Perl Cheat Sheet: a handy reference

Getting Help:

```
perldoc perl
perldoc perlSOMETHING
perldoc perlfaq
perldoc perlfaqN        where N is 1..9
perldoc -f FUNC
```

Running Perl

-e *CMD*	single line of script
-w	warnings on
-c	checks syntax only
-n	non-printing input loop
-p	printing input loop
-i *[EXT]*	in-place edit, *EXT* of backup
-a	auto split (Splits to @F)
-F *REGEXP*	auto split delimiter
-0 *[OCT|HEX]*	set input record sep '$/'
-00	paragraph record mode
-M *MOD*	load a module
-I *[DIR]*	perpend *DIR* to *@INC*.
-S	use PATH for exec
-l *[OCT]*	auto-chomp and set '$\'
-CSD	enable UTF-8
-T	taint checking (***perldoc* perlsec**)
-d	enable debugger. "perl -de 0"
-D *NUM*	sets debugging flags
-u	dump core, see *undump(1)*
-U	unsafe operations mode
-v	ver and patchlevel of script
-x *[DIR]*	seek script after "#!" in input
-s -*xyz=abc*	cause $xyz='abc'

EXAMPLE: perl -s -e 'print "$myvar\n"' -- -myvar="myval"

***PerlDoc:* perlrun**

Contexts:	void	scaler	list		
SIGILS:	$scalar	@array	%hash	&sub	*glob
Scalers:	number	string	reference	glob	undef

	Arrays	Hashes
Whole:	@array	%hash
Slice:	@array[0, 2]	@hash{'a', 'b'}
Element:	$array[0]	$hash{'a'}

```
$#array                 last index
@a=1..3 ⇔ @a=(1,2,3)          @b=A..C ⇔ @b=("A","B","C")
@a=keys%hash                  @a=values%hash
$$foo[1]        ⇔   $foo->[1]
$$foo{bar}      ⇔   $foo->{bar}
${$$foo[1]}[2]  ⇔   $foo->[1]->[2] ⇔   $foo->[1][2]
```

References

\	**reference**
$@%&*	**dereference**
[ ]	**anon. arrayref**
{ }	**anon. hashref**
\( )	**list of refs**

***PerlDoc*: perlreftut, perlref, perldsc, perllol,perldata**

Special Varialbles

$_	default variable
$/	input record sep. (\n)
$\	output record sep.
$,	output field sep.
$"	sep. for "@ARRAY"
$:	(eg. \s and hyphens)
$;	multiD. array emu(\034)
$]	perl version str.
$0	script filename
$$	PID
$<	real UID
$>	effective UID
$(	real GID
$)	effective GID
$&	string being matched
$`	the preceding str
$'	the following str
$ARGV	filename of <>
@ARGV	command line args
%ENV	environment
@_	args in subs
@INC	include paths
$~	format (perldoc perlform)
$^	top-of-page format name
$^A	formline accumulator
$^L	output formfeed (\f)
$^T	script run time (epoch)
$^X	interpreter filename
$!	errno or errstr
$@	eval error msg
$?	return value child proc
$.	line number (last read)
$%	page # of output channel
$=	lines in output channel page (default 60)
$-	lines remaining on page
$|	autoflush (write)
$ARGV	filename of <>
$0	script filename
$+	last sub-pattern
$1 .. $9 .. ${100}..	sub-patterns
$^I	in-place edit

NOTE: "undef $^I" to turn off

***PerlDoc:* perlvar**

Operator Precedence

```
->
++ --
**
! ~ \ u+ u-
=~ !~
* / % x
+ - .
<< >>
named uops
< > <= >= lt gt le ge
== != <=> eq ne cmp
&
| ^
&&
||
.. ...
?:
= += -= *= etc.
, =>
list ops
not
and
or xor
```

***PerlDoc:* perlop**

SYNTAX

```
LABEL:              # a label for a loop/condition
for   (LIST) { }    # set $_ to current LIST member
for (a;b;c) { }
while ( ) { }                 until ( ) { }
if    ( ) { } elsif ( ) { } else { }
unless( ) { } elsif ( ) { } else { }
goto LABEL
```

NOTES:

for ⇔ foreach **ALWAYS**

conditions and loops could use **last**, **next**, **redo** followed by optional label

a condition or a loop could have a label, so use it

conditions could be used after a single statement like

```
print "Ok" if ($ok);
```

***PerlDoc:* perlsyn**

NUMBERS vs STRINGS

=	**=**
+	**.**
== !=	**eq ne**
< > <= >=	**lt gt le ge**
<=>	**cmp**

String literals:

