

By Jitendra Narayan, BioinformaticsOnline.com

Perl and BioPerl programming is essential part of almost all Bioinformatics and Computational Biology programming. I hope this Perl and BioPerl Cheat Sheet will help in you during research scripting work.

I need your valuable comments on [info@bioinformaticsonline.com](mailto:info@bioinformaticsonline.com)

## 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 a single line of script. Multiple -e commands may be given to build up a multiline script.
- F REGEXP specifies a regular expression to split on if -a is in effect.
- iEXT files processed by the < > construct are to be edited in place.
- IDIR with -P, tells the C preprocessor where to look for include files. The directory is prepended to @INC.
- l [OCTNUM] enables automatic line-end processing, e.g., -l013.
- 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 a switch and sets the corresponding variable \$xxx in the script.
- S uses the PATH environment variable to search for the script.
- T forces taint checking.
- u dumps core after compiling the script. To be used with the undump(1) program (where available).
- U allows perl to perform 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.

By Jitendra Narayan, BioinformaticsOnline.com

- x [DIR]  
extracts Perl program from input stream. If DIR is specified, switches to this directory before running the program.
- 0VAL  
(that's the number zero.) Designates an initial value for the record separator \$/. See also -l

## Literals

### Numeric:

123  
 1\_234  
 123.4  
 5E-10  
 0xff (hex)  
 0377 (octal)

### String:

'abc'

literal string, no variable interpolation or 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 lowercase/uppercase the following character. \L and \U lowercase/uppercase until a \E is encountered. \Q quote regular expression 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 ('a'..'z').  
 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.

By Jitendra Narayan, BioinformaticsOnline.com

## Here-Is:

```
<<IDENTIFIER
Shell-style "here document."
```

## Special tokens:

\_\_FILE\_\_: filename; \_\_LINE\_\_: line number; \_\_END\_\_: end of program; remaining lines can be read using the filehandle DATA.

## Variables

```
$var          a simple scalar variable.
$var[28]      29th element of array @var.
$sp = \@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]  $jth element of the $ith element of array @var.
$var{'Feb'}   one value from hash (associative array) %var.
$sp = \%var   now $p is a reference to hash %var.
$$p{'Feb'}    a value from hash referenced by $p.
              Also, $p->{'Feb'}.

$#var         last index of array @var.
@var          the entire array; in a scalar context, the number of elements in the array.
@var[3,4,5]   a slice of array @var.
@var{'a','b'} a slice of %var; same as ($var{'a'},$var{'b'}).
%var          the entire hash; in a scalar context, true if the hash has elements.
$var{'a',1,...} emulates a multidimensional array.
('a'...'z')[4,7,9] a slice of an array literal.
PKG::VAR      a variable from a package, e.g., $pkg::var, @pkg::ary.
\OBJECT       reference to an object, e.g., \svar, \%hash.
*NAME         refers to all objects represented by NAME.
              *n1 = *n2 makes n1 an alias for n2.
              *n1 = $n2 makes $n1 an alias for $n2.
```

You can always use a {BLOCK} returning the right type of reference instead of the variable identifier, e.g., \${...}, &{...}. \$\$p is just a shorthand for\${\$p}.

## Operators

```
**          Exponentiation
+ - * /     Addition, subtraction, multiplication, division
%           Modulo division
& | ^      Bitwise AND, bitwise OR, bitwise exclusive OR
>> <<      Bitwise shift right, bitwise shift left
| | &&     Logical OR, logical AND
.           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., .=
->         Dereference operator
```

By Jitendra Narayan, BioinformaticsOnline.com

\        Reference (unary)  
 !    ~    Negation (unary), bitwise complement (unary)  
 ++ --    Auto-increment (magical on strings), auto-decrement  
 == !=    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 to  
 <=>cmp    Numeric (string) compare. Returns -1, 0, 1.  
 =~ !~    Search pattern, substitution, or translation (negated)  
 ..        Range (scalar context) or enumeration (array context)  
 ?:        Alternation (if-then-else) operator  
 ,        Comma operator, also list element separator. You can also use =>.  
 not      Low-precedence negation  
 and      Low-precedence AND  
 or    xor    Low-precedence OR, exclusive OR

All Perl functions can be used as list operators, in which case they have very high or very low precedence, depending on whether you look at the left or the right side of the operator. Only the operators not, and, or and xor have lower precedence.

