Perl Reference Guide

for Perl version 4.010

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

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

Contents

Rev. 4.010.2.1

$: The set of characters after which a string may be broken to fill continuation fields (starting with "^") in a format.

$0 The name of the file containing the perl script being executed. May be assigned to.

$$ The process number of the perl running this script. Altered (in the child process) by **fork**.

$< The real uid of this process.

$> The effective uid of this process.

$(The real gid of this process.

$) The effective gid of this process.

$^D The debug flags as passed to perl using **-D** .

$^F The highest system file descriptor, ordinarily 2.

$^I In-place edit extension as passed to perl using **-i** .

$^P Internal debugging flag.

$^T The time (as delivered by **time**) when the program started. This value is used by the file test operators "**-M**", "**-A**" and "**-C**".

$^W The value if the **-w** option as passed to perl.

$^X The name by which this perl was invoked.

The following variables are context dependent and need not be localized:

$% The current page number of the currently selected output channel.

$= The page length of the current output channel. Default is 60 lines.

$- The number of lines left on the page.

$~ The name of the current report format.

$^ The name of the current top-of-page format.

$| If set to nonzero, forces a flush after every write or print on the currently selected output channel. Default is 0.

$ARGV
The name of the current file when reading from **<>** .

The following variables are always local to the current block:

$& The string matched by the last pattern match.

$` The string preceding what was matched by the last pattern match.

$' The string following what was matched by the last pattern match.

$+ The last bracket matched by the last search pattern.

$1... **$9**...
Contains the subpattern from the corresponding set of parentheses in the last pattern matched. **$10**... and up are only available if the match contained that many sub-expressions.

25. Special arrays

@ARGV Contains the command line arguments for the script (not including the command name).

@INC Contains the list of places to look for perl scripts to be evaluated by the **do** FILENAME and **require** commands.

@_ Parameter array for subroutines. Also used by **split** if not in array context.

%ENV Contains the current environment.

%INC List of files that have been **require**d or **do**ne.

%SIG Used to set signal handlers for various signals.

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
$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**.

22. Info from system files

passwd

Info is (**$**name, **$**passwd, **$**uid, **$**gid, **$**quota, **$**comment, **$**gcos, **$**dir, **$**shell).

endpwent	Ends lookup processing.
getpwent	Gets next info.
getpwnam(NAME**)**	Gets info by name.
getpwuid(UID**)**	Gets info by uid.
setpwent	Resets lookup processing.

group

Info is a 4-item array: (**$**name, **$**passwd, **$**gid, **$**members).

endgrent	Ends lookup processing.
getgrgid(GID**)**	Gets info by group id.
getgrnam(NAME**)**	Gets info by name.
getgrent	Gets next info.
setgrent	Resets lookup processing.

hosts

Info is (**$**name, **$**aliases, **$**addrtype, **$**length, @addrs).

endhostent	Ends lookup processing.
gethostbyname(NAME**)**	Gets info by name.
gethostent	Gets next info.
sethostent(STAYOPEN**)**	Resets lookup processing.

networks

Info is (**$**name, **$**aliases, **$**addrtype, **$**net).

endnetent	Ends lookup processing.
getnetbyaddr(ADDR**,**TYPE**)**	Gets info by address and type.
getnetbyname(NAME**)**	Gets info by name.
getnetent	Gets next info.
setnetent(STAYOPEN**)**	Resets lookup processing.

services

Info is (**$**name, **$**aliases, **$**port, **$**proto).

endservent	Ends lookup processing.
getservbyname(NAME**,** PROTO**)**	Gets info by name.
getservbyport(PORT**,** PROTO**)**	Gets info by port.
getservent	Gets next info.
setservent(STAYOPEN**)**	Resets lookup processing.

protocols

Info is (**$**name, **$**aliases, **$**proto).

endprotoent	Ends lookup processing.
getprotobyname(NAME**)**	Gets info by name.
getprotobynumber(NUMBER**)**	Gets info by number.
getprotoent	Gets next info.
setprotoent(STAYOPEN**)**	Resets lookup processing.

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".

