mishin/cheat-sheets/20-killer-perl-programming-tips-for-beginners/

2 Basic Syntax

($a, $b) = shift(@ARGV);	read command line params
sub p{my $var = shift; ...}	define subroutine
p("bla");	execute subroutine
if(expr){} elsif {} else {}	conditional
unless (expr){}	negative conditional
while (expr){}	while-loop
until (expr){}	until-loop
do {} until (expr)	postcheck until-loop
for($i=1; $i<=10; $i++){}	for-loop
foreach $i (@list){}	foreach-loop
last, next, redo	end loop, skip to next, jump to top

2 Basic Syntax (cont)

eval {$a=$a/$b; }; warn $@ if $@;	exception handling

6 Regular Expressions

($var =~ /re/), ($var !~ /re/)	matches / does not match
m/pattern/igmsoxc	matching pattern
qr/pattern/imsox	store regex in variable
s/pattern/replacement/igmsoxe	search and replace
Modifiers:	
i case-insensitive	o compile once
g global	x extended
s as single line (. matches \n)	e evaluate replacement
Syntax:	
\	escape
.	any single char
^	start of line
$	end of line
, ?	0 or more times (greedy / nongreedy)
+, +?	1 or more times (greedy / nongreedy)
?, ??	0 or 1 times (greedy / nongreedy)
\b, \B	word boundary (\w - \W) / match except at w.b.
\A	string start (with /m)
\Z	string end (before \n)
\z	absolute string end

Perl Reference Card Cheat Sheet
by Nikolay Mishin (mishin) via cheatography.com/1008/cs/399/

6 Regular Expressions (cont)

\G	continue from previous m//g
[...]	character set
(...)	group, capture to $1, $2
(?:...)	group without capturing
{n,m} , {n,m}?	at least n times, at most m times
{n,} , {n,}?	at least n times
{n} , {n}?	exactly n times
|	or
\1, \2	text from nth group ($1, ...)
Escape Sequences:	
\a alarm (beep)	\e escape
\f formfeed	\n newline
\r carriage return	\t tab
\cx control-x	\l lowercase next char
\L lowercase until \E	\U uppercase until \E
\Q diable metachars until \E	\E end case modifications
Character Classes:	
[amy]	'a', 'm', or 'y'
[f-j.-]	range f-j, dot, and dash
[^f-j]	everything except range f-j
\d, \D	digit [0-9] / non-digit

6 Regular Expressions (cont)

\w, \W	word char [a-zA-Z0-9_] / non-word char
\s, \S	whitepace [\t\n\r\f] / non-space
\C	match a byte
\pP, \PP	match p-named unicode / non-p-named-unicode
\p{...}, \P{...}	match long-named unicode / non-named-unicode
\X	match extended unicode
Posix:	
[:alnum:]	alphanumeric
[:alpha:]	alphabetic
[:ascii:]	any ASCII char
[:blank:]	whitespace [\t]
[:cntrl:]	control characters
[:digit:]	digits
[:graph:]	alphanum + punctuation
[:lower:]	lowercase chars
[:print:]	alphanum, punct, space
[:punct:]	punctuation
[:space:]	whitespace [\s\ck]
[:upper:]	uppercase chars
[:word:]	alphanum + '_'
[:xdigit:]	hex digit
[:^digit:]	non-digit
Extended Constructs	
(?#text)	comment
(?imxs-imsx:...)	enable or disable option
(?=...), (?!...)	positive / negative look-ahead