A "list" is a list of expressions, variables, or lists. An array variable or an array slice may always be used instead of a list.

Parentheses can be added around the parameter lists to avoid precedence problems.

## Statements

Every statement is an expression, optionally followed by a modifier, and terminated by 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, for example:

```

EXPR1 if EXPR2 ;
EXPR1 until EXPR2 ;
  
```

The logical operators ||, && or ?: also allow conditional execution:

```

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 ]
  
```

By Jitendra Narayan, BioinformaticsOnline.com

```

[ LABEL: ] while (EXPR) BLOCK [ continue BLOCK ]
[ LABEL: ] until (EXPR) BLOCK [ continue BLOCK ]
[ LABEL: ] for (EXPR; EXPR; EXPR) BLOCK
[ LABEL: ] foreach VAR† (LIST) 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.

## 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. caller returns this info for the current subroutine, caller(1) for the caller of this subroutine, etc. Returns false if no caller.

**do SUBROUTINE LIST**

Deprecated form of &SUBROUTINE.

**goto &SUBROUTINE**

Substitutes a call to SUBROUTINE for the current subroutine.

**import MODULE [ [ VERSION ] LIST ]**

Imports the named subroutines from MODULE.

**no MODULE [ LIST ]**

Cancels imported semantics. See use.

**package NAME**

Designates the remainder of the current block as a package.

By Jitendra Narayan, BioinformaticsOnline.com

### require EXPR†

If EXPR is numeric, requires Perl to be at least that version. Otherwise EXPR must be the name of a file that is included from the Perl library. Does not include more than once, and yields a fatal error if the file does not evaluate to a true value. If EXPR is a bare word, assumes extension .pm for the name of the file.

### return EXPR

Returns from a subroutine with the value specified.

### sub NAME { EXPR ; ... }

Designates NAME as a subroutine. Parameters are passed by reference as array @\_. Returns the value of the last expression evaluated.

### [ sub ] BEGIN { EXPR ; ... }

Defines a setup BLOCK to be called before execution.

### [ sub ] END { EXPR ; ... }

Defines a cleanup BLOCK to be called upon termination.

### tie VAR, PACKAGE, [ LIST ]

Ties a variable to a package that will handle it. Can be used to bind a dbm or ndbm file to a hash.

### untie VAR

Breaks the binding between the variable and the package.

### use MODULE [ [ VERSION ] LIST ]

Imports semantics from the named module into the current package.

## Object-Oriented Programming

Perl rules of object oriented programming:

- An object is simply a reference that happens to know which class it belongs to. Objects are blessed, references are not.
- A class is simply a package that happens to provide methods to deal with object references. If a package fails to provide a method, the base classes as listed in @ISA are searched.
- A method is simply a subroutine that expects an object reference (or a package name, for static methods) as the first argument.

Methods can be applied with:

METHOD OBJREF PARAMETERS      OR  
OBJREF->METHOD PARAMETERS

## Arithmetic Functions

<b>abs</b> EXPR†	Returns the absolute value of its operand.
<b>atan2</b> Y,X	Returns the arctangent of Y/X in the range -pi to pi.
<b>cos</b> EXPR†	Returns the cosine of EXPR (expressed in radians).
<b>exp</b> EXPR†	Returns e to the power of EXPR.
<b>int</b> EXPR†	Returns the integer portion of EXPR.
<b>log</b> EXPR†	Returns natural logarithm (base e) of EXPR.
<b>rand</b> [ EXPR ]	Returns a random fractional number between 0 and the value of EXPR. If EXPR is

By Jitendra Narayan, BioinformaticsOnline.com

sin	EXPR†	omitted, returns a value between 0 and 1.
sqrt	EXPR†	Returns the sine of EXPR (expressed in radians).
srand	[ EXPR ]	Returns the square root of EXPR.
time		Sets the random number seed for the rand operator.
		Returns the number of seconds since January 1, 1970. Suitable for feeding to gmtime and localtime.

## 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 a hex string.

**localtime** EXPR†

Converts a time as returned by the time function to time(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.

## 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 / HHex string, low / high nybble first
- i / I Signed / unsigned integer value
- l / L Signed / unsigned long value
