C

Function Reference

Below you'll find an alphabetical list of every function in Perl 5.6 starting with a runthrough of the file tests which are themselves functions. Marked against each function will be the syntax for the function, a brief description of what it does and any directly related functions.

File Tests

Function	Syntax	Description
`-X` (file tests)	`-X` *filehandle* `-X` *expression* `-X`	Runs a file test, as described in Chapter 6, determined by X, where X is one of the following letters: `ABCMORSTWX bcdefgkloprstuwxz` If the filehandle or expression arument is omitted, the file test checks against `$_`, with the exception of `-t`, which tests `STDIN`.

Here's a complete rundown of what each file test checks for.

Test	Meaning
`-A`	How long in days between the last access to the file and latest startup.
`-B`	True if the file is a binary file, (compare with `-T`).
`-C`	How long in days between the last inode change and latest startup.
`-M`	How long in days between the last modification to the file and latest startup.
`-O`	True if the file is owned by a real uid/gid.
`-R`	True if the file is readable by a real uid/gid.
`-S`	True if the file is a socket.

Table continued on following page

Test	Meaning
`-T`	True if the file is a text file, (Compare with `-B`).
`-W`	True if the file is writable by a real uid/gid.
`-X`	True if the file is executable by a real uid/gid.
`-b`	True if the file is a block special file.
`-c`	True if the file is a character special file.
`-d`	True if the file is a directory.
`-e`	True if the file exists.
`-f`	True if the file is a plain file - not a directory.
`-g`	True if the file has the setgid bit set.
`-k`	True if the file has the sticky bit set.
`-l`	True if the file is a symbolic link.
`-o`	True if the file is owned by an effective uid/gid.
`-p`	True if the file is a named pipe or if the filehandle is a named pipe.
`-r`	True if the file is readable by an effective uid/gid.
`-s`	True if the file has nonzero size - returns size of file in bytes.
`-t`	True if the filehandle is opened to a tty.
`-u`	True if the file has the setuid bit set.
`-w`	True if the file is writable by an effective uid/gid.
`-x`	True if the file is executable by an effective uid/gid.
`-z`	True if the file has zero size.

A

Function	Syntax	Description
`abs`	`abs` *value* `abs`	Returns the absolute (non-negative) value of an integer. E.g. `abs(-1)` and `abs(1)` both return `1` as a reuslt. If no *value* argument is given, `abs` returns the absolute value of `$_`.
`accept`	`accept` *newsocket, genericsocket*	Accepts an incoming socket connect with sessions enabled, if applicable.

Function	Syntax	Description
`alarm`	`alarm` *num_seconds* `alarm`	Starts a timer with *num_seconds* seconds on the clock before it trips a `SIGALRM` signal. Before the timer runs out, another call to `alarm` cancels it and starts a new timer with `num_seconds` on the clock. If `num_seconds` equals zero, the previous timer is cancelled without starting a new one.
`atan2`	`atan2` *x, y*	Returns the arctangent of *x/y* within the range -• to •.

B

Function	Syntax	Description
`bind`	`bind` *socket*, *name*	Binds a network address (TCP/IP, UDP, etc) to a *socket*, where *name* should be the packed address for the socket.
`binmode`	`binmode` *filehandle*	Sets the specified *filehandle* to be read in binary mode explicitly for those systems that cannot do this automatically. Unix and MacOS can, and thus `binmode` has no effect under these OS's.
`bless`	`bless` *ref, classname* `bless` *ref*	Takes the variable referenced by *ref* and makes it an object of class *classname.*

C

Function	Syntax	Description
`caller`	`caller` *expression* `caller`	Called within a subroutine, `caller` returns a list of information outlining what called it - the sub's context. Actually returns the caller's package name, its filename and line number of the call. Returns the undefined value if not in a subroutine. If *expression* is used, also returns some extra debugging information to make a stack trace.
`chdir`	`chdir` *new_directory* `chdir`	Changes your current working directory to *new_directory.* If *new_directory* is omitted, the working directory is changed to that one specified in `$ENV(HOME)`.
`chmod`	`chmod` *list*	Changes the permissions on a list of files. The first element of *list* must be the octal representation of the permissions to be given those files.
`chomp`	`chomp` *variable* `chomp` *list* `chomp`	Usually removes `\n` from a string. Actually removes the trailing record separator as set in `$/` from a string or from each string in a list, and then returns the number of characters deleted. If no argument is given, `chomp` acts on `$_`.

Table continued on following page

Function	Syntax	Description
`chop`	`chop` *variable* `chop` *list* `chop`	Removes the last character from a string or from each string in a list, and returns the (last) character chopped. If no argument is given, `chop` acts on `$_`.
`chown`	`chown` *list*	Changes the ownership on a list of files. Within *list*, the first two elements must be the user id and group id of that user and group to get ownership, followed by any number of filenames. Setting -1 for either id means, 'Leave this value unchanged.'
`chr`	`chr` *number* `chr`	Returns ASCII character number *number* as determined by Appendix F. If *number* is omitted, `$_` is used.
`chroot`	`chroot` *directory* `chroot`	Changes the root directory for all further path lookups to *directory*. If *directory* is not given, `$_` is used as the new root directory.
`close`	`close` *filehandle* `close`	Closes the file, pipe, or socket associated with the nominated *filehandle*, resetting the line counter `$.` as well. If *filehandle* is not given, closes the currently selected filehandle. Returns `true` on success.
`closedir`	`closedir` *dirhandle*	Closes the directory opened by `opendir()` given by *dirhandle.*
`connect`	`connect` *socket, address*	Tries to connect to a *socket* at the given *address.*
`cos`	`cos` *num_in_radians*	Calculates and returns the cosine of a number given in radians. If *num_in_radians* is not given, calculates the cosine of `$_`.
`crypt`	`crypt` *plaintext, key*	A one-way encryption function (there is no `decrypt` function) that takes some *plaintext* (a password usually) and encrypts it with a two character *key.*

