

Perl Reference Card

This is version 2 of the perl reference card.
(c) 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

c	character
d(i)	signed decimal int
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.)
$a = shift(@a);             removes first element
reverse(@a);                reverse @a
@a = sort{$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
```

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?
```

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 $i (@list){}     foreach-loop
last, next, redo          end loop, skip to next, jump to top
eval {$a=$a/$b; };       exception handling
warn $@ if $@;
```

3 References and Data Structures

```
$aref = \@a;               reference to array
$aref = [1,"foo",undef,13]; anonymous array
$el = $aref->[0];          access element of array
$el = @{$aref}[0];
$aref2 = [@{$aref1}];     copy array
$href = \%h;              reference to hash
$href={APR => 4,AUG => 8}; anonymous hash
$el = $href->{APR};        access element of hash
$aref2 = {%{$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"],      hash of arrays
      sp=>["h","m"]);
$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
sub createcnt{ my $c=shift; closure, $c persists
  return sub {
    print "$c++"; }; }
*foo{THING}               foo-syntax for creating refs
```

4 System Interaction

```
system("cat $f|sort -u>$f.s"); system call
@a = readpipe("lsmod");     catch output
$today = "Today: "`date`;  catch output
chroot("/home/user/");     change root
while (<*.c>){}             operate on all c-files
unlink("/tmp/file");       delete file
if (-f "file.txt"){...}   file test
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
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,'>', \ $var);   open in-memory-file
open(OUT,">out.txt") or die; open output file
open(LOG,">>my.log") or die; open file for append
open(PRC,"caesar <$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
```

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/igmsuxe search and replace
Modifiers:
i case-insensitive        o compile once
g global                  x extended
m multiline               c don't reset pos (with g)
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
\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	diabla 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
\w, \W	word char [a-zA-Z0-9_] / non-word char
\s, \S	whitespace [\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
(?<=...), (?<!...)	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

7 Object-Oriented Perl and Modules

Defining a new class:

```
package Person;
use strict;
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;
    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
```

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 <>
-p	same as -n, but will also print the contents of \$_

Interactive Mode: perl -de 42

Examples:

- just lines 15 to 17, efficiently
perl -ne 'print if \$. >= 15; exit if \$. >= 17;'
- just lines NOT between line 10 and 20
perl -ne 'print unless 10 .. 20'
- lines between START and END
perl -ne 'print if /^START\$/ .. /^END\$/'
- in-place edit of *.c files changing all foo to bar
perl -pi.bak -e 's/\bfoo\b/bar/g' *.c
- delete first 10 lines
perl -i.old -ne 'print unless 1 .. 10' foo.txt
- change all the isolated oldvar occurrences to newvar
perl -i.old -pe 's{\boldvar\b}{newvar}g' *.ch[xy]
- printing each line in reverse order
perl -e 'print reverse <>' file1 file2 file3
- find palindromes in the /usr/dict/words dictionary file
perl -lne '\$_ = lc \$_; print if \$_ eq reverse' /usr/dict/words
- command-line that reverses all the bytes in a file
perl -0777e 'print scalar reverse <>' f1 f2 f3
- word wrap between 50 and 72 chars
perl -p000e 'tr/ \t\n\r/ /; s/(.{50,72})\s/\$1\n/g; \$_.="x2"
- strip and remove double spaces
perl -pe '\$_ = " \$_ "; tr/ \t/ /s; \$_ = substr(\$_,1,-1)'
- move *.txt.out to *.out
perl -e '(\$n = \$_) =~ s/\.txt(\.out)\$/\$1/ and not -e \$n and rename \$_, \$n for @ARGV' *