pcwdld.com

Perl Cheat Sheet Updated for 2022 - From Functions to CL Commands!

Hitesh J

16-20 Minuten

Perl stands for "Practical Extraction and Reporting Language" and is a high-level, general-purpose, interpreted, dynamic programming language developed by Larry Wall in 1987.

It was originally designed for text manipulation. But now, it is used for a wide range of tasks, including system administration, web development, network programming, GUI development, and more.

It supports HTML, XML, and other mark-up languages and supports third-party databases including Oracle, Postgres, MySQL, Sybase, etc. In addition, it is extensible, and there are 20,000+ third-party modules available from the Comprehensive Perl Archive Network.

A cheat sheet is a set of commands and notes that will be helpful for both beginners and professionals as a quick reference. This Perl cheat sheet will provide basic and advanced management and syntax of the Perl programming language.

Perl Functions

A Perl function is a group of statements used to perform a specific task. It allows users to divide their code into separate parts to reuse the code defined in the function.

Functions for Strings	
Function	**Explanation**
chop	Remove the last character of a string and returns the character chopped.
chomp	Removes any trailing string that corresponds to the current value of $/.
crypt	It is a one-way hash function.
chr	Returns the character represented by that NUMBER in the character set.
fc	Returns the case folded version of EXPR. This is the internal function implementing the \F escape in double-quoted strings.
hex	Interprets EXPR as a hex string and returns the corresponding numeric value.
index	It searches for one string within another, but without the wildcard-like behavior of a full regular-expression pattern match.
lc	Returns a lowercased version of EXPR.
lcfirst	Returns the value of EXPR with the first character lowercased.
length	Returns the length in characters of the value of EXPR.

oct	Interprets EXPR as an octal string and returns the corresponding value.
ord	Returns the numeric value of the first character of EXPR. If EXPR is an empty string, it returns 0.
pack	Takes a LIST of values and converts it into a string using the rules given by the TEMPLATE.
reverse	Returns a list value consisting of the elements of LIST in the opposite order.
rindex	Works just like index except that it returns the position of the last occurrence of SUBSTR in STR. If POSITION is specified, returns the last occurrence beginning at or before that position.
sprintf	Returns a string formatted by the usual printf conventions of the C library function sprintf.
substr	Extracts a substring out of EXPR and returns it.
uc	Returns an uppercased version of EXPR.
ucfirst	Returns the value of EXPR with the first character in uppercase.
Numeric Functions	
Function	**Explanation**

abs	Returns the absolute value of its argument.
atan2	Returns the arctangent of Y/X in the range -PI to PI.
cos	Returns the cosine of EXPR.
exp	Returns the (natural logarithm base) to the power of EXPR.
hex	Interprets EXPR as a hex string and returns the corresponding numeric value.
int	Returns the integer portion of EXPR.
log	Returns the natural logarithm (base e) of EXPR.
oct	Interprets EXPR as an octal string and returns the corresponding value.
rand	Returns a random fractional number greater than or equal to 0 and less than the value of EXPR.
sin	Returns the sine of EXPR (expressed in radians).
sqrt	Return the positive square root of EXPR.
srand	Sets and returns the random number seed for the rand operator.
Functions for real @ARRAYs	
Function	**Explanation**

shift	Shifts the first value of the array off and returns it.
each	When called on a hash in list context, returns a 2-element list consisting of the key and value for the next hash element.
keys	Called in list context, returns a list consisting of all the keys of the named hash.
pop	Pops and returns the last value of the array, shortening the array by one element.
push	Treats ARRAY as a stack by appending the values of LIST to the end of ARRAY.
splice	Removes the elements designated by OFFSET and LENGTH from an array, and replaces them with LIST elements, if any. In the list context, returns the elements removed from the array.
unshift	Does the opposite of a shift. Or the opposite of a push, depending on how you look at it.
values	Returns a list consisting of all the values of the named hash.
delete	Deletes the specified elements from that hash, so that exists on that element no longer returns true.

Functions for list data	
Function	**Explanation**
join	Joins the separate strings of LIST into a single string with fields separated by the value of EXPR, and returns that new string.
map	Evaluates the BLOCK or EXPR for each LIST element and composes a list of the results of each such evaluation.
reverse	Returns a list value consisting of the elements of LIST in the opposite order.
sort	Sorts the LIST and returns the sorted list value.
unpack	Takes a string and expands it out into a list of values.
pack	Takes a LIST of values and converts it into a string using the rules given by the TEMPLATE.
read	Read LENGTH characters of data into variable SCALAR from the specified FILEHANDLE.
syscall	Execute an arbitrary system call.
sysread	Read LENGTH bytes of data into variable SCALAR from the specified FILEHANDLE.
sysseek	Bypasses normal buffered IO, so mixing it with reads other than sysread.

syswrite	Attempts to write LENGTH bytes from SCALAR to the file associated with FILEHANDLE.