D

Function	Syntax	Description
`dbmclose`	`dbmclose` *hash*	Deprecated in favor of `untie()`. Breaks the binding between a dbm file and the given *hash.*

Function	Syntax	Description
`dbmopen`	`dbmopen` *hash, dbname, mode*	Deprecated in favor of `tie()`. Binds the specified *hash* to the database *dbname.* If the database does not exist, it is created with the specified read\write *mode*, given as an octal number.
`defined`	`defined` *expression* `defined`	Checks whether the value, variable, or function in *expression* is defined. If *expression* is omitted, `$_` is checked.

`delete`	`delete` *$hash{key}*
	`delete` *@hash{keys %hash}*
		Deletes one or more specified *key* and corresponding value from the *hash.* Returns the associated value(s).

`die`	`die` *message*	Writes *message* to the standard error output and then exits the currently running program with `$!` as its return value.
`do`	`do` *filename*	Executes the contents of *filename* as a perl script. Returns `undef` if it cannot read the file. Note: `do` *block* is not a function.
`dump`	`dump` *label* `dump`	Initiates a core dump to be undumped into a new binary executable file, which when run will start at *label.* If *label* is left out, the executable will start from the top of the file.

E

Function	Syntax	Description
`each`	`each` *hash*	Returns the next key/value pair from a *hash* as a two-element list. When *hash* is fully read, returns `null`.
`endgrent`	`engrent`	Frees the resources used to scan the `/etc/group` file or system equivalent.
`endhostent`	`endhostent`	Frees the resources used to scan the `/etc/hosts` file or system equivalent.
`endnetent`	`endnetent`	Frees the resources used to scan the `/etc/networks` file or system equivalent.
`endprotoent`	`endprotoent`	Frees the resources used to scan the `/etc/protocols` file or system equivalent.

Table continued on following page

Function	Syntax	Description
`endpwent`	`endpwent`	Frees the resources used to scan the `/etc/passwd` file or system equivalent.
`endservent`	`endservent`	Frees the resources used to scan the `/etc/services` file or system equivalent.

`eof`	`eof` *filehandle*
	`eof()`
	`eof`
		Returns `1` if *filehandle* is either not open or will return end of file on next read. `eof()` checks for the end of the pseudo file containing the files listed on the command line as program was run. If `eof` does not have an argument, it will check the lst file to be read.

`eval`	`eval` *string* `eval` *block* `eval`	Parses and executes *string* as if it were a mini-program and returns its result. If no argument is given, it evaluates `$_`. If an error occurs or `die()` is called `eval`, returns `undef`. Works similarly with *block* except `eval` *block* is parsed only once. `eval` *string* is reparsed each time `eval` executes.
`exec`	`exec` *command*	Abandons the current program to run the specified system *command.*
`exists`	`exists` *$hash{$key}*	Returns `true` if the specified *key* exists within the specified *hash.*
`exit`	`exit` *status*	Terminates current program immediately with return value *status.* (N.B. The only universally recognized return values are 1 for failure and 0 for success.)
`exp`	`exp` *number*	Returns the value of e to the power of *number* (or `$_` if number is omitted).

F

Function	Syntax	Description
`fcntl`	`fcntl` *filehandle, function, args*	Calls the `fcntl` function, to use on the file or device opened with *filehandle.*
`fileno`	`fileno` *filehandle*	Returns the file descriptor for *filehandle.*
`flock`	`flock` *filehandle, locktype*	Tries to lock or unlock a write-enabled file for use by the program. Note that this lock is only advisory and that other systems not supporting flock will be able to write to the file. *locktype* can take one of four values; `LOCK_SH` (new shared lock), `LOCK_EX` (new exclusive lock), `LOCK_UN` (unlock file), and `LOCK_NB` (do not block access to the file for a new lock if file not instantly available). Returns `true` for success, `false` for failure.

Function	Syntax	Description
fork	fork	System call that creates a new system process also running this program from the same point the fork was called. Returns the new process' id to the original program, 0 to the new process, or undef if the fork did not succeed.
format	format	Declares an output template for use with write().
formline	formline *template, list*	An internal function used for formats. Applies *template* to the *list* of values and stores the result in $^A. Always returns true.