- n / NShort / long in network (big endian) byte order

By Jitendra Narayan, BioinformaticsOnline.com

s / S Signed / unsigned short value  
 u / P Uuencoded string / pointer to a string  
 v / V Short / long in VAX (little endian) byte order  
 x / @Null byte / null fill until position  
 X Backup a byte

Each character may be followed by a decimal number that will be used as a repeat count; an asterisk (\*) 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.

## String Functions

**chomp** LIST†

Removes line endings from all elements of the list; returns the (total) number of characters removed.

**chop** LIST†

Chops off the last character on all elements of the list; returns the last chopped character.

**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. See also eval in section Miscellaneous.

**index** STR, SUBSTR [ , OFFSET ]

Returns the position of SUBSTR in STR at or after OFFSET. If the substring is not found, returns -1 (but see \$[ in section Special Variables).

**length** EXPR†

Returns the length in characters of the value of EXPR.

**lc** EXPR

Returns a lowercase version of EXPR.

**lcfirst** EXPR

Returns EXPR with the first character lowercase.

**quotemeta** EXPR

Returns EXPR with all regular expression metacharacters 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 of length LEN 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 uppercased version of EXPR.

**ucfirst** EXPR

Returns EXPR with the first character uppercased.

## Array and List Functions

**delete** SHASH{KEY}

By Jitendra Narayan, BioinformaticsOnline.com

Deletes the specified value from the specified hash. Returns the deleted value unless HASH is tied to a package that does not support this.

**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 with all of 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 the 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.

**scalar** %HASH

Returns a true value if the hash has elements defined.

**shift** [ @ARRAY ]

Shifts the first value of the array off and returns it, shortening the array by 1 and moving everything down. If @ARRAY is omitted, shifts @ARGV in main and @\_ in subroutines.

**sort** [ SUBROUTINE ] LIST

Sorts the LIST and returns the sorted array value. SUBROUTINE, if specified, must return less than zero, zero, or greater than zero, depending on how the elements of the array (available to the routine as \$a and \$b) are to be ordered. SUBROUTINE may be the name of a user-defined routine, or a BLOCK.

**splice** @ARRAY, OFFSET [ , LENGTH [ , LIST ] ]

Removes the elements of @ARRAY designated by OFFSET and LENGTH, and replaces them with LIST (if specified). Returns the elements removed.

**split** [ PATTERN [ , EXPR† [ , LIMIT ] ] ]

By Jitendra Narayan, BioinformaticsOnline.com

Splits a string into an array of strings, and returns it. If LIMIT is specified, splits into at most that number of fields. If PATTERN is also omitted, splits at the whitespace. If not in array context, returns number of fields and splits to @\_. See also Search and Replace Functions.

**unshift** @ARRAY, LIST

Prepends list to the front of the array, and returns the number of elements in the new array.

**values** %HASH

Returns a normal array consisting of all the values of the named hash.

## 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 multiline mode (see m//m) also matches after every newline character.
\$	matches the end of the line. In multiline 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:

By Jitendra Narayan, BioinformaticsOnline.com

`\1 ... \9` refer to matched subexpressions, grouped with `()`, inside the match.  
`\10` and up can also be used if the pattern matches that many subexpressions.

See also `$1 ... $9`, `$+`, `$&`, `$'`, and `$'` in section Special Variables.

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

## Search and Replace Functions

`[ EXPR =~ ] [ m ] /PATTERN/ [ g ] [ i ] [ m ] [ o ] [ s ] [ x ]`

Searches EXPR (default: `$_`) for a pattern. If you prepend an `m` you can use almost any pair of delimiters instead of the slashes. If used in array context, an array is returned consisting of the subexpressions matched by the parentheses in the 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. `m` treats the string as multiple lines; `s` treats the string as a single line; `x` allows for regular expression extensions.  