Input and output functions	
Function	**Explanation**
binmode	Arranges for FILEHANDLE to be read or written in “binary” or “text” mode on systems where the run-time libraries distinguish between binary and text files.
close	Closes the file or pipe associated with the filehandle.
closedir	Closes a directory opened by opendir and returns the success of that system call.
die	Raise an exception.
fileno	Returns the file descriptor for a filehandle or directory handle, or undefined if the filehandle is not open.
eof	End of file.
format	Declare a picture format for use by the write function.
getc	Returns the next character from the input file attached to FILEHANDLE.
print	Prints a string or a list of strings. Returns true if successful.

printf	Output redirect to a filehandle.
read	Attempts to read LENGTH characters of data into variable SCALAR from the specified FILEHANDLE.

Functions for filehandles, files, or directories	
Function	**Explanation**
chdir	Change current working directory.
chmod	Changes the permissions on a file/list of files.
chown	Changes the owner (and group) of a list of files.
chroot	Changes the root directory for the current process to dirname.
link	Creates a new filename linked to the old filename.
lstat	Stats a symbolic link instead of the file the symbolic link points to.
mkdir	Create a directory
opendir	Opens the directory EXPR, associating it with DIRHANDLE for processing, using the readdir function.
readlink	Returns the value of a symbolic link, if symbolic links are implemented.
rename	Changes the name of a file.
rmdir	Remove a directory

select	Returns the currently selected filehandle.
symlink	Creates a new filename symbolically linked to the old filename.
umask	Sets the umask for the process to EXPR and returns the previous value.
unlink	Deletes a list of files.
Functions for fetching user and group info	
Function	**Explanation**
getlogin	Return who is logged in at this TTY.
getpwent	Get the next passwd record.
getpwnam	Get passwd record given user login name.
getpwuid	Get passwd record delivered user ID.
getgrgid GID	Gets information by group ID.
getgrnam NAME	Gets information by name.
getgrent	Gets following group information.
gethostent	Gets next host information.
Functions for fetching network info	
Function	**Explanation**
getnetbyname	Get networks record given name.
getnetent	Get the next network's record.
getpeername	Find the other end of a socket

	connection.
getprotobyname	Get protocol record given name.
getprotobynumber	Get protocol record numeric protocol.
gethostbyaddr	Gets information by IP address.
getprotoent	Get the next protocols record.
gethostbyname	Get host record given name.
accept	Accepts a new socket.
bind	Binds the NAME to the SOCKET.
connect	Connects the NAME to the SOCKET.
getsockname	Returns the name of the socket.
listen	Starts listening on the specified SOCKET.
send	Sends a message on the SOCKET.
shutdown	Shuts down a SOCKET.

Comparison Operators

Comparison operators are used for comparing operands and return a Boolean value based upon whether the comparison is true or not. Perl has two types of comparison operator sets, numeric scalar values and string scalar values.

Boolean Operators		
Operator	**Example**	
Logical AND operator	&& or and	($a && $b) is false

Logical OR operator	|| or or	($a or $b) is true
Logical NOT operator	! or not	not($a) is false or !($a) is false
Arithmetic Operators		
Numeric	**String**	**Description**
Less than	<	lt
Greater than	>	gt
Less than or equal	<=	le
Greater than or equal	>=	ge
Equality	==	eq
Inequality	!=	ne
Assoc Operators		
Operator		**Explanation**
left	**->**	Infix dereference operator
	=++	Auto-increment
	—	Auto-decrement
right	**	Exponentiation
right	\	Reference to an object (unary)
right	**! ˜**	Unary negation, bitwise complement
right	**+ –**	Unary plus, minus

left	**=˜**	Binds a scalar expression to a pattern match.
left	**!˜**	Same, but negates the result.
left	*** / % x**	Multiplication, division, modulo, repetition.
left	**+ – .**	Addition, subtraction, concatenation.
left	**>> <<**	Bitwise shift right, bitwise shift left.
	< > <= >=	Numerical relational operators.
	lt gt le ge	String relational operators.
	== != <=>	Numerical equal, not equal, compare.
	eq ne cmp	Stringwise equal, not equal, compare.
left	**&**	Bitwise AND
left	**| ^**	Bitwise OR, exclusive OR.
left	**&&**	Logical AND
left	**||**	Logical OR
right	**= += -= *=**	Assignment operators
left	**,**	Comma operator

Regular Expressions

Regular Expressions are an essential part of Perl Programming used for searching the specified text pattern. The following cheat sheet contains the different classes, Characters, and modifiers used in the regular expression.

Character Classes

Classes	**Explanation**
[abc.]	Includes only one of specified characters i.e. 'a', 'b', 'c', or '.'
[a-j]	Includes all the characters from a to j.
[a-z]	Includes all lowercase characters from a to z.
[^az]	Includes all characters except a and z.
\w	Includes all characters like [a-z, A-Z, 0-9]
\d	Matches for the digits like [0-9]
[ab][^cde]	Matches that the characters a and b should not be followed by c, d, and e.
\s	Matches for [\f\t\n\r] i.e form feed, tab, newline and carriage return.
\W	Complement of \w
\D	Complement of \d
\S	Complement of \s

Anchors

Anchors	**Explanation**
^	Matches at the beginning of the string.
$	Matches at the end of the string.
\b	Matches at the word boundary of the string from \w to \W.
\A	Matches at the beginning of the string.
\Z	Matches at the ending of the string or before the newline.