18. Networking

accept(NEWSOCKET,GENERICSOCKET**)**
Accepts a new socket.

bind(SOCKET,NAME**)**
Binds the NAME to the SOCKET.

connect(SOCKET,NAME**)**
Connects the NAME to the SOCKET.

getpeername(SOCKET**)**
Returns the socket address of the other end of the SOCKET.

getsockname(SOCKET**)**
Returns the name of the socket.

getsockopt(SOCKET,LEVEL,OPTNAME**)**
Returns the socket options.

listen(SOCKET,QUEUESIZE**)**
Starts listening on the specified SOCKET.

recv(SOCKET,SCALAR,LENGTH,FLAGS**)**
Receives a message on SOCKET.

send(SOCKET,MSG,FLAGS[,TO]**)**
Sends a message on the SOCKET.

setsockopt(SOCKET,LEVEL,OPTNAME,OPTVAL**)**
Sets the requested socket option.

shutdown(SOCKET,HOW**)**
Shuts down a SOCKET.

socket(SOCKET,DOMAIN,TYPE,PROTOCOL**)**
Creates a SOCKET in DOMAIN with TYPE and PROTOCOL.

socketpair(SOCKET1,SOCKET2,DOMAIN,TYPE,PROTOCOL**)**
As socket, but creates a pair of bi-directional sockets.

19. SystemV IPC

The following functions all perform the same action as the corresponding system calls.

msgctl(ID,CMD,ARGS**)**
msgget(KEY,FLAGS**)**
msgsnd(ID,MSG,FLAGS**)**
msgrcv(ID,**$**VAR,SIZE,TYPE,FLAGS**)**
semctl(ID,SEMNUM,CMD,ARG**)**
semget(KEY,NSEMS,SIZE,FLAGS**)**
semop(KEY,...**)**
shmctl(ID,CMD,ARG**)**
shmget(KEY,SIZE,FLAGS**)**
shmread(ID,**$**VAR,POS,SIZE**)**
shmwrite(ID,STRING,POS,SIZE**)**

10. Structure conversion

pack(TEMPLATE,LIST**)**
Packs the values into a binary structure using TEMPLATE.

unpack(TEMPLATE,EXPR**)**
Unpacks the structure EXPR into an array, using TEMPLATE.

TEMPLATE is a sequence of characters as follows:

a / **A**	Ascii string, null / space padded
b / **B**	Bit string in ascending / descending order
c / **C**	Native / unsigned char value
f / **d**	Single / double float in native format
h / **H**	Hex string, low / high nybble first.
i / **I**	Signed / unsigned integer value
l / **L**	Signed / unsigned long value
n / **N**	Short / long in network byte order
s / **S**	Signed / unsigned short value
u / **p**	Uuencoded string / Pointer to a string
x / **@**	Null byte / null fill until position
X	Backup a byte

Each character may be followed by a decimal number which will be used as a repeat count, an ***** specifies all remaining arguments.
If the format is preceded with **%**N, **unpack** returns an N-bit checksum instead.
Spaces may be included in the template for readability purposes.

11. String functions

chop(LIST†**)**
Chops off the last character on all elements of the list; returns the last chopped character. The parentheses may be omitted if LIST is a single variable.

crypt(PLAINTEXT,SALT**)**
Encrypts a string.

eval(EXPR†**)***
EXPR is parsed and executed as if it were a perl program. The value returned is the value of the last expression evaluated. If there is a syntax error or runtime error, an undefined string is returned by eval, and **$@** is set to the error message.

index(STR,SUBSTR[,OFFSET]**)**
Returns the position of SUBSTR in STR at or after OFFSET. If the substring is not found, returns **$[-1**.

length(EXPR†**)***
Returns the length in characters of the value of EXPR.

rindex(STR,SUBSTR[,OFFSET]**)**
Returns the position of the last occurrence of SUBSTR in STR at or before OFFSET.

substr(EXPR,OFFSET[,LEN]**)**
Extracts a substring out of EXPR and returns it. If OFFSET is negative, counts from the end of the string. May be used as an lvalue.