 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.

`[ $VAR =~ ] s/PATTERN/REPLACEMENT/ [ e ] [ g ] [ i ] [ m ] [ o ] [ s ] [ x ]`

Searches a string for a pattern, and if found, replaces that pattern with the replacement text. It returns the number of substitutions made, if any; if no substitutions are made, it returns false.  
 Optional modifiers: `g` replaces all occurrences of the pattern; `e` evaluates the replacement string as a Perl expression; for any other modifiers, see `/PATTERN/` matching. Almost any delimiter may replace the slashes; if single quotes are used, no interpretation is done on the strings between the delimiters, otherwise the strings are interpolated as if inside double quotes.  
 If bracketing delimiters are used, PATTERN and REPLACEMENT may have their own delimiters, e.g., `s(foo)[bar]`. If PATTERN is empty, the most recent pattern from a previous match or replacement is used.

`[ $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 found in SEARCHLIST that do not have a corresponding character in REPLACEMENTLIST; `s` squeezes all sequences of characters that are translated into the same target character into one occurrence of this character.

`pos SCALAR`

Returns the position where the last `m//g` search left off for SCALAR. May be assigned to.

`study [ $VAR† ]`

Study the scalar variable \$VAR in anticipation of performing many pattern matches on its contents before the variable is next modified.

## File Test Operators

These unary operators takes one argument, either a filename or a filehandle, and test the associated file to see if something is true about it. If the argument is omitted, they test `$_` (except for `-t`, which tests STDIN). If the special argument `_` (underscore) is passed, they use the information from the preceding test or stat call.

By Jitendra Narayan, BioinformaticsOnline.com

- r -w -x File is readable/writable/executable by effective uid/gid.
- R -W -X File is readable/writable/executable by real uid/gid.
- o -O File is owned by effective/real uid.
- e -z File exists/has zero size.
- s File exists and has non-zero size. Returns the 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 modification / access / inode-change time. Measured in days. Value returned reflects the file age at the time the script started. See also \$^T in the section Special Variables.

## 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.

### **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 (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 statcall. 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

By Jitendra Narayan, BioinformaticsOnline.com

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

## Input / Output

In input/output operations, FILEHANDLE may be a filehandle as opened by the open operator, a predefined filehandle (e.g., STDOUT) or a scalar variable that 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 filehandle.

**dbmclose %HASH**

Deprecated, use untie instead.

**dbmopen %HASH, DBMNAME, MODE**

Deprecated, use tie instead.

**eof FILEHANDLE**

Returns true 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.

**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 an empty string on end of file. If FILEHANDLE is omitted, reads from STDIN.

**ioctl FILEHANDLE, FUNCTION, \$VAR**

Performs ioctl(2) on the file. This function has non-standard return values.

**open FILEHANDLE [ , FILENAME ]**

Opens a file and associates it with FILEHANDLE. If FILENAME is omitted, the scalar variable of the same name as the FILEHANDLE must contain the filename.

The following filename conventions apply when opening a file.

"FILE" open FILE for input. Also "<FILE".

">FILE" open FILE for output, creating it if necessary.

">>FILE" open FILE in append mode.

"+<FILE" open FILE with read/write access (file must exist).

By Jitendra Narayan, BioinformaticsOnline.com

"+>FILE" open FILE with read/write access (file truncated).  
 "|CMD" opens a pipe to command CMD; forks if CMD is -.  
 "CMD|" opens a pipe from command CMD; forks if CMD is -.  
 FILE may be &FILEHND in which case the new filehandle is connected to the (previously opened) filehandle FILEHND. If it is &=N, FILE will be connected to the given file descriptor. open returns undef upon failure, true otherwise.