6 Regular Expressions (cont)

(?<=..), (?<!..)	positive / negative look-behind
(?>...)	prohibit backtracking
(?{ code })	embedded code
(??{ code })	dynamic regex
(?(cond)yes|no)	condition corresponding to captured parentheses
(?(cond)yes)	condition corresponding to look-around
Variables	
$&	entire matched string
$`	everything prior to matched string
$'	everything after matched string
$1, $2 ...	n-th captured expression
$+	last parenthesis pattern match
$^N	most recently closed capt.
$^R	result of last (?{...})
@-, @+	offsets of starts / ends of groups

http://perldoc.perl.org/perlrequick.html

http://habrahabr.ru/post/17126/

Perl Reference Card Cheat Sheet

by Nikolay Mishin (mishin) via cheatography.com/1008/cs/399/

Debugging regexp

```
use re 'taint';
# Contents of $match are tainted if $dirty was also tainted.
($match) = ($dirty =~ /^(.*)$/s);
# Allow code interpolation:
use re 'eval';
$pat = '(?{ $var = 1 })'; # embedded code execution
/alpha${pat}omega/; # won't fail unless under -T
# and $pat is tainted
use re 'debug'; # like "perl -Dr"
/^(.*)$/s; # output debugging info during
# compile time and run time
use re 'debugcolor'; # same as 'debug',
# but with colored output
```

4 System Interaction

system("cat $f\|sort -u>$f.s");	system call
@a = readpipe("lsmod");	catch output
$today = "Today: ".`date`;	catch output
better: use IPC::Open3 'open3';!	
chroot("/home/user/");	change root
while (<*.c>) {}	operate on all c-files
unlink("/tmp/file");	delete file
if (-f "file.txt"){...}	file test

4 System Interaction (cont)

File Tests:	
-r, -w	readable, writeable
-x	executable
-e	exists
-f, -d, -l	is file, directory, symlink
-T, -B	text file, binary file
-M, -A	mod/access age in days
@stats = stat("filename");	13-element list with status
File Tests in Perl	http://www.devshed.com/c/a/Perl/File-Tests-in-Perl/
More:	
chmod, chown, chroot, fcntl, glob, ioctl, link, lstat, mkdir,	opendir, readlink, rename, rmdir, symlink, umask, utime

5 Input/Output

open(INFILE,"in.txt") or die;	open file for input
open(INFILE,"<:utf8","file");	open file with encoding
open(TMP, "+>", undef);	open anonymous temp file
open(MEMORY,'>', \$var);	open in-memory-file
open(OUT,">out.txt") or die;	open output file
open(LOG,">>my.log") or die;	open file for append

5 Input/Output (cont)

open(PRC,"caesar <$file |");	read from process
open(EXTRACT, "|sort >Tmp$$");	write to process
$line = <INFILE>;	get next line
@lines = <INFILE>;	slurp infile
foreach $line (<STDIN>){...}	loop of lines from STDIN
print STDERR "Warning 1.\n";	print to STDERR
close INFILE;	close filehandle
More:	
binmode, dbmopen, dbmclose, fileno, flock, format, getc, read, readdir, readline, rewinddir, seek, seekdir	select, syscall, sysread, sysseek, tell, telldir,truncate, pack, unpack, vec

7 Object-Oriented Perl and Modules

Defining a new class:

```
package Person;
use strict;
my $Census;
sub new { #constructor, any name is fine
my $class = shift;
my $self = {};
$self->{NAME} = undef; # field
$self->{"_CENSUS"} = \$Census; # class data
++ ${ $self->{"_CENSUS"} };
bless ($self, $class);
return $self;
}
sub name { #method
my $self = shift;
```

Perl Reference Card Cheat Sheet

by Nikolay Mishin (mishin) via cheatography.com/1008/cs/399/

7 Object-Oriented Perl and Modules (cont)

```
if (@_) { $self->{NAME} = shift }
return $self->{NAME};
}
sub DESTROY { #destructor
my $self = shift; -- ${$self->{"_CENSUS"} };}
1; # so the 'require' or 'use' succeeds
```

Using the class:

```
use Person;
$him = Person->new();
$him->name("Jason");
printf "There's someone named %s.\n", $him->name;
use Data::Dumper; print Dumper($him); # debug
```

http://www.codeproject.com/Articles/3152/Perl-Object-Oriented-Programming

http://ynonperek.com/course/perl/oo.html

Installing Modules:

perl -MCPAN -e shell;

8 One-Liners

-0	(zero) specify the input record separator
-a	split data into an array named @F
-F	specify pattern for -a to use when splitting
-i	edit files in place
-n	run through all the @ARGV arguments as files, using <>