\z	Matches only at the end of the string.
\G	Matches at the specified position pos().
\p{….}	Unicode character class like IsLower, IsAlpha, etc.
\P{….}	Complement of Unicode character class.
Meta Characters	
Characters	**Explanation**
.	Any character except newline.
*****	Matches 0 or more times.
+	Matches 1 or more times.
?	Matches 0 or more times.
()	Used for grouping.
\	Use for quotes or special characters.
[]	Used for a set of characters.
{}	Used as repetition modifier.
Quantifiers	
Quantifiers	**Explanation**
a?	Checks if ‘a’ occurs 0 or 1 time.
a+	Checks if ‘a’ occurs 1 or more times.
a*	Checks if ‘a’ occurs 0 or more times.
a{2, 6}	Checks if ‘a’ occurs 2 to 6 times.
a{2, }	Checks if ‘a’ occurs 2 to infinite times.

a{2}	Checks if 'a' occurs 2 times.
Modifiers	
Modifiers	**Explanation**
\g	Used to replace all the occurrences of the string.
\gc	Allows continued search after \g match fails.
\s	Treats string as a single line.
i	Turns off the case sensitivity.
\x	Disregard all the white spaces.
(?#text)	Used to add comments in the code.
(?:pattern)	Used to match the pattern of the non-capturing group.
(?|pattern)	Used to match the pattern of the branch test.
(?=pattern)	Used for positive lookahead assertion.
(?!pattern)	Used for negative lookahead assertion.
(<=pattern)	Used for a positive look-behind assertion.
(<!pattern)	Used for a negative look-behind assertion.
\t	Used for inserting tab space.
\r	Carriage return character.
\n	Used for inserting a new line.
\h	Used for inserting horizontal white space.
\v	Used for inserting vertical white space.
\L	Used for lowercase characters.

\U	Used for upper case characters.

Variable and Special Variables

Variables are the reserved memory locations used to store and manipulate data throughout the program. When a variable is created, it occupies memory space. Perl provides three types of variables, scalars, lists, and hashes. They are used to manipulate the corresponding data types, including scalars, lists, and hashes.

Variables	
Variables	**Explanation**
$var	Default variable
$var[10]	11st element of array @var
$p = \@var	Now $p is a reference to @var
$$p[10]	11st element of array referenced by $p
$var[-1]	Last element of array @var
$var[$x][$y]	$y-th element of $x-th element of array @var
$var{'JAN'}	A value from 'hash' %var
$p = \%var	Now $p is a reference to hash %var
$$p{'JAN'}	A value from hash referenced by $p
$#var	Last index of array @var
@var	The entire array
%var	The entire hash
Special Variables	

Variable	Explanation
$	Default variable
$/	The input record separator, newline by default.
$	The output record separator for the print operator.
$(	The real GID (Group ID) of this process.
$)	The effective GID (Group ID) of this process.
$&	The last successful pattern match matches the string.
$<	The real user ID of this process.
$>	The effective user ID of this process.
$(	The actual group ID of this process.
$)	The influential group ID and groups of this process.
$~	The name of the current report format for the currently selected output channel.
$^	The name of the current top-of-page format for the currently selected output channel.
$^A	The current value of the write() accumulator for format() lines.
$^L	What formats output as a form feed. The default is \f.
$^T	The time at which the program began running, in seconds since the epoch (beginning of 1970).

$^X	The name used to execute the current copy of Perl.
$!	Each element of %! has a true value only if $! is set to that value – %ERRNO.
$@	The Perl error from the last eval operator, i.e. the last exception that was caught.
$?	The status returned by the last pipe close, backtick (“) command, successful call to wait() or waitpid(), or from the system() operator.
$.	The current line number for the last filehandle accessed.
$%	The current page number of the currently selected output channel.
$=	The current page length (printable lines) of the currently selected output channel. The default is 60.
$-	The number of lines left on the page of the currently selected output channel.
$|	If set to nonzero, forces a flush right away and after every write or print on the currently selected output channel.
$0	Contains the name of the program being executed.
$+	The text matched by the highest used capture group of the last successful search pattern.

Command-line options

Perl comes with a wide range of command-line options that can be used to turn on or turn off different behaviors. You can create one-off command-line scripts to make your programs more concise.

Command	**Explanation**
-a	Turns on autosplit mode.
-c	Checks syntax but does not execute.
-d	Runs the script under the debugger.
-h	Prints the Perl usage summary.
-m	Imports the MODULE before executing the script.
-n	Assumes an input loop around the script.
-P	Runs the C preprocessor on the script before compilation by Perl.
-S	Uses the PATH environment variable to search for the script.
-T	Turns on taint checking.
-u	Dumps core after compiling the script.
-U	Allows Perl to perform unsafe operations.
-v	Prints the version and patchlevel of your Perl executable.
-V	Prints Perl configuration information.
-w	Prints warnings about possible spelling errors.
-x	Extracts the Perl program from the input stream.

Related Post: Introduction to Supernetting