**pipe** READHANDLE, WRITEHANDLE

Returns a pair of connected pipes.

**print** [ FILEHANDLE ] [ LIST† ]

Equivalent to print FILEHANDLE sprintf LIST.

**printf** [(FILEHANDLE) LIST†]

Equivalent to print FILEHANDLE sprintf(LIST).

**read** FILEHANDLE, \$VAR, LENGTH [ , OFFSET ]

Reads LENGTH binary bytes from the file into the variable at OFFSET. Returns number of bytes actually read.

**seek** FILEHANDLE, POSITION, WHENCE

Arbitrarily positions the file. Returns true if successful.

**select** [ FILEHANDLE ]

Returns the currently selected filehandle. Sets the current default filehandle for output operations if FILEHANDLE is supplied.

**select** RBITS, WBITS, NBITS, TIMEOUT

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

**sprintf** FORMAT, LIST

Returns a string formatted by (almost all of) the usual printf(3) conventions.

**sysread** FILEHANDLE, \$VAR, LENGTH [ , OFFSET ]

Reads LENGTH bytes into \$VAR at OFFSET.

**syswrite** FILEHANDLE, SCALAR, LENGTH [ , OFFSET ]

Writes LENGTH bytes from SCALAR at OFFSET.

**tell** [ FILEHANDLE ]

Returns the current file position for the file. If FILENAME is omitted, assumes the file last read.

**write** [ FILEHANDLE ]

Writes a formatted record to the specified file, using the format associated with that file.

## 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

.

By Jitendra Narayan, BioinformaticsOnline.com

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 multiline 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.

Use \$- to zero to force a page break on the next write.

See also \$^, \$~, \$^A, \$^F, \$- and \$= in section Special Variables.

## 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) from the directory.

**rewinddir** DIRHANDLE

Positions the directory to the beginning.

**seekdir** DIRHANDLE, POS

Sets position for readdir on the directory.

**telldir** DIRHANDLE

Returns the position in the directory.

## 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.

**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".

**exec** LIST

Executes the system command in LIST; does not return.

**exit** [ EXPR ]

By Jitendra Narayan, BioinformaticsOnline.com

Exits immediately with the value of `EXPR`, which defaults to 0 (zero). Calls `END` routines and object destructors before exiting.

## **fork**

Does a `fork(2)` system call. Returns the process ID of the child to the parent process and zero to the child process.

## **getlogin**

Returns the current login name as known by the system.

## **getpgrp [ PID ]**

Returns the process group for process `PID` (0, or omitted, means the current process).

## **getppid**

Returns the process ID of the parent process.

## **getpriority WHICH, WHO**

Returns the current priority for a process, process group, or user.

## **glob PAT**

Returns a list of filenames that match the shell pattern `PAT`.

## **kill LIST**

Sends a signal to a list of processes. The first element of the list must be the signal to send (either numeric, or its name as a string).

## **setpgrp PID, PGRP**

Sets the process group for the `PID` (0 means the current process).

## **setpriority WHICH, WHO, PRIORITY**

Sets the current priority for a process, process group, or a user.

## **sleep [ EXPR ]**

Causes the program to sleep for `EXPR` seconds, or forever if no `EXPR`. Returns the number of seconds actually slept.

## **syscall LIST**

Calls the system call specified in the first element of the list, passing the rest of the list as arguments to the call.

## **system LIST**

Does exactly the same thing as `exec LIST` except that a `fork` is performed first, and the parent process waits for the child process to complete.

## **times**

Returns a 4-element array (0: `$user`, 1: `$system`, 2: `$cuser`, 3: `$csystem`) giving the user and system times, in seconds, for this process and the children of this process.

## **umask [ EXPR ]**

Sets the `umask` for the process and returns the old one. If `EXPR` is omitted, returns current `umask` value.

## **wait**

Waits for a child process to terminate and returns the process ID of the deceased process (-1 if none). The status is returned in  `$?` .

## **waitpid PID, FLAGS**

Performs the same function as the corresponding system call.