G

Function	Syntax	Description
getc	getc *filehandle* getc	Waits for the user to press Return and then retrieves the next character from *filehandle*'s file. Returns undef if at the end of a file. If *filehandle* is omitted, uses STDIN instead.
getgrent	getgrent	Gets the next group record from /etc/group or the system equivalent, returning an empty record when the end of the file is reached.
getgrgid	getgrgid *gid*	Gets the group record from /etc/group or the system equivalent whose id field matches the given group number *gid*. Returns an empty record if no match occurs.
getgrnam	getgrnam *name*	Gets the group record from /etc/group or the system equivalent whose name field matches the given group *name*. Returns an empty record if no match occurs.
gethostbyaddr	gethostbyaddr *address, addrtype* gethostbyaddr *address*	Returns the hostname for a packed binary network *address* of a certain address type. By default, *addrtype* is assumed to be IP.
gethostbyname	gethostbyname *hostname*	Returns the network address given its corresponding *hostname.*

Table continued on following page

Function	Syntax	Description
`gethostent`	`gethostent`	Gets the next network host record from `/etc/hosts` or the system equivalent, returning an empty record when the end of the file is reached.
`getlogin`	`getlogin`	Returns the user id for the currently logged in user.
`getnetbyaddr`	`getnetbyaddr` *address, addrtype* `getnetbyaddr` *address*	Returns the net name for a given network *address* of a certain address type. By default, *addrtype* is assumed to be IP.
`getnetbyname`	`getnetbyname` *name*	Returns the net address given its corresponding net *name.*
`getnetent`	`getnetent`	Gets the next entry from `/etc/networks` or the system equivalent, returning an empty record when the end of the file is reached.
`getpeername`	`getpeername` *socket*	Returns the address for the other end of the connection to this *socket.*
`getpgrp`	`getpgrp` *process_id* `getpgrp`	Returns the process group in which the specified process is running. Assumes current process if *process_id* is not given.
`getppid`	`getppid`	Returns the process id of the current process' parent process.
`getpriority`	`getpriority` *type, id*	Returns current priority for a process, process group, or user as determined by *type.*
`getprotobyname`	`getprotobyname` *name*	Returns the number for the protocol given in *name.*
`getprotobynumber`	`getprotobynumber` *number*	Returns the name of the protocol given its *number.*
`getprotoent`	`getprotoent`	Gets the next entry from `/etc/protocols` or the system equivalent, returning an empty record when the end of the file is reached.
`getpwent`	`getpwent`	Gets the next entry from `/etc/passwd` or the system equivalent, returning an empty record when the end of the file is reached.
`getpwnam`	`getpwnam` *name*	Gets the password record whose login name field matches the given *name.* Returns an empty record if no match occurs.

Function	Syntax	Description
getpwuid	getpwuid *uid*	Gets the password record whose user id field matches the given *uid*. Returns an empty record if no match occurs.
getservbyname	getservbyname *name, protocol*	Returns the port number for the *name*d service on the given *protocol.*
getservbyport	getservbyport *port, protocol*	Returns the port name for the service *port* on the given *protocol.*
getservent	getservent	Gets the next entry from /etc/services or the system equivalent, returning an empty record when the end of the file is reached.
getsockname	getsockname *socket*	Returns the address for this end of the connection to this *socket.*
getsockopt	getsockopt *socket, level,optname*	Returns the specified socket option or undef if an error occurs.
glob	glob *expression* glob	Returns a list of filenames matching the regular *expression* in the current directory. If *expression* is omitted, the comparison is made with $_.
gmtime	gmtime	Returns a nine-element integer array representing the given *time* (or time() if not given) converted to GMT. By index order, the nine elements (all zero-based) represent: 0 Number of seconds in the current minute. 1 Number of minutes in the current hour. 2 Current hour. 3 Current day of month. 4 Current month.

Table continued on following page

Function	Syntax	Description
gmtime (cont.)	gmtime *time* (cont.)	5 Number of years since 1900 6 Weekday (Sunday = 0) 7 Number of days since January 1. 8 Whether daylight savings time is in effect.
goto	goto *tag* goto *expression* goto *&subroutine*	Looks for *tag* either given literally or dynamically derived by resolving expression and resumes execution of the program there on the provision that it is not inside a construct that requires initializing. For example, a for loop. Alternatively, goto *&subroutine* switches a call to *subroutine* for the currently running subroutine.
grep	grep *expression, list* grep {*block*} *list*	Evaluates a given *expression* or *block* of code against each element in *list* and returns a list of those elements for which the evaluation returned true.

H

Function	Syntax	Description
hex	hex *string* hex	Reads in *string* as a hexadecimal number and returns the corresponding decimal equivalent. Uses $_ if string is omitted.

I

Function	Syntax	Description

import	import *module list*
	import *module*
		Patches a module's namespace into your own, incorporating the *list*ed subroutines and variables into your own package (or all of them if *list* isn't given).

Function	Syntax	Description
index	index *string, substring, position* index *string, substring*	Returns the zero-based position of *substring* in *string* first occuring after character number *position.* Assumes *position* equals zero if not given. Returns -1 if match not found.
int	int *number* int	Returns the integer section of *number* or $_ if *number* is omitted.
ioctl	ioctl *filehandle, function, argument*	Calls the ioctl function, to use on the file or device opened with *filehandle.*

J

Function	Syntax	Description
join	join *character*, *list*	Returns a single string comprising the elements of *list*, separated from each other by *character*.

K

Function	Syntax	Description
keys	keys *hash*	Returns a non-ordered list of the keys contained in *hash.*
kill	kill *signal*, *process_list*	Sends a *signal* to the processes and/or process groups in process_list. Returns number of signals successfully sent.