Name	is week?	Method 1	Method 2
Literal		' '	q{ }
Literal	yes	" "	qq{ }
Command	yes	` `	qx{ }
Word list			qw{ }
Pattern match	yes	/ /	m{ }
compile pat	yes		qr{ }
Substitution	yes	s/ / /	s{ }{ }
Trans			tr{ }{ }
here-doc	yes	<<EOS	

NOTE: you can replace {} with any non-alphanum delimiter like = or +

NOTE: week works if delimiter is not ' ', interpolation is done with week,

NOTE: $n="123"; ⇔ $n=123; and $a="ok" ⇔ $a=v111.107

NOTE: binary operators with strings, eg. print "wxyz" ^ " " ,"\n";

PerlDoc: perlop

Escape Specials:

\t	TAB
\n	newline
\r	return
\f	form feed
\b	backspace
\a	alert (beep)
\e	escape
\l	lower next
\u	upper next
\L	lower till \E
\W	upper till \E
\Q	quote till \E
\E	end \L \W or \Q

\x*HEX* \0*OCT* \x{} \N{}

EXAMPLE: $a="err"; print"\U$a\E\n";

Numerical Literals

decimal	1234
binary	0b1100101
octal	01234
hexadecimal	0xDEADbeef
exponential	2.998e+8
legibility	299_792_458

NOTE: underscore _ is ignored

***PerlDoc:* perlnumber**

Regex Modifiers:

/i	case insens.
/m	line based ^$
/s	case . to include \n
/x	ign. wh.space
/g	global
/o	evaluate vars only once
/e	evaluate replacement

Regex Charclasses

.	⇔	[^\n]	anything
\s	⇔	[\x20\f\t\r\n]	white spaces
\w	⇔	[A-Za-z0-9_]	word (alphanum_)
\d	⇔	[0-9]	digit
\S \W \D	negate	\s \w \d	non-...etc.

***PerlDoc:*perlrequick, perlretut,perlre, perlreref**

Regex Metachars:

^	string begin
$	str. end(before \n)
+	one or more
*	zero or more
?	zero or one
{*3*,*7*}	repeated in range 3-7 times
[ ]	character class, set memeber
|	alternation
\b	word boundary
\z	string end
()	capture
(?:)	no capture
(?=)	+ve ahead assertion
(?!)	-ve ahead assertion
(?<=)	+ve back assertion
(?M)	Emb.REGEX MODIFIERS

UTF8 I/O:

```
#! /usr/bin/perl
use utf8;
use encoding 'utf8';
use open ':utf8'; # input and output default layer will be UTF-8
#...
```

***PerlDoc:*perlunicode**

Some Important Modules:

CGI, DBI (databases like MySQL), Digest::MD5this, Encode, File::Basename, Getopt::Long, Gtk, Locale::gettext, Math::BigFloat, Math::BigInt, Math::BigInt::Calc, Math::BigInt::CalcEmu, Math::BigRat, Math::Complex, Math::Trig, Net:Ping, Net::SMTP, O (compiler backend), ops, overload, POSIX, Safe, SDL, Tk, Unicode::Normalize, XML:Parser

Perl One-Liner: *Linux Gazette*

```
perl -wlne '/title>([^<]+)/i&&rename$ARGV,"$1.html"' *
```

to raname html files according to their titles

```
perl -ne 'END{print "@a\n"}push @a,/(\w+\@\w+\.\w+)/g'
```

silly spambot, to get emails from STDIN (or followed files)

Recommendations:

DO:

```
use strict; use warnings;
use Modules;      # Do not reinvent the wheel
my $var;          # declare locale variable scope
open(..) or die $!; # expect errors
```

DO NOT:

```
"$foo"
$$variable_name
`$userinput`      # execute arbitrary string from user/client
/$userinput/      # compile arbitrary RE from user/client
die "Can't open: $!" unless open(FOO,$foo); # not readable
```

***PerlDoc:* perlstyle**

Web Sites:

perl.com	**perlmonks.org**	**perl.apache.org**	**perl.plover.com**
tpj.com	**pm.org**	**cpan.org**	**search.cpan.org**
perldoc.com	**use.perl.org**	**parrotcode.org**	

JAPH: *Just Another Perl Hacker*

```
perl -e 'printf "U R %x man.\n", 57005;'
perl -e '$s=sprintf "%x\n", 3735928559; $s=~s/db/d b/; print $s'
```

By Muayyad Saleh Al-Sadi <alsadi@gmail.com>

Built-in Functions *(underlined defaults to $_)*

Conversion

```
abs X            absolute value
int X            str to integer
hex X            hex to dec
oct X            oct to dec
chr X            num(ASCII) to str
ord X            first char to num(ASCII)
```

Arithmetics

```
sin X
cos X
atan2 Y,X        arctan of Y/X
sqrt X           square root of X
exp X            e to the power X
log X            log X base e
```