## **warn [ LIST ]**

Prints the message on `STDERR` like `die`, but doesn't exit. `LIST` defaults to "Warning: something's wrong".

## 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

Works the same as socket, but creates a pair of bidirectional sockets.

## System V IPC

You need to require "sys/ipc.ph" before you can use the symbolic names of the operations.

**msgctl** ID, CMD, ARGS

Calls `msgctl(2)`. If CMD is `&IPC_STAT` then ARGS 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

By Jitendra Narayan, BioinformaticsOnline.com

Calls `semctl(2)`. If `CMD` is `&IPC_STAT` of `&GETALL` then `ARG` must be a variable.

**semget** KEY, NSEMS, SIZE, FLAGS

Creates a set of semaphores for `KEY`. Returns the message semaphore identifier.

**semop** KEY, ...

Performs semaphore operations.

**shmctl** ID, CMD, ARG

Calls `shmctl(2)`. If `CMD` is `&IPC_STAT` then `ARG` must be a variable.

**shmget** KEY, SIZE, FLAGS

Creates shared memory. Returns the shared memory segment identifier.

**shmread** ID, \$VAR, POS, SIZE

Reads at most `SIZE` bytes of the contents of shared memory segment `ID` starting at offset `POS` into `VAR`.

**shmwrite** ID, STRING, POS, SIZE

Writes at most `SIZE` bytes of `STRING` into the contents of shared memory segment `ID` at offset `POS`.

## Miscellaneous

**defined** EXPR

Tests whether the lvalue `EXPR` has an actual value.

**do** FILENAME

Executes `FILENAME` as a Perl script.

Immediate core dump. When reincarnated, starts at `LABEL`.

**eval** { EXPR ; ... }

Executes the code between `{` and `}`. Traps runtime errors as described with `eval(EXPR)`, in the section `String Functions`.

**local** VARIABLE

**local** ( LIST )

Creates a scope for the listed variables local to the enclosing block, subroutine or `eval`.

**my** VARIABLE

**my** ( LIST )

Creates a scope for the listed variables lexically local to the enclosing block, subroutine or `eval`.

**ref** EXPR<sup>†</sup>

Returns a true value if `EXPR` is a reference. Returns the package name if `EXPR` has been blessed into a package.

**reset** [ EXPR ]

Resets ?? searches so that they work again. `EXPR` is a list of single letters. All variables and arrays beginning with one of those letters are reset to their pristine state. Only affects the current package.

**scalar** EXPR

Forces evaluation of `EXPR` in scalar context.

**undef** [ LVALUE ]

Undefines the `LVALUE`. Always returns the undefined value.

**wantarray**

Returns true if the current context expects an array value.

## Information from System Files

By Jitendra Narayan, BioinformaticsOnline.com

• **passwd**

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

<b>endpwent</b>	Ends lookup processing.
<b>getpwent</b>	Gets next user information.
<b>getpwnam</b> NAME	Gets information by name.
<b>getpwuid</b> UID	Gets information by user ID.
<b>setpwent</b>	Resets lookup processing.

• **group**

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

<b>endgrent</b>	Ends lookup processing.
<b>getgrgid</b> GID	Gets information by group ID.
<b>getgrnam</b> NAME	Gets information by name.
<b>getgrent</b>	Gets next information.
<b>setgrent</b>	Resets lookup processing.

• **hosts**

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

<b>endhostent</b>	Ends lookup processing.
<b>gethostbyaddr</b> ADDR, ADDRTYPE	Gets information by IP address.
<b>gethostbyname</b> NAME	Gets information by hostname.
<b>gethostent</b>	Gets next host information.
<b>sethostent</b> STAYOPEN	Resets lookup processing.

• **networks**

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

<b>endnetent</b>	Ends lookup processing.
<b>getnetbyaddr</b> ADDR, TYPE	Gets information by address and type.
<b>getnetbyname</b> NAME	Gets information by network name.
<b>getnetent</b>	Gets next network information.
<b>setnetent</b> STAYOPEN	Resets lookup processing.

• **services**

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

<b>endservent</b>	Ends lookup processing.
<b>getservbyname</b> NAME, PROTO	Gets information by service name.
<b>getservbyport</b> PORT, PROTO	Gets information by service port.
<b>getservent</b>	Gets next service information.
<b>setservent</b> STAYOPEN	Resets lookup processing.

• **protocols**

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

<b>endprotoent</b>	Ends lookup processing.