L

Function	Syntax	Description
last	last *label* last	Causes the program to break out of the *label*ed loop (or the innermost loop, if *label* is not given) surrounding the command and to continue with the statement immediately following the loop.
lc	lc *string*	Returns *string* in lower case or $_ in lower case if *string* is omitted.
lcfirst	lcfirst *string*	Returns *string* with the first character in lower case. Works on $_ if *string* is omitted.

Table continued on following page

Function	Syntax	Description
`length`	`length` *expression*	Evaluates *expression* and returns the number of characters in that value. Returns `length $_` if *expression* is omitted.
`link`	`link` *thisfile*, *thatfile*	Creates a hard link in the filesystem, from *thatfile* to *thisfile.* Returns `true` on sucess, `false` on failure.
`listen`	`listen` *socket*, *max_connectons*	Listens for connections to a particular *socket* on a server and reports when the number of connections exceeds *max_connections.*
`local`	`local` *var*	Declares a 'private' variable that is available to the subroutine in which it is declared and any other subroutines that may be called by this subroutine. Actually creates a temporary value for a global variable for the duration of the subroutine's execution.
`localtime`	`localtime` *time*	Returns a nine-element array representing the given *time* (or `time()` if not given) converted to system local time. See `gmtime()` for desription of elements.
`log`	`log` *number*	Returns the natural logarithm for a *number.* That is, returns *x* where e^x=*number.*
`lstat`	`lstat` *filehandle* `lstat` *expression* `lstat`	Returns a thirteen element status array for the symbolic link to a file and not the file itself. See `stat()` for further details.

M

Function	Syntax	Description
`m//`	`m//`	Tries to match a regular expression pattern against a string.
`map`	`map` *expression, list* `map` {*block*} *list*	Evaluates a given *expression* or *block* of code against each element in *list* and returns a list of the results of each evaluation.
`mkdir`	`mkdir` *dirname, mode*	Creates a directory called *dirname* and gives it the read\write permissions as specified in *mode* (an octal number).
`msgctl`	`msgctl` *id, cmd, arg*	Calls the System V IPC `msgctl` function.
`msgget`	`msgget` *key, flags*	Calls the System V IPC `msgget` function.
`msgrcv`	`msgrcv` *id, var, size, type, flags*	Calls the System V IPC `msgrcv` function.

Function	Syntax	Description
msgsnd	msgsnd *id, msg, flags*	Calls the System V IPC msgsnd function.
my	my *variable_list*	Declares the variables in *variable_list* to be lexically local to the block or file it has been declared in.

N

Function	Syntax	Description
next	next *label* next	Causes the program to start the next iteration of the *label*led loop (or the innermost loop, if *label* is not given) surrounding the command.
no	no *module_name*	Removes the functionality and semantics of the named module from the current package. Compare with use() which does the opposite.

O

Function	Syntax	Description
oct	oct *string* oct	Reads in *string* as an octal number and returns the corresponding decimal equivalent. Uses $_ if string is omitted.
open	open *filehandle, filename* open *filehandle*	Opens the file called *filename* and associates it with *filehandle*. If *filename* is omitted, open assumes that the file has the same name as *filehandle*.
opendir	opendir *dirhandle,dirname*	Opens the directory called *dirname* and associates it with *dirhandle*.
ord	ord *expression*	Returns the numerical ASCII value of the first character in *expression*.

P

Function	Syntax	Description
pack	pack *template, list*	Takes a *list* of values and puts them into a binary structure using *template* (a sequence of characers as shown below) to give the structure an ordered composition. The possible characters for *template* are:

Table continued on following page

Function	Syntax	Description
`pack` (cont.)	`pack` *template, list* (cont.)	a Null-padded ASCII string A Space-padded ASCII string. b A bit string (low-to-high). B A bit string (high-to-low). c A signed char value. C An unsigned char value. d A double-precision float in the native format. f A single-precision float in the native format. h A hexadecimal string, low to high. H A hexadecimal string, high to low. i A signed integer. I An unsigned integer. l A signed long value. L An unsigned long value. n A big-endian short (16-bit) value. N A big-endian long (32-bit) value. p A pointer to a null-terminated string. P A pointer to a fixed-length string. q A signed quad (64-bit) value. Q An unsigned quad (64-bit) value. s A signed short (16-bit) value. S An unsigned short (16-bit) value. v A little-endian short (16-bit) value. V A big-endian long (32-bit) value. u A uuencoded string. w A BER compressed integer - an unsigned integer in base 128, high-bit first. x A null byte. X Back up a byte.

Function	Syntax	Description
`pack` (cont.)	`pack` *template, list* (cont.)	Z A null-padded, null-terminated string.
		@ Null-fill to absolute position.
`package`	`package` *namespace*	Declares that the following block of code is to be defined within the specified *namespace.*
`pipe`	`pipe` *readhandle, writehandle*	Opens and connects two filehandles, such that the pipe reads content from *readhandle* and passes it to *writehandle.*
`pop`	`pop` *array* `pop`	Removes and returns the last element (at largest index position) from *array*. Pops `@ARGV` if *array* is not specified.
`pos`	`pos` *scalar*	Returns the position in *scalar* of the character following the last `m//g` match. Uses `$_` for *scalar* if omitted.
`print`	`print` *filehandle list* `print` *list* `print`	Prints a *list* of comma-separated strings to the file associated with *filehandle* or `STDOUT` if not specified. If both arguments are omitted, prints `$_` to the currently selected output channel.
`printf`	`printf` *filehandle format, list* `printf` *format, list*	As `print()` but prints to the output channel using a specified *format.*
`prototype`	`prototype` *function*	Returns the prototype of a *function* as a string or `undef` if the prototype does not exist.
`push`	`push` *array, list*	Adds the elements of *list* to the *array* at position max_index.

