Notes

Perl Reference Guide

for Perl version 5.001

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

Reference guide designed and created by
Johan Vromans **<jvromans@squirrel.nl>**

Contents

Rev. 5.001.1

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

$| If set to nonzero, forces a flush after every write or print on the output channel currently selected. 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 successful pattern match.

$‘ The string preceding what was matched by the last successful match.

$’ The string following what was matched by the last successful match.

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

$1…**$9**… Contain the subpatterns from the corresponding sets of parentheses in the last pattern successfully matched. **$10**…and up are only available if the match contained that many subpatterns.

26. Special arrays

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

@EXPORT
Names the methods a package exports by default.

@EXPORT_OK
Names the methods a package can export upon explicit request.

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

@ISA List of *base classes* of a package.

@_ 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 included with **require** or **do**.

%OVERLOAD
Can be used to overload operators in a package.

%SIG Used to set signal handlers for various signals.

27. Environment variables

Perl uses the following environment variables.

HOME Used if **chdir** has no argument.

LOGDIR
Used if **chdir** has no argument and **HOME** is not set.

PATH Used in executing subprocesses, and in finding the Perl script if ‘**-S**’ is used.

PERL5LIB
A colon-separated list of directories to look in for Perl library files before looking in the standard library and the current directory.

PERL5DB
The command to get the debugger code.
Defaults to **BEGIN { require ’perl5db.pl’ }**.

PERLLIB
Used instead of **PERL5LIB** if the latter is not defined.

2. Literals

Numeric: **123 1_234 123.4 5E-10 0xff** (hex) **0377** (octal).

String: **’abc’** literal string, no variable interpolation nor escape characters, except **\’** and ****. 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.
\Q quote regexp characters until a **\E** is encountered.

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

Array: **(1,2,3)**. **()** is an empty array.
(1..4) is the same as **(1,2,3,4)**. Likewise **(’abc’..’ade’)**.
qw/foo bar …**/** is the same as **(’foo’,’bar’,**…**)**.

Array reference: **[1,2,3]**.

Hash (associative array): **(**KEY1**,** VAL1**,** KEY2**,** VAL2**,** …**)**.
Also: **(**KEY1 **=>** VAL1**,** KEY2 **=>** VAL2**,** …**)**.

Hash reference: **{**KEY1**,** VAL1**,** KEY2**,** VAL2**,** …**}**.

Code reference: **sub {** STATEMENTS **}**

Filehandles: **STDIN**, **STDOUT**, **STDERR**, **ARGV**, **DATA**.
User-specified: HANDLE, **$**VAR.

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

Here-Is: **<<**IDENTIFIER *See the Perl 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**.
$p = \@var	now **$p** is a reference to array **@var**.
$$p[28]	29th element of array referenced by **$p**. Also: **$p->[28]**.
$var[-1]	last element of array **@var**.
$var[$i][$j]	**$j**-th element of **$i**-th element of array **@var**.
$var{’Feb’}	a value from ‘hash’ (associative array) **%var**.
$p = \%var	now **$p** is a reference to hash **%var**.
$$p{’Feb’}	a value from hash referenced by **$p**. Also: **$p->{’Feb’}**.
$#var	last index of array **@var**.
@var	the entire array; in a scalar context: the number of elements in the array.
@var[3,4,5]	a slice of array **@var**.

24. Information from system files

See the manual about return values in scalar context.

passwd

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

endpwent Ends look-up processing.
getpwent Gets next user information.
getpwnam NAME Gets information by name.
getpwuid UID Gets information by user ID.
setpwent Resets look-up processing.

group

Returns (**$**name, **$**passwd, **$**gid, **$**members).

endgrent Ends look-up processing.
getgrgid GID Gets information by group ID.
getgrnam NAME Gets information by name.
getgrent Gets next group information.
setgrent Resets look-up processing.

hosts

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

endhostent Ends look-up processing.
gethostbyaddr ADDR, ADDRTYPE Gets information by IP address.
gethostbyname NAME Gets information by host name.
gethostent Gets next host information.
sethostent STAYOPEN Resets look-up processing.

networks

Returns (**$**name, **$**aliases, **$**addrtype, **$**net).

endnetent Ends look-up processing.
getnetbyaddr ADDR, TYPE Gets information by address and type.
getnetbyname NAME Gets information by network name.
getnetent Gets next network information.
setnetent STAYOPEN Resets look-up processing.

services

Returns (**$**name, **$**aliases, **$**port, **$**proto).

endservent Ends look-up processing.
getservbyname NAME, PROTO Gets information by service name.
getservbyport PORT, PROTO Gets information by service port.
getservent Gets next service information.
setservent STAYOPEN Resets look-up processing.

protocols

Returns (**$**name, **$**aliases, **$**proto).

endprotoent Ends look-up processing.
getprotobyname NAME Gets information by protocol name.
getprotobynumber NUMBER Gets information by protocol number.
getprotoent Gets next protocol information.
setprotoent STAYOPEN Resets look-up processing.