8 One-Liners (cont)

-p	same as -n, but will also print the contents of $_
Interactive Mode:	perl -de1;use Term::ReadKey;
	http://szabgab.com/using-the-built-in-debugger-of-perl-as-repl.html
perl-debugger	http://www.thegeekstuff.com/2010/05/perl-debugger/
The Perl Debugger	http://docstore.mik.ua/orelly/perl/prog3/ch20_01.htm
-T	enables taint checking, which instructs perl to keep track of data from the user and avoid doing anything insecure with it. Here this option is used to avoid taking the current directory name from the @INC variable and listing the available .pm files from the directory recursively.

8 One-Liners (cont)

-l	enables automatic line-ending processing in the output. Print statements will have the new line separator (\n) added at the end of each line.
-w	prints any warning messages.
-e	indicates that the following string is to be interpreted as a perl script (i.e., sequence of commands).

http://perldoc.perl.org/perlrun.html

Perl flags -pe, -pi, -p, -w, -d, -i, -t? perldoc perlrun	perl -e '$x = "Hello world!n"; print $x;'
	perl -MO=Deparse -p -e 1
	perl -MO=Deparse -p -i -e 1
	perl -MO=Deparse -p -i.bak -e 1

https://twitter.com/#!/perloneliner

Examples:

1. just lines 15 to 17, efficiently

> perl -ne 'print if $. >= 15; exit if $. >= 17;'

2. just lines NOT between line 10 and 20

> perl -ne 'print unless 10 .. 20'

Examples: (cont)

3. lines between START and END

```
perl -ne 'print if /START$/ .. /END$/'
```

4. in-place edit of *.c files changing all foo to bar

```
perl -pi.bak -e 's/\bfoo\b/bar/g' *.c
```

5. delete first 10 lines

```
perl -i.old -ne 'print unless 1 .. 10' foo.txt
```

6. change all the isolated oldvar occurrences to newvar

```
perl -i.old -pe 's{\boldvar\b}{newvar}g' *.[chy]
```

7. printing each line in reverse order

```
perl -e 'print reverse <>' file1 file2 file3 ....
```

8. find palindromes in the /usr/dict/words dictionary file

```
perl -lne '$_ = lc $_; print if $_ eq reverse'
/usr/dict/words
```

9. command-line that reverses all the bytes in a file

```
perl -0777e 'print scalar reverse <>' f1 f2 f3
```

10. word wrap between 50 and 72 chars

```
perl -p000e 'tr/ \t\n\r/ /; s/(.
{50,72})\s/$1\n/g;$_.="\n"x2'
```

11. strip and remove double spaces

```
perl -pe '$_ = " $_ "; tr/ \t/ /s; $_ =
substr($_,1,-1)'
```

12. move '*.txt.out' to '*.out'

```
perl -e '($n = $_) =~ s/\.txt(\.out)$/$1/ and
not -e $n and rename $_, $n for @ARGV' *
```

13. write a hash slice, which we have come as a reference to a hash

```
perl -E'my $h={1..8}; say for @{$h}{1,3,5,7}'
```

Examples: (cont)

14. If you had installed any modules from CPAN, then you will need to re-install all of them. (Naveed Massjouni)

```
perl -E 'say for grep /site_perl/,@INC'| xargs
find | perl -Fsite_perl/ -lane 'print $F[1] if
/\.pm$/' | cpanm --reinstall
```

15. Give executable rights to all perl file in dir

```
find /home/client0/public_html -type f -name
'*.pl' -print0 | xargs -0 chmod 0755
```

16. Find files matching name-pattern
https://gist.github.com/563679

```
perl -MFile::Find -le 'find(sub{print
$File::Find::name if /\b[a-z]{2}_[A-Z]
{2}/},"/usr")'
```

By **Nikolay Mishin** (mishin)
cheatography.com/mishin/
mishin.narod.ru

Published 2nd June, 2012.
Last updated 5th June, 2014.
Page 6 of 6.