Q

Function	Syntax	Description
`q//`	`q/`*string*/	Alternative method of putting single quotes around a string.
`qq//`	`qq/`*string*/	Alternative method of putting double quotes around a string.
`quotemeta`	`quotemeta` *expression*	Scans through *expression* and returns it having prefixed all non-alphanumeric or -underscore characters with a backslash.

Table continued on following page

Function	Syntax	Description
`qw//`	`qw/`*strings/*	Returns a list of strings, the elements of which are created by splitting a *string* by whitespace or the *strings* sent to `qw//`.
`qx//`	`qx/`*string/*	Alternative method of backtick-quoting a *string* (which now acts as a command-line command.

R

Function	Syntax	Description
`rand`	`rand` *expression*	Evaluates expression and then returns a random value *x* where 0 <= *x* < the value of *expression.*
`read`	`read` *filehandle, scalar, length, offset* `read` *filehandle, scalar, length*	Reads *length* number of bytes in from *filehandle*, placing them in *scalar*. Starts by default from the start of the file, but you can specify *offset*, the position in the file you wish to start reading from. Returns the number of bytes read, zero if at the end of the file or `undef` if file doesn't exist.
`readdir`	`readdir` *dirhandle*	Returns the next entry in the directory specified by *dirhandle* or if being used in list context, the entire contents of the directory.
`readline`	`readline` *filehandle*	Returns a line from *filehandle*'s file if in scalar context or returns a list containing all the lines of the file as its elements.
`readlink`	`readlink` *linkname*	Returns the name of the file at the end of symbolic link *linkname.*
`readpipe`	`readpipe` *command*	Executes *command* on the command line and then returns the standard output generated by it as a string. Returns a list of lines from the standard output if in list context.
`recv`	`recv` *socket, scalar, length, flags*	Receives a *length* byte message over the named *socket* and reads it into a *scalar* string.
`redo`	`redo` *label* `redo`	Causes the program to restart the current iteration of the *label*ed loop (or the innermost loop, if *label* is not given) surrounding the command without checking the `while` condition.
`ref`	`ref` *reference* `ref`	Returns the type of object being referenced by *reference* or a package name if the object has been blessed into a package.

Function	Syntax	Description
`rename`	`rename` *oldname, newname*	Renames file *oldname* as *newname*. Returns 1 for success or 0 if otherwise.
`require`	`require` *file* `require` *package* `require` *num* `require`	Ensures that the named *package* or *file* are included at runtime. If *num* is argument, ensure that version of Perl currently running is greater than or equal to *num* (or $_ if omitted).
`reset`	`reset` *expression*	Resets all variables in current package beginning with one of the characters in *expression* and all `??` searches to their original state.
`return`	`return` *expression*	Returns the value of *expression* from a subroutine or `eval()`.
`reverse`	`reverse` *list*	Returns either *list* with its elements in reverse order if in list context or a string consisting of the elements of *list* concatenated together and then written backwards.
`rewinddir`	`rewinddir` *dirhandle*	Resets the point of access for `readdir()` to the top of directory *dirhandle.*
`rindex`	`rindex` *string, substring, position* `rindex` *string, substring*	Returns the zero-based position of the last occurence of *substring* in *string* at or before character number *position.* Returns `-1` if match not found.
`rmdir`	`rmdir` *dirname*	If directory *dirname* (or that given in `$_` if omitted) is empty, it is removed. Returns `true` on success or `false` if otherwise.

S

Function	Syntax	Description
`s///`	`s/`*matchstring*/*replacestring*/	Searches for *matchstring* in `$_` and replaces it with *replacestring* if found.
`scalar`	`scalar` *expression*	Evaluates *expression* in scalar context and returns the resultant value.
`seek`	`seek` *filehandle, position, flag*	Sets the *position* (character number) in a file denoted by *filehandle* from which the file will be read/written. *Flag* tells seek whether to goto character number *position* (*flag* = 0), number current position + *position* (*flag* = 1) or number EOF + *position* (*flag*=2). Returns 1 on success or 0 if otherwise.