Binary Manipulations

```
vec X, O, B      get B bits at O offset of X
pack T, LS       list to binary struct
unpack T, X      struct to list
```

Time and Random Gen.

```
rand [X]         0≤rand(X)<X
srand [X]        seed (not needed)
time             seconds since epoch
gmtime X         time to array
localtime X      time to array
```

EXAMPLE: `($sec, $min, $hour,$mday, $mon, $year,$wday,$yday,$isdst) = localtime(time);`

String Manipulations

```
chomp L          del ending "\n" from each L
chop L           del each last char from L
length E         num characters in E
lc E             return lowercase E
lcfirst E        same with 1st char only
uc E             return uppercased E
ucfirst E        same with 1st char only
quotemeta E      return E with special RE quoted
crypt S,SALT     Encrypts a string
eval E           run perl expression
index S,SUB[,O]  find 1st occurrence of SUBSTR in S
rindex S,SUB[,O] same but last
substr E,O[,L]   get L chars after O from E
```

Search and Replace Functions

```
pos S            \G position m//g (May be assigned to.)
study[$V]        study $V to speed next matching on it
[E=~][m]/RE/[g][i][m][o][s][x]
```

Search E for a ***RE***. first m is to use different pair of delimiters.
return array of subexpressions parentheses

```
[$V=~]s/RE/STR/[e][g][i][m][o][s][x]
```

Replace all non-overlapping matches of RE with STR
return number of replacements.

RE Modifiers:

```
g                global match (more than once)
i                case-insensitive
s                single line string
m                multi-line string (^ $ differ)
o                opt. interpolates variables once
x                enable extensions
e                replacement is to be evaluated as exp
```

```
[$VAR=~]y/SET1/SET2/[c][d][s]
[$VAR=~]tr/SET1/SET2/[c][d][s]
```

Translates SET1 characters to corresponding SET2 characters, return number of characters replaced.

Modifiers: c complements SET1; s squeezes repetitions d del. chars having no corresponding chars in SET2;

Array and List Functions

```
delete $H{K}     Delete value from hash.
each %H          Return next (key,val) array or null
exists $H{K}     Check if hash key exists.
keys %H          Return an array of hash keys
values %H        Return an array of hash values
scalar %H        true if H has elements defined
scalar @A        Return number of A elements
reverse L        invert order of L
unshift @A, L
```

Prepends L to A, and returns the number of elements in the new array.

```
shift [@A]       return first element and remove it
pop @A           Pop off and returns last value of A
push @A, L       Push values of L onto the end of A
join E, L        return strings of L joined by E separator
map E, L  or  map {BLOCK} L
```

Evaluate E or BLOCK for each element of L, locally setting $_ to it. Modifying $_ will modify the corresponding element from L. Returns the list of results.

```
grep E,L  or  grep {BLOCK} L
```

set $_ for each L element, allow modification,returns array where E true.

```
splice @A, O [ , N [ , L ] ]
```

Removes the N elements of @A designated by offset O, replacing them with L, Returns the elements removed.

```
sort [FUNC] L
```

return sorted version of L, FUNC receives $a $b

EXAMPLE: `@L = sort {$a cmp $b} @str_array;`

```
split [PAT [ , E [ , M ] ] ]
```

Splits a string E into an array of strings, and returns it.
at most M element are get, If no PAT splits at the whitespace.
If not in array context, returns number of fields and splits to @_.

File unary tests

```
-r -w -x         is file readable/writable/executable
-R -W -X         for real UID/GID not the effective one
-o -O            is file owned by effective/real uid.
-e -z            File exists/has zero size.
-s               return size (ie. exists and non-empty)
-f -d            is file regular file/a directory.
-l -S -p         is symbolic link/a socket/FIFO pipe
-b -c            is file a block/character device
-u -g -k         has setuid/setgid/sticky bit set.
-t               if filehandle (default STDIN) a tty.
-M -A -C    modification/access/inode-change time(days)
-T -B            is text/non-text (binary) file.
```

NOTE: both -T and -B return true on a null file (or a file at EOF when testing a filehandle)

File Operations

```
chmod MODE,L     set permissions of a list of files
chown U,G,L      owner and group of a list of files
truncate F, S    set size of F to S. filename or handle
link ORIG, LINK  make a hard link
symlink ORG,LINK make a soft link
mkdir DIR, MODE  Creates a directory
readlink E       Returns the value of a symbolic link
rename OLD, NEW  Changes the name of a file, or move it
rmdir FILENAME   Deletes the directory if it is empty
unlink L         Deletes a list of filenames
utime A,M,L      set access and modification times
lstat F          like stat, but does not follow link
stat F           Returns a 13-element array
(0: $dev, 1: $ino, 2: $mode, 3: $nlink, 4: $uid,
5: $gid, 6: $rdev, 7: $size, 8: $atime, 9: $mtime,
10: $ctime, 11: $blksize, 12: $blocks)
```