5. Statements

Every statement is an expression, optionally followed by a modifier, and terminated with a semicolon. The semicolon may be omitted if the statement is the final one in a BLOCK.

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 **{}**. BLOCKs 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]

Program flow can be controlled with:

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.

Special forms are:

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

which are guaranteed to perform BLOCK once before testing EXPR, and

do BLOCK

which effectively turns BLOCK into an expression.

6. Subroutines, packages and modules

&SUBROUTINE LIST
Executes a SUBROUTINE declared by a **sub** declaration, and returns the value of the last expression evaluated in SUBROUTINE .
SUBROUTINE can be an expression yielding a reference to a code object.
The **&** may be omitted if the subroutine has been declared before being used.

bless REF [**,** PACKAGE]
Turns the object REF into an object in PACKAGE. Returns the reference.

caller [EXPR]
Returns an array (**$**package,**$**file,**$**line,...) for a specific subroutine call.

waitpid PID, FLAGS
Performs the same function as the corresponding system call.

warn [LIST]
Prints the message on **STDERR** like **die**, but does not exit.
LIST defaults to **"Warning: something's wrong"**.

21. 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.

22. SystemV IPC

msgctl ID, CMD, ARGS
Calls *msgctl*(2). If CMD is **&IPC_STAT** then ARG must be a variable.

msgget KEY, FLAGS
Creates a message queue for KEY. Returns the message queue identifier.

msgsnd ID, MSG, FLAGS
Sends MSG to queue ID.

msgrcv ID, **$**VAR, SIZE, TYPE, FLAGS
Receives a message from queue ID into VAR.

semctl ID, SEMNUM, CMD, ARG
Calls *semctl*(2).
If CMD is **&IPC_STAT** of **&GETALL** then ARG must be a variable.

8. Arithmetic functions

abs EXPR†
Returns the absolute value of its operand.

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†
Returns 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

chr EXPR†
Returns the character represented by the decimal value EXPR.

gmtime EXPR†
Converts a time as returned by the **time** function to a 9-element array (0:**$**sec, 1:**$**min, 2:**$**hour, 3:**$**mday, 4:**$**mon, 5:**$**year, 6:**$**wday, 7:**$**yday, 8:**$**isdst) with the time analyzed for the Greenwich time zone. **$**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 *ctime*(3) string. In array context, returns a 9-element array with the time analyzed for the local time zone.

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 string EXPR as a vector of unsigned integers, and yields the bit at OFFSET. BITS must be between 1 and 32. May be assigned to.

18. Formats

formline PICTURE, LIST
: Formats LIST according to PICTURE and accumulates the result into **$^A**.

write [FILEHANDLE]
: Writes a formatted record to the specified file, using the format associated with that file.

Formats are defined as follows:

format [NAME] **=**
FORMLIST
.

FORMLIST pictures the lines, and contains the arguments which will give values to the fields in the lines. NAME defaults to **STDOUT** if omitted.
Picture fields are:

@<<<…	left adjusted field, repeat the **<** to denote the desired width;
@>>>…	right adjusted field;
@|||…	centered field;
@#.##…	numeric format with implied decimal point;
@*	a multi-line field.

Use **^** instead of **@** for multi-line block filling.

Use **~** at the beginning of a line to suppress unwanted empty lines.

Use **~~** at the beginning of a line to have this format line repeated until all fields are exhausted.

Set **$-** to zero to force a page break.

See also **$^**, **$~**, **$^A**, **$^F**, **$-** and **$=** in section ‘Special variables’.

19. Directory reading routines

closedir DIRHANDLE
: Closes a directory opened by opendir.

opendir DIRHANDLE, DIRNAME
: Opens a directory on the handle specified.

readdir DIRHANDLE
: Returns the next entry (or an array of entries) in the directory.

rewinddir DIRHANDLE
: Positions the directory to the beginning.

seekdir DIRHANDLE, POS
: Sets position for readdir on the directory.

telldir DIRHANDLE
: Returns the postion in the directory.

20. System interaction

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

chdir [EXPR]
: Changes the working directory.
Uses **$ENV{"HOME"}** or **$ENV{"LOGNAME"}** if EXPR is omitted.

quotemeta EXPR
: Returns EXPR with all regexp meta-characters quoted.

rindex STR, SUBSTR [, OFFSET]
: Returns the position of the last 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 assigned to.

uc EXPR
: Returns an upper case version of EXPR.

ucfirst EXPR
: Returns EXPR with the first character in upper case.