Table continued on following page

Function	Syntax	Description
`seekdir`	`seekdir` *dirhandle, position*	Sets the *position* (entry number) in a directory denoted by *dirhandle* from which directory entries will be read.
`select`	`select` *filehandle* `select`	Changes the current default filehandle (starts as `STDOUT`) to *filehandle.* Returns the current default filehandle if *filehandle* is omitted.
`select`	`select` *rbits, wbits, ebits, timeout*	Calls the system select command to wait for *timeout* seconds until one (if any) of your filehandles become available for reading or writing and returns either success or failure.
`semctl`	`semctl` *id, sem_num, command, argument*	Calls the System V IPC `semctl` function.
`semget`	`semget` *id, semnum, command, argument*	Calls the System V IPC `semget` function.
`semop`	`semop` *key, opstring*	Calls the System V IPC `semop` function.
`send`	`send` *socket, message, flags, destination* `send` *socket, message, flags*	Sends a *message* from a *socket* to the connected socket, or, if *socket* is disconnected, to *destination.* Takes account of any system *flags* given it.
`setpgrp`	`setpgrp` *process_id, process_group*	Sets the *process_group* in which the process with the given *process_id* should run. The arguments default to 0 if not given.
`setpriority`	`setpriority` *which, id, priority*	Adds to or diminshes the priority of either a process, process group, or user, as determined by *which* and specifically identified by its *id.*
`setsockopt`	`setsockopt` *socket, level, option, optional_value*	Sets the *option* for the given *socket.* Returns `undef` if an error occurs.
`shift`	`shift` *array* `shift`	Returns the element at position 0 in array and then removes it from array. Returns `undef` if there are no elements in the array. `Shifts @_` within subroutines and formats or `@ARGV` otherwise if *array* is omitted.
`shmctl`	`shmctl` *id, command, argument*	Calls the System V IPC `shmctl` function.

Function	Syntax	Description
shmget	shmget *key, size, flags*	Calls the System V IPC shmget function.
shmread	shmread *id, variable, position, size*	Calls the System V IPC shmread function.
shmwrite	shmwrite *id, string, position, size*	Calls the System V IPC shmwrite function.
shutdown	shutdown *socket, manner*	Shuts down the *socket* specified in the following *manner*. 0 Stop reading data. 1 Stop writing data. 2 Stop using this socket altogether.
sin	sin *expression* sin	Evaluates *expression* as a value in radians and then returns the sine of that value. Returns the sine of $_ if exressino is omitted.
sleep	sleep *n* sleep	Causes the running script to 'sleep' for *n* seconds or forever if *n* is not given.
socket	socket *filehandle, domain, type, protocol*	Opens a socket and associates it to the given *filehandle.* This socket exists within the given *domain* of communcation, is of the given *type* and uses the given *protocol* to communicate.
socketpair	socketpair *sock1, sock2, domain, type, protocol*	Creates a pair of sockets named sock1 and sock2. These sockets exist within the given *domain* of communcation, are of the given *type* and use the given *protocol* to communicate.
sort	sort *subroutine list* sort *block list* sort *list*	Takes a *list* of values and returns it with the elements after being sorted into an order. If *subroutine* is given, this is used to sort *list.* If *block* is given, this is used as an anonymous subroutine to sort *list.* If neither are given, *list* is sorted by simple string comparisons.

Table continued on following page

Function	Syntax	Description
`splice`	`splice` *array, offset, length, list* `splice` *array, offset, length* `splice` *array, offset*	Takes *array* elements from index *offset* to (*offest+length*) and replaces them with the elements of *list*, if any. If *length* is removed, removes all the elements of array from index *offset* onwards. If negative, leaves that many elements at the end of the array. If offset is negative, splice starts from index number (maxindex-*offset*). Returns the last element removed if in scalar context or `undef` if nothing was removed.
`split`	`split` */pattern/, string, limit* `split` */pattern/, string* `split` */pattern/* `split`	Takes the given *string* and returns it as an array of smaller strings where any instances in the string matching *pattern* have been used as the delimiter for the array elements. If given, *limit* denotes the number of times pattern will be searched for in string. If *string* is omitted, `$_` is split. If *pattern* is omitted, `$_` is split by whitespace.
`sprintf`	`sprintf` *format, list*	As `printf()` but prints *list* to a string using a specified *format.*
`sqrt`	`sqrt` *expression* `sqrt`	Evaluates *expression* and then returns the square root of either it or `$_` if it was left out of the call.
`srand`	`srand` *expression* `srand`	Seeds the random-number generator.
`stat`	`stat` *filehandle* `stat` *expression* `stat`	Returns a thirteen-element array comprising the following information about a file named by *expression*, represented by *filehandle* or contained in `$_` (by index number). 0 `$dev` Device number of filesystem
		1 `$ino` Inode number

Function	**Syntax**	**Description**
`stat` (cont.)	`stat` *filehandle*	2 `$mode` File mode.
	`stat` *expression*	
	`stat`	3 `$nlink` number of links to the file.
	(cont.)	
		4 `$uid` User id of file's owner.
		5 `$gid` Group id of file's owner.
		6 `$rdev` Device identifier.
		7 `$size` Total size of file.
		8 `$atime` Last time file was accessed.
		9 `$mtime` Last time file was modified.
		10 `$ctime` Last time inode was changed.
		11 `$blksize` Preferred block size for file I/O.
		12 `$blocks` Number of blocks allocated to file.
`study`	`study` *string* `study`	Tells perl to optimize itself for repeated searches on *string* or on `$_` if *string* is omitted.
`sub`	`sub` *subname block* `sub` *subname* `sub` *block*	Declares a *block* of code to be a subroutine with the name *subname.* If *block* is omitted, this is just a forward reference to a later declaration. If *subname* is omitted, this is an anonymous function declaration.