<b>getprotobyname</b> NAME	Gets information by protocol name.

By Jitendra Narayan, BioinformaticsOnline.com

getprotobynumber	NUMBER	Gets information by protocol number.
getprotoent		Gets next protocol information.
setprotoent	STAYOPEN	Resets lookup processing.

## Special Variables

The following variables are global and should be localized in subroutines:

\$_	The default input and pattern-searching space.
\$.	The current input line number of the last filehandle that was read.
\$/	The input record separator, newline by default. May be multicharacter.
\$,	The output field separator for the print operator.
\$"	The separator that joins elements of arrays interpolated in strings.
\$\	The output record separator for the print operator.
\$#	The output format for printed numbers. Deprecated.
\$*	Set to 1 to do multiline matching within strings. Deprecated, see the m and s modifiers in section Search and Replace Functions.
\$?	The status returned by the last `...` command, pipe close or system operator.
\$]	The perl version number, e.g., 5.001.
\$[	The index of the first element in an array, and of the first character in a substring. Default is 0. Deprecated.
\$;	The subscript separator for multidimensional array emulation. Default is "\034".
\$!	If used in a numeric context, yields the current value of errno. If used in a string context, yields the corresponding error string.
\$@	The Perl error message from the last eval or do EXPR command.
\$:	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 ID of the currently executing Perl program. Altered (in the child process) by fork.
\$<	The real user ID of this process.
\$>	The effective user ID of this process.
\$(	The real group ID of this process.
\$)	The effective group ID of this process.
\$\$A	The accumulator for formline and write operations.
\$\$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.
\$\$L	Formfeed character used in formats.
\$\$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 of the -w option as passed to Perl.
\$\$X	The name by which the currently executing program 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 remaining 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 <>.

By Jitendra Narayan, BioinformaticsOnline.com

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.

## 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.

## 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.

## The Perl Debugger

The Perl symbolic debugger is invoked with perl -d.

- h Prints out a help message.
- T Prints a stack trace.
- s Single steps.
- n Single steps around subroutine call.
- RETURN key Repeats last s or n.
- r Returns from the current subroutine.
- c [ LINE ] Continues (until LINE, or another breakpoint, or exit).
- p EXPR Prints EXPR.
- l [ RANGE ] Lists a range of lines. RANGE may be a number, start-end, start+amount, or a subroutine name. If RANGE is omitted, lists next window.
- w Lists window around current line.

By Jitendra Narayan, BioinformaticsOnline.com

-	Lists previous window.
f FILE	Switches to FILE and starts listing it.
l SUB	Lists the named subroutine.
S	List the names of all subroutines.
/PATTERN/	Searches forward for PATTERN.
?PATTERN?	Searches backward for PATTERN.
b [ LINE [ CONDITION ] ]	Sets breakpoint at LINE; default is the current line.
b SUB [ CONDITION ]	Sets breakpoint at the subroutine.
d [ LINE ]	Deletes breakpoint at the given line.
D	Deletes all breakpoints.
L	Lists lines that have breakpoints or actions.
a LINE COMMAND	Sets an action for line.
A	Deletes all line actions.
< COMMAND	Sets an action to be executed before every debugger prompt.
> COMMAND	Sets an action to be executed before every s, c or n command.
V [ PACKAGE [ VARS ] ]	Lists all variables in a package. Default package is main.
X [ VARS ]	Like V, but assumes current package.
! [ [-]NUMBER ]	Re-executes a command. Default is the previous command.
H [ -NUMBER ]	Displays the last -NUMBER commands of more than one letter.
t	Toggles trace mode.
= [ ALIAS VALUE ]	Sets alias, or lists current aliases.
q	Quits. You may also use your EOF key character.
COMMAND	Executes COMMAND as a Perl statement.

Sorry, but I do not have time to provide free answers to questions about Perl and Bioperl.  
(I'm busy trying to make a living!) So please don't e-mail your questions to me.

**Enjoy the beauty of Perl Programming.**