Perl Reference Guide

for Perl version 4.019

Perl program designed and created by
Larry Wall **<**lwall@netlabs.com**>**

Reference guide designed and created by
Johan Vromans **<**jv@mh.nl**>**

Contents

Rev. 4.019.1

Conventions

fixed	denotes literal text.
THIS	means variable text, i.e. things you must fill in.
THIS†	means that THIS will default to **$_** if omitted.
word	is a keyword, i.e. a word with a special meaning.
RET	denotes pressing a keyboard key.
[. . .]	denotes an optional part.
(. . .)*	means that the parentheses may be omitted.

1. Command line options

-a turns on autosplit mode when used with **-n** or **-p**. Splits to **@F**.

-c checks syntax but does not execute.

-d runs the script under the debugger. Use **-de 0** to start the debugger without a script.

-D NUMBER
: sets debugging flags.

-e COMMANDLINE
: may be used to enter one line of script. Multiple **-e** commands may be given to build up a multi-line script.

-i EXT
: files processed by the **<>** construct are to be edited in-place.

-I DIR
: with **-P**: tells the C preprocessor where to look for include files. The directory is prepended to **@INC**.

-L OCTNUM
: enables automatic line ending processing.

-n assumes an input loop around your script. Lines are not printed.

-p assumes an input loop around your script. Lines are printed.

-P runs the C preprocessor on the script before compilation by perl.

-s interprets "**-xxx**" on the command line as switches and sets the corresponding variables **$xxx** in the script.

-S uses the **PATH** environment variable to search for the script.

-u dumps core after compiling the script. To be used with the *undump* program (where available).

-U allows perl to do unsafe operations.

-v prints the version and patchlevel of your perl executable.

-w prints warnings about possible spelling errors and other error-prone constructs in the script.

-x extracts perl program from input stream.

-0 VAL
: (that's the number zero) designates an initial value for the record terminator **$/**. See also **-L**.

2. Literals

Numeric: **123 123.4 5E-10 0xff** (hex) **0377** (octal).

String: **'abc'** literal string, no variable interpolation nor escape characters.
Also: **q/abc/**.
(Almost any pair of delimiters can be used instead of **/.../**.)

"abc" Variables are interpolated and escape sequences are processed.
Also: **qq/abc/**.
Escape sequences: **\t** (Tab), **\n** (Newline), **\r** (Return), **\f** (Formfeed), **\b** (Backspace), **\a** (Alarm), **\e** (Escape), **\033**(octal), **\x1b**(hex), **\c[** (control).
\l and **\u** lowcase/upcase the following character;
\L and **\U** lowcase/upcase until a **\E** is encountered.

`COMMAND` evaluates to the output of the COMMAND.
Also: **qx/**COMMAND**/**.

Array: **(1,2,3)**. **()** is an empty array.
Also: **($a,$b,@rest) = (1,2,...);**
(1..4) is the same as **(1,2,3,4)**. Likewise **('abc'..'ade')**

Associative array: **(**KEY1**,**VAL1**,**KEY2**,**VAL2**,...)**

Filehandles:
Pre-defined: **<STDIN>**, **<STDOUT>**, **<STDERR>**, **<ARGV>**, **<DATA>**;
User-specified: **<**HANDLE**>**, **<$**VAR**>**.
<> is the input stream formed by the files specified in **@ARGV**, or standard input if no arguments are supplied.

Globs: **<**PATTERN**>** evaluates to all filenames according to the pattern.
Use **<${**VAR**}>** to glob from a variable.

Here-Is: **<<**IDENTIFIER
See the manual for details.

Special tokens:
__FILE__: filename; **__LINE__**: line number.
__END__: end of program; remaining lines can be read using **<**DATA**>**.

3. Variables

$var	a simple scalar variable
$var[28]	29th element of array **@var** (the **[]** are part of it)
$var{'Feb'}	one value from associative array **%var**
$#var	last index of array **@var**
@var	the entire array; in scalar context: the number of elements in the array
@var[3,4,5]	a slice of the array **@var**
@var{'a','b'}	a slice of **%var**; same as **($var{'a'},$var{'b'})**
%var	the entire associative array; in scalar context: TRUE if the array has elements
$var{'a',1,...}	emulates a multi-dimensional array
('a'..'z')[4,7,9]	a slice of an array literal
***NAME**	refers to all objects represented by NAME. "***name1 = *name2**" makes **name1** a reference to **name2**.

4. Statements

Every statement is an expression, optionally followed by a modifier, and terminated by a semi-colon.

Execution of expressions can depend on other expressions using one of the modifiers **if**, **unless**, **while** or **until**, e.g.:

EXPR1 **if** EXPR2 **;**
EXPR1 **until** EXPR2 **;**

Also, by using one of the logical operators **||**, **&&** or **? :**, e.g.:

EXPR1 **||** EXPR2 **;**
EXPR1 **?** EXPR2 **:** EXPR3 **;**

Statements can be combined to form a BLOCK when enclosed in **{}**.