Table continued on following page

Function	Syntax	Description
`substr`	`substr` *string, position, length, replacement* `substr` *string, position, length* `substr` *string, position*	Returns a substring of *string* that is *length* characters long, starting with the character at index number *position.* If given, that substring is then silently replaced with *replacement.* If *length* is not given, `substr` assumes the entire string from *position* onwards.
`symlink`	`symlink` *oldfile, newfile*	Creates *newfile* as a symbolic link to *oldfile.* Returns 1 on sucess, 0 on failure.
`syscall`	`syscall` *list*	Assumes the first element in the *list* is the name of a system call and calls it, taking the other elements of the *list* to be arguments to that call.
`sysopen`	`sysopen` *filehandle, filename, mode, permissions* `sysopen` *filehandle, filename, mode*	Opens file *filename* under the specified *mode* and associates it with the given *filehandle. Permissions* is the octal value representing the permissions that you want to assign to the file. If not given, the default is `0666`.
`sysread`	`sysread` *filehandle, scalar, length, offset* `sysread` *filehandle, scalar, length*	Reads *length* number of bytes in from *filehandle*, placing them in *scalar* using the system call `read`. Starts by default from the start of the file, but you can specify *offset*, the position in the file you wish to start reading from. Returns the number of bytes read, 0 if at the end of the file or `undef` if file doesn't exist.
`sysseek`	`sysseek` *filehandle, pos, flag*	Sets the system position for the file denoted by *filehandle. Flag* tells `sysseek` whether to goto position number *pos* (*flag* = 0), number current position + *pos* (*flag* = 1), or number EOF + *pos* (*flag* = 2). Returns 1 on success, 0 otherwise.
`system`	`system` *list*	Forks the process that the current program is running on, lets it complete, and then abandons the current program to run the specified system command in *list.* This will be the first element of *list*, and any arguments to it are stored in subsequent list elements.
`syswrite`	`syswrite` *filehandle, scalar, length, offset* `syswrite` *filehandle, scalar, length* `syswrite` *filehandle, scalar*	Writes *length* number of bytes from the *scalar* to the file denoted by *filehandle*, starting at character number *offset* if specified. If *length* is not given, writes the entire scalar to the file.

T

Function	Syntax	Description
`tell`	`tell` *filehandle* `tell`	Returns the current read/write position for the file marked by *filehandle.* If filehadle is not given, the info is given for the last accessed file.
`telldir`	`telldir` *dirhandle*	Returns the current `readdir` position for the directory marked by *dirhandle.*
`tie`	`tie` *variable, classname, list*	Binds the named *variable* to package class *classname*, which works on a variable of that type. Passes any arguments (in *list*) to the `new` function of the class (`TIESCALAR`, `TIEHASH`, or `TIEARRAY`).
`tied`	`tied` *variable*	Returns a reference to the object tied to the given *variable.*
`time`	`time`	Returns the number of non-leap seconds elpased since Jan 1, 1970. Can be translated into recognizable time values with `gmtime()` and `localtime()`.
`times`	`times`	Returns a four-element list holding the user and system CPU times (in seconds) for both the current process and its child processes. The list is comprised as follows: $user Current process user time. $system Current process system CPU time. $cuser Child process user time. $csystem Child process system time.
`tr///`	`tr/`*string1*`/`*string2*`/`	Transliterates a string (also known as `y///`).
`truncate`	`truncate` *filehandle, length* `truncate` *expression, length*	Truncates the file given by *filehandle* or named literally by *expression* to *length* characters. Returns `true` on success, `false` if otherwise.

U

Function	Syntax	Description
`uc`	`uc` *string* `uc`	Returns *string* in upper case or `$_` in upper case if *string* is omitted.
`ucfirst`	`ucfirst` *string* `ucfirst`	Returns *string* with the first character in upper case. Works on `$_` if *string* is omitted.
`umask`	`umask` *expression* `umask`	Returns the current umask and then sets it to *expression* if this is given. The umask is a group of three octal numbers representing the access permissions for a file or directory of its owner, a group and other users, where execute = 1, write = 2, and read = 4. So an umask of 0777 would give all permissions to all three levels of user. 0744 would restrict all except the owner to read access only.
`undef`	`undef` *expression* `undef`	Removes the value of *expression*, leaving it undefined, or else it just returns the undefined value.
`unlink`	`unlink` *list*	Deletes the files specified in *list* (or `$_` if not given), returning the number of files deleted. (N.B. For Unix users, `unlink()` removes a link to each file but not the fields themselves if other links to them still exist.)
`unpack`	`unpack` *template, string*	The reverse of `pack()`. Takes a `packed` *string* and then uses *template* to read through it and return an array of the values stored within it. See `pack()` for how *template* is constructed.
`unshift`	`unshift` *array, list*	Adds the elements of *list* in the same order to the front (index 0) of *array*, returning the number of elements now in *array*.
`untie`	`untie` *variable*	Unbinds the *variable* from the package class it had previously been `tied` to.

Function	Syntax	Description
`use`	`use` *module version list* `use` *module list* `use` *module* `use` *version*	`Requires` and `imports` the (*list*ed elements of the) named *module* at compile time. Checks that module being used is the specified *version* if combined with *module* and *list*. `use` *version* meanwhile makes sure that the perl interpreter used is no older than the stated *version.*
`utime`	`utime` *atime, mtime, filelist*	Sets the access (*atime*) and modification (*mtime*) times on files listed in *filelist*, returning the number of successful changes that were made.