Directory Listing

```
opendir DH,DIR   close it with closedir DH
readdir DH       Returns the next entry
```

NOTE: in array context return an array of entries

I/O Operations

FH is a file handle that is created with open (or predefined like STDOUT)

```
<>  reads from file passed with @ARGV (STDIN elsewhere)
<FH>             reads a line (or array of all lines)
```

NOTE: FH is recommended to be in upper case

NOTE: use **or die "could not ...\n$!"**

```
open FH, M, FN [,L]
                 there are many more see perldoc -f open
M: 'r', 'r+', 'w', 'w+', 'a', and 'a+'
M: '>','>>','<' and all with '+' (eg. 'a+' '+>>')
M: '-|', '|-'  create pipe, L arguments for FN command
                                        PerlDoc: perlipc
binmode FH       set to binary mode (useless on UNIX)
close FH         closes file or pipe FH
eof FH           true if the all reading is done
eof              EOF status for the last file read.
fcntl FH,N,$V    UNIX fcntl, see man fcntl(2)
fileno FH        Returns the file descriptor
flock FH, OP     file lock like UNIX flock(2), where
OP: 1 shared, 2 exclusive, 4 non-blocking, or 8 unlock
getc [FH]        read character(or empty string)
ioctl FH,N,$V    UNIX ioctl(2)
pipe FH_R, FH_W  create a pair of connected pipes
sprintf F,L      return string of formatting L as F
print [FH][L]    output to FH
printf[FH][L]    output to FH
syswrite FH,SCALER,L[,O]     output to FH
read FH,$V,L[,O]
sysread FH,$V,L[,O]
                 input $V from FH, L: length,O: offset
seek FH,P,W      set offset, W: 0 abs,1 cur, 2 EOF
$nfound=select $RV, $WV, $EV,TIMEOUT
 like UNIX select(2),all can be undef, set vectors with
 $RV=''; vec($RV,fileno(FH),1) = 1;
tell [FH]        current file position
```

Formatted Output

```
formline PIC,L   Formats L according to PIC, add to $^A
write [FH]       writes a formatted record to FH
                 to set the format use

format[FH] =
FORMLIST
.
```

PerlDoc: **perlform**

see $-, $^, $~, $^A, $^F, $- and $=

System interaction

```
alarm SEC        send SIGALRM after SEC seconds.
chdir [E]        Changes the working directory.
chroot DIR       for process and its children.
die [L]          Prints L to STDERR
warn [L]         like die, but doesn't exit
exec L           system command in L; does not return
exit [E]         Exits immediately
fork             UNIX fork(2), used to in IPC
getlogin         current login name
getpgrp [PID]    group for process PID (or current)
setpgrp PID, PGRP
getppid          parent PID
getpriority WHICH, WHO
setpriority WHICH, WHO, PRIORITY
                 see UNIX getpriority(2)
glob PAT         shell pattern matching, return array
kill SIG,L       Sends a signal to a list of PIDs
sleep [SEC]
syscall L
system L         like exec L with a fork, and wait
wait             waits for a child process to terminate
waitpid PID, FLAGS
                 like wait but for PID
times            Returns a 4-element array
 (0: $user, 1: $system, 2: $cuser, 3: $csystem)
umask [EXPR]     sets the umask for the process
getpwnam NAME    get user information by login name
getpwuid UID     same but by UID
 returns array (0:$name, 1:$passwd, 2:$uid, 3:$gid,
 4:$quota, 5:$comment, 6:$gcos, 7:$dir, 8:$shell)
getgrnam NAME    get group information by group name.
getgrgid GID     same but by group GID.
 returns array (0:$name, 1:$passwd, 2:$gid, 3:$members)
```

Network Information:

```
gethostbyaddr, gethostbyname, getnetbyaddr,
getnetbyname, getservbyname, getservbyport,
getprotobyname, getprotobynumber,..etc
```

EXAMPLE: list network services, protocols and ports

```
perl -e 'while (($s,$a,$p,$P)=getservent) {print
"$s\t$P:$p\n"}'
```

Networking and IPC

```
accept, bind, connect, getpeername, getsockname,
getsockopt, recv, send, setsockopt, shutdown, socket
socketpair, msgctl,msgget, msgsnd, msgrcv, semctl,
semget, semop, shmctl, shmget, shmread, shmwrite, ...
```


By Muayyad Saleh Al-Sadi <alsadi@gmail.com>
based on perlcheat
by Juerd Waalboer<juerd@cpan.org>
http://juerd.nl/site.plp/perlcheat
and Perl 5 Desktop Reference
by Johan Vromans

Suggested References: Perl 5 Pocket Reference