16. Search and replace functions

[EXPR =~] [**m**]/PATTERN/[**g**][**i**][**o**]

Searches EXPR (default: **$_**) for a pattern. If you prepend an **m** you can use almost any pair of characters as delimiters. If used in array context, an array is returned consisting of the subexpressions matched by the parentheses in pattern, i.e. **($1,$2,$3,...)**.

Optional modifiers: **g** matches as many times as possible; **i** searches in a case-insensitive manner; **o** interpolates variables only once.

If PATTERN is empty, the most recent pattern from a previous match or replacement is used.

With **g** the match can be used as an iterator in scalar context.

?PATTERN**?**

This is just like the **/**PATTERN**/** search, except that it matches only once between calls to the reset operator. If PATTERN is empty, the most recent pattern from a previous match or replacement is used.

[$VAR =~] **s/**PATTERN/REPLACEMENT**/**[**g**][**i**][**e**][**o**]

Searches a string for a pattern, and if found, replaces that pattern with the replacement text and returns the number of substitutions made. Otherwise it returns false.

Optional modifiers: **g** replaces all occurrences of the pattern; **e** interprets the replacement string as an expression; **i** and **o** as with **/**PATTERN**/** matching. Almost any delimiter may replace the slashes; if single quotes are used, no interpretation is done on the replacement string.

If PATTERN is empty, the most recent pattern from a previous match or replacement is used.

study[**(**$VAR†**)***]

Study the contents of $VAR in anticipation of doing many pattern matches on the contents before it is next modified.

[$VAR =~] **tr/**SEARCHLIST/REPLACEMENTLIST**/**[**c**][**d**][**s**]

Translates all occurrences of the characters found in the search list with the corresponding character in the replacement list. It returns the number of characters replaced. **y** may be used instead of **tr**.

Optional modifiers: **c** complements the SEARCHLIST; **d** deletes all characters not found in SEARCHLIST; **s** squeezes all sequences of characters that are translated into the same target character into one occurrence of this character.

17. System interaction

alarm(EXPR**)***

Schedules a **SIGALRM** to be delivered after EXPR seconds.

chdir [**(**EXPR**)***]

Changes the working directory, **$ENV{"HOME"}** if EXPR is omitted.

chroot(FILENAME†**)***

Changes the root directory for the process and its children.

die[**(**LIST**)***]

Prints the value of LIST to **STDERR** and exits with the current value of **$!** (errno). If **$!** is 0, exits with the value of **($? >> 8)**. If **($? >> 8)** is 0, exits with 255. LIST defaults to **"Died."**.

13. File operations

Functions operating on a list of files return the number of files successfully operated upon.

chmod(LIST**)***

Changes the permissions of a list of files. The first element of the list must be the numerical mode.

chown(LIST**)***

Changes the owner and group of a list of files. The first two elements of the list must be the numerical uid and gid.

truncate(FILE**,**SIZE**)**

truncates FILE to SIZE. FILE may be a filename or a filehandle.

link(OLDFILE**,**NEWFILE**)**

Creates a new filename linked to the old filename.

lstat(FILE**)**

Like stat, but does not traverse a final symbolic link.

mkdir(DIR**,**MODE**)**

Creates a directory with given permissions. Sets **$!** on failure.

select(RBITS**,**WBITS**,**NBITS**,**TIMEOUT**)**

Performs a *select*(2) system call with the same parameters.

readlink(EXPR†**)***

Returns the value of a symbolic link.

rename(OLDNAME**,**NEWNAME**)**

Changes the name of a file.

rmdir(FILENAME†**)***

Deletes the directory if it is empty. Sets **$!** on failure.

stat(FILE**)**

Returns a 13-element array (**$**dev, **$**ino, **$**mode, **$**nlink, **$**uid, **$**gid, **$**rdev, **$**size, **$**atime, **$**mtime, **$**ctime, **$**blksize, **$**blocks). FILE can be a filehandle, an expression evaluating to a filename, or **_** to refer to the last file test operation.

symlink(OLDFILE**,**NEWFILE**)**

Creates a new filename symbolically linked to the old filename.

unlink(LIST**)***

Deletes a list of files.

utime(LIST**)***

Changes the access and modification times. The first two elements of the list must be the numerical access and modification times.