V

Function	Syntax	Description
`values`	`values` *hash*	Takes the named *hash* and returns a list containing copies of each of the values in it.
`vec`	`vec` *string, offset, bits*	Takes *string* and regards it as a vector of unsigned integers. Then returns the value of the element at position *offset*, given that each element has 2 to the power of *bits* in it.

W

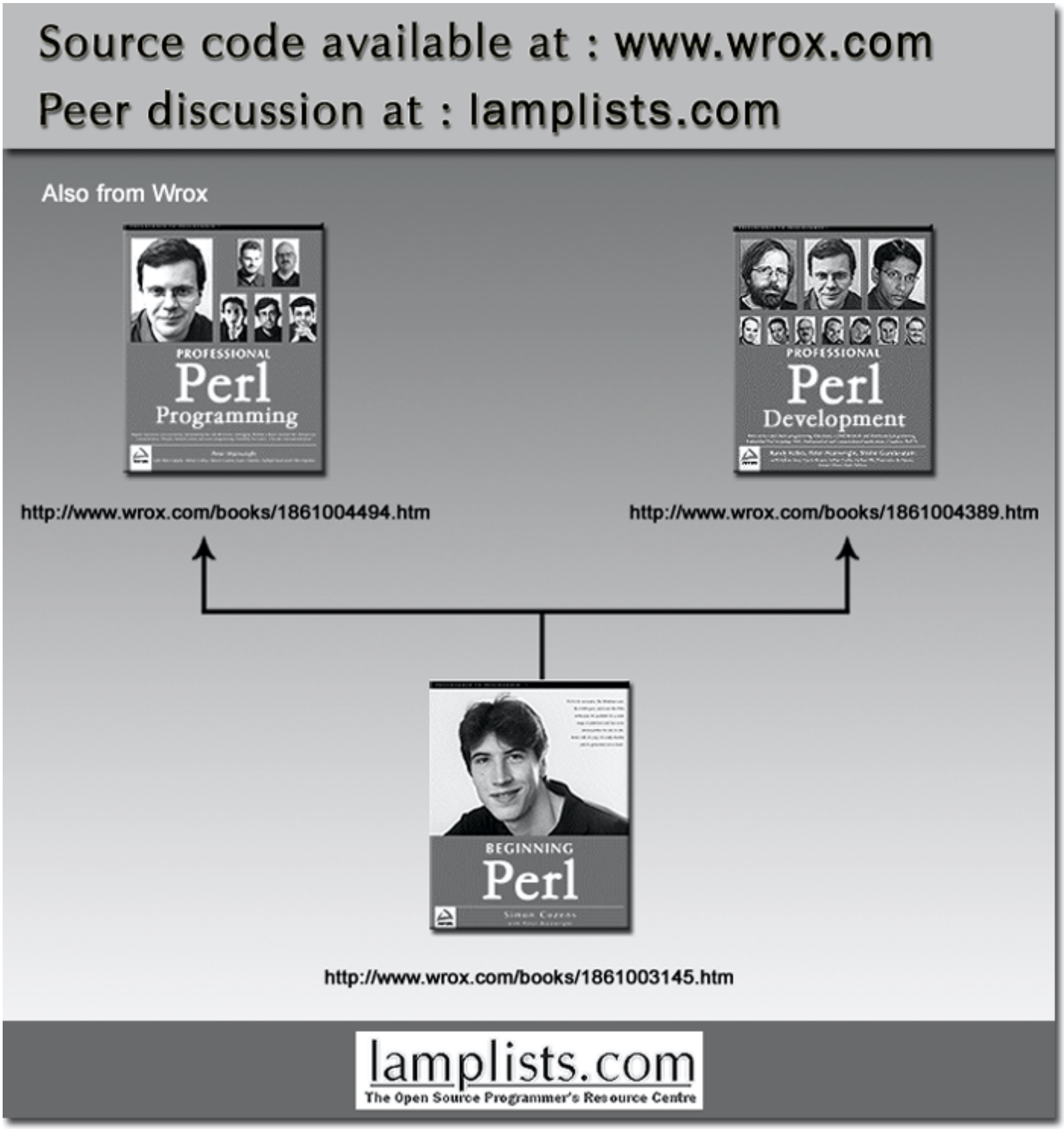
Function	Syntax	Description
`wait`	`wait`	Waits for a(ny) child process to die and then returns the process id of the child process that did or `-1` if there are no child processes.
`waitpid`	`waitpid` *pid, flags*	Waits for the child process with process id *pid* to die

Table continued on following page

Function	Syntax	Description
`wantarray`	`wantarray`	Returns `true` if the subroutine currently running is running in list context. Returns `false` if not. Returns `undef` if the subroutine's return value is not going to be used.
`warn`	`warn` *message*	Prints *message* to `STDERR`, but doesn't throw an error or exception.
`write`	`write` *filehandle* `write` *expression* `write`	Writes a formatted record to *filehandle*, the file named after evaluating *expression*, or the current default output channel if neither are given.

Y

Function	Syntax	Description
`y///`	`y/`*string1*`/`*string2*`/`	Transliterates a string (also known as `tr///`).



PROFESSIONAL
Perl
Programming
PROFESSIONAL
Perl
Development
BEGINNING
Perl
Simon Cozens