Compound statements may be used to control flow:

if (EXPR**)** BLOCK [[**elsif (**EXPR**)** BLOCK ...] **else** BLOCK]
unless (EXPR**)** BLOCK [**else** BLOCK]
[LABEL**:**] **while (**EXPR**)** BLOCK [**continue** BLOCK]
[LABEL**:**] **until (**EXPR**)** BLOCK [**continue** BLOCK]
[LABEL**:**] **for (**EXPR**;** EXPR**;** EXPR**)** BLOCK
[LABEL**:**] **foreach** VAR† **(**ARRAY**)** BLOCK
[LABEL**:**] BLOCK [**continue** BLOCK]

Special forms are:

do BLOCK **while** EXPR **;**
do BLOCK **until** EXPR **;**

which are guaranteed to perform BLOCK once before testing EXPR.

5. Flow control

do BLOCK

Returns the value of the last command in the sequence of commands indicated by BLOCK. **next**, **last** and **redo** cannot be used here.

do SUBROUTINE**(LIST)**

Executes a SUBROUTINE declared by a **sub** declaration, and returns the value of the last expression evaluated in SUBROUTINE .
Preferred form is: **&**SUBROUTINE .

do FILENAME

Executes the contents of FILENAME as a perl script. Errors are returned in **$@**.
Preferred form is: **require** FILENAME .

goto LABEL

Continue execution at the specified label.

last [LABEL]

Immediately exits the loop in question. Skips continue block.

next [LABEL]

Starts the next iteration of the loop.

redo [LABEL]

Restarts the loop block without evaluating the conditional again.

return EXPR

Returns from a subroutine with the value specified.

6. Operators

+ – * /	Addition, subtraction, multiplication, division.
%	Modulo division.
| & ^	Bitwise or, bitwise and, bitwise exclusive or.
>> <<	Bitwise shift right, bitwise shift left.
******	Exponentiation.
.	Concatenation of two strings.
x	Returns a string or array consisting of the left operand (an array or a string) repeated the number of times specified by the right operand.

All of the above operators also have an assignment operator, e.g. "**.=**".

++ --	Auto-increment (magical on strings), auto-decrement.
? :	Alternation (if-then-else) operator.
|| &&	Logical or, logical and.
== !=	Numeric equality, inequality.
eq ne	String equality, inequality.
< >	Numeric less than, greater than.
lt gt	String less than, greater than.
<= >=	Numeric less (greater) than or equal to.
le ge	String less (greater) than or equal.
<=>	Numeric compare. Returns -1, 0 or 1.
cmp	String compare. Returns -1, 0 or 1.
=~ !~	Search pattern, substitution, or translation (negated).
..	Enumeration, also input line range operator.
,	Comma operator.

7. File test operators

These unary operators takes one argument, either a filename or a filehandle, and tests the associated file to see if something is true about it. If the argument is omitted, tests **$_** (except for **-t,** which tests **STDIN**). If the special argument **_** (underscore) is passed, uses the info of the preceding test.

-r -w -x -o	File is readable/writable/executable/owned by effective uid.
-R -W -X -O	File is readable/writable/executable/owned by real uid.
-e -z -s	File exists / has zero/non-zero size.
-f -d	File is a plain file, a directory.
-l -S -p	File is a symbolic link, a socket, a named pipe (FIFO).
-b -c	File is a block/character special file.
-u -g -k	File has setuid/setgid/sticky bit set.
-t	Tests if filehandle (**STDIN** by default) is opened to a tty.
-T -B	File is a text/non-text (binary) file. **-T** and **-B** return TRUE on a null file, or a file at EOF when testing a filehandle.
-M -A -C	File creation / access / inode change time. Measured in days since this program started. See also **$^T** in section "Special Variables".

A LIST is a (possibly parenthesised) list of expressions, variables or LISTs. An array variable or an array slice may always be used instead of a LIST.

8. Arithmetic functions

atan2(Y,X**)**
: Returns the arctangent of Y/X in the range -π to π.

cos(EXPR†**)***
: Returns the cosine of EXPR (expressed in radians).

exp(EXPR†**)***
: Returns **e** to the power of EXPR.

int(EXPR†**)***
: Returns the integer portion of EXPR.

log(EXPR†**)***
: Returns natural logarithm (base **e**) of EXPR.

rand[**(**EXPR**)***]
: Returns a random fractional number between 0 and the value of EXPR. If EXPR is omitted, returns a value between 0 and 1.

sin(EXPR†**)***
: Returns the sine of EXPR (expressed in radians).

sqrt(EXPR†**)***
: Return the square root of EXPR.

srand[**(**EXPR**)***]
: Sets the random number seed for the rand operator.

time Returns the number of seconds since January 1, 1970. Suitable for feeding to **gmtime** and **localtime**.

9. Conversion functions

gmtime(EXPR**)***
: Converts a time as returned by the **time** function to a 9-element array (**$**sec, **$**min, **$**hour, **$**mday, **$**mon, **$**year, **$**wday, **$**yday, **$**isdst) with the time analyzed for the Greenwich timezone. **$**mon has the range 0..11 and **$**wday has the range 0..6.

hex(EXPR†**)***
: Returns the decimal value of EXPR interpreted as an hex string.

localtime(EXPR**)***
: Converts a time as returned by the **time** function to a 9-element array with the time analyzed for the local timezone.

oct(EXPR†**)***
: Returns the decimal value of EXPR interpreted as an octal string. If EXPR starts off with **0x**, interprets it as a hex string instead.

ord(EXPR†**)***
: Returns the ascii value of the first character of EXPR.

vec(EXPR,OFFSET,BITS**)**
: Treats EXPR as a string of unsigned ints, and yields the bit at OFFSET. BITS must be between 1 and 32. May be used as an lvalue.