12. Array and list functions

delete $HASH**{**KEY**}**
: Deletes the specified value from the specified hash. Returns the deleted value unless HASH is **tie**d to a package that does not support it.

each %HASH
: Returns a 2-element array consisting of the key and value for the next value of the hash. Entries are returned in an apparently random order. After all values of the hash have been returned, a null array is returned. The next call to **each** after that will start iterating again.

exists EXPR†
: Checks if the specified hash key exists in its hash array.

grep EXPR, LIST
grep BLOCK LIST
: Evaluates EXPR or BLOCK for each element of the LIST, locally setting **$_** to refer to the element. Modifying **$_** will modify the corresponding element from LIST. Returns the array of elements from LIST for which EXPR returned **true**.

join EXPR, LIST
: Joins the separate strings of LIST into a single string with fields separated by the value of EXPR, and returns the string.

keys %HASH
: Returns an array of all the keys of the named hash.

map EXPR, LIST
map BLOCK LIST
: Evaluates EXPR or BLOCK for each element of the LIST, locally setting **$_** to refer to the element. Modifying **$_** will modify the corresponding element from LIST. Returns the list of results.

pop @ARRAY
: Pops off and returns the last value of the array.

push @ARRAY, LIST
: Pushes the values of LIST onto the end of ARRAY.

reverse LIST
: In array context: returns the LIST in reverse order.
In scalar context: returns the first element of LIST with bytes reversed.

scalar @ARRAY
: Returns the number of elements in the array.

stat FILE
: 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). FILE can be a filehandle, an expression evaluating to a filename, or _ to refer to the last file test operation or **stat** call.
Returns a null list if the **stat** fails.

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.

17. Input / Output

In input/output operations, FILEHANDLE may be a filehandle as opened by the **open** operator, a pre-defined filehandle (e.g. **STDOUT**) or a scalar variable which evaluates to the name of a filehandle to be used.

<FILEHANDLE**>**
: In scalar context: reads a single line from the file opened on FILEHANDLE.
In array context: reads the whole file.

< > Reads from the input stream formed by the files specified in **@ARGV**, or standard input if no arguments were supplied.

binmode FILEHANDLE
: Arranges for the file opened on FILEHANDLE to be read or written in *binary* mode as opposed to *text* mode (null-operation on UNIX).

close FILEHANDLE
: Closes the file or pipe associated with the file handle.

dbmclose %HASH
: Deprecated, use **untie** instead.

dbmopen %HASH**,** DBMNAME**,** MODE
: Deprecated, use **tie** instead.

eof FILEHANDLE
: Returns 1 if the next read will return end of file, or if the file is not open.

eof Returns the eof status for the last file read.

eof() Indicates eof on the pseudo-file formed of the files listed on the command line.

fcntl FILEHANDLE**,** FUNCTION**, $**VAR
: Implements the *fcntl*(2) function. This function has non-standard return values. See the manual for details.

fileno FILEHANDLE
: Returns the file descriptor for a given (open) file.

flock FILEHANDLE**,** OPERATION
: Calls *flock*(2) on the file. OPERATION formed by adding 1 (shared), 2 (exclusive), 4 (non-blocking) or 8 (unlock).

getc [FILEHANDLE]
: Yields the next character from the file, or **""** on end of file.
If FILEHANDLE is omitted, reads from **STDIN**.

13. Regular expressions

Each character matches itself, unless it is one of the special characters **+?.*^$()[]{}|**. The special meaning of these characters can be escaped using a '****'.

. matches an arbitrary character, but not a newline unless it is a single-line match (see **m//s**).

(…**)** groups a series of pattern elements to a single element.

^ matches the beginning of the target. In multi-line mode (see **m//m**) also matches after every newline character.

$ matches the end of the line. In multi-line mode also matches before every newline character.

[…**]** denotes a class of characters to match. **[^**…**]** negates the class.

(…**|**…**|**…**)** matches one of the alternatives.

(?# TEXT **)** Comment.

(?: REGEXP **)** Like **(**REGEXP**)** but does not make back-references.

(?= REGEXP **)** Zero width positive look-ahead assertion.

(?! REGEXP **)** Zero width negative look-ahead assertion.

(? MODIFIER **)** Embedded pattern-match modifier. MODIFIER can be one or more of **i**, **m**, **s** or **x**.

Quantified subpatterns match as many times as possible. When followed with a '**?**' they match the minimum number of times. These are the quantifiers:

+ matches the preceding pattern element one or more times.

? matches zero or one times.

***** matches zero or more times.

{N**,**M**}** denotes the minimum N and maximum M match count. **{**N**}** means exactly N times; **{**N**,}** means at least N times.

A '****' escapes any special meaning of the following character if non-alphanumeric, but it turns most alphanumeric characters into something special:

\w matches alphanumeric, including '_', **\W** matches non-alphanumeric.

\s matches whitespace, **\S** matches non-whitespace.

\d matches numeric, **\D** matches non-numeric.

\A matches the beginning of the string, **\Z** matches the end.

\b matches word boundaries, **\B** matches non-boundaries.

\G matches where the previous **m//g** search left off.

\n, **\r**, **\f**, **\t** etc. have their usual meaning.

\w, **\s** and **\d** may be used within character classes, **\b** denotes backspace in this context.

Back-references:

\1…**\9** refer to matched sub-expressions, grouped with **()**, inside the match.

\10 and up can also be used if the pattern matches that many sub-expressions.

See also **$1**…**$9**, **$+**, **$&**, **$`** and **$'** in section 'Special variables'.

With modifier **x**, whitespace can be used in the patterns for readability purposes.