See also:     Perl @ Wikipedia	<ul> <li>Perl Intro - a quick introduction to Perl. PerlCheat , Learn Perl in Y minutes, or in 2 hours 30 minutes</li> <li>Online Perl books and tutorials : Beginning Perl , Modern Perl (html) , Perl Maven Tutorial, Intro to Perl-old line options , perlrun , perlive , perloco, perlbug / perlthanks</li> <li>Learning Perl LPo, Intermediate Perl IntPo , Mastering Perl or , Effective Perl Programming or Other exist but are not recommended for various reasons.</li> </ul>					
Perl mailing lists  Perl Guidelines and tools	Perl Style Guide, 10 Essential Development Practices,  Books: Perl Best Practices or, Modern Perl Best Practices (course) or  perlcritic script uses Perl:Critic to scan Perl code. The pel-perl-critic command invokes it to check code in buffer.  The perltidy application reformats Perl code. Older perltidy home page. PerlTidy @ Wikipedia, PBP recommended .perltidyrc					
<ul><li>peridoc browser</li><li>In Emacs: C-c C-h F</li></ul>	• perioc: about perioc itself • perioc: table of content: names of all pages • perisyn: Perl syntax • perifunc: Perl built-in functions  • Use perioc to find if a Perl module is installed, as in: perioc local::lib prints the documentation of local::lib if it is installed. • periocal::lib is useful to get modules installed in your home directory or					
CPAN (@ Wikipedia)  Search: meta::cpan CPAN Testers CPANdeps	<ul> <li>The Zen of Comprehensive Archive Networks</li> <li>PAUSE - Perl Authors Upload Server</li> <li>Installing Local Perl Modules with CPAN</li> <li>CPAN Issue tracker: CPAN RT</li> <li>See Also: IntPo</li> </ul> Command line tools interacting with CPAN to install Perl modules of cycles (see also this StackOverflow Q/ <ul> <li>cpan: (requires config. but has defaults). Use local::lib; cpan will be able to install into your ~/perls</li> <li>Type cpan to open the cpan shell, then type install The::Module to install packages.</li> <li>cpanm: (panmand line tools interacting with CPAN to install Perl modules of cycles (see also this StackOverflow Q/</li> <li>cpan: (requires config. but has defaults). Use local::lib; cpan will be able to install into your ~/perls</li> <li>Type cpan to open the cpan shell, then type install The::Module to install packages.</li> <li>cpanm: (panmand line tools interacting with CPAN to install Perl modules of cycles (see also this StackOverflow Q/</li> </ul>					

Last updated on: 2025-02-11

### **Perl scripts**

Writing Perl scripts	Impose strictures in Perl files	Impose strictures in Perl files to prevent errors by adding one of the following use lines. Also see the strictures package.				
Use the following at the beginning of Perl script files.  perldiag @ perldoc	<pre>#!/usr/bin/env perl use strict; use warnings;  # for testing only: use diagnostics;</pre>	#!/usr/bin/perl -w use v5.12; # loads strict use v5.35; # &loads warnings  use diagnostics produces more info but increases startup time.  Alternative: perl -Mdiagnostics . Emacs p	Executable Perl script should have a valid shebang line identifying the appropriate location of the Perl interpreter. It may have to be modified at installation time (OpenGroup/SUS).  It's best to: use warnings; perl -w generates warning for all Perl code in the program including modules used by the program. Also use the _c option to check syntax.  But most Perl code should also activate the strict Perl rules and warnings to detect warnings. See: Barewords in Perl  eel-perl-critic command can report diagnostic.			
use version/features	<u>use</u> v5.36;	This can be used to enable both the strict and warning pramas as well as several <u>named features</u> .  • See the <u>table listing the feature bundles per Perl versions</u> .				
Perl version history     at perldoc  M: minor, P: patch level	Perl Versions Guide     Perl versions @ perldoc  Equivalence between decimal	<ul> <li>5.even: maintenance track version</li> <li>5.odd: development track version</li> <li>and dot-decimal versions: AAA.MMMPP </li> </ul>	decimal: 1.02. # old way     dot-decimal: v5.38.2      AAA.MMM.PP . Note that 3 Minor digits are used in the decimal versions. Patch use 2 or 3.			
Perl 5 Operators						

```
Perl 5 Operators
                                          Perl operators, listed below with their precedence and associativity
                                                                                                                                                                                                               C Operators missing from Perl: unary &, unary * and (type)
                            Note:
                                          • Quote and Quote-like operators: in Perl quotes are operators and they provide various kind of interpolating and pattern matching capabilities
Associativity: one of:
                                          left
                                                             terms and list operators (leftward)
                                                                                                                                                                                                        Note: print, sort, reverse, chmod, are list operators
   right
                                          1eft
                                                             Arrow Operator:
   left
                                          NA
                                                             Auto-increment and Auto-decrement: ++ --
• NA : not associative:
                                          right
                                                             Exponentiation:
   cannot use more than one of these operators
                                          right
                                                             Symbolic Unary Operators:
                                                                                                                             1 ~
                                                                                                                                      -. \ and unary + and -
                                                                                                                                                                                                        Note: The operator \ <u>creates a reference</u>. See <u>example</u>.
                                          left
                                                             Binding operators:
                                                                                                                                   ! ~
    in sequence.
                                          left
                                                                                                                            * /
                                                                                                                                      용
                                                             Multiplicative Operators:

    CH: chained

                                          left.
                                                             Additive Operators:
                                          left
                                                             Shift Operators:
                                                                                                                             <<
                                                                                                                                      >>
                                          NA
                                                             named unary operators
To get this information.
                                          NA
                                                             Class instance Operator:
                                                                                                                           isa
                                          СН
perldoc perlop
                                                                                                                           as numbers: < >
                                                                                                                                                                                        as strings: 1t
                                                             Relational Operators:
                                                                                                                                                                  <=
                                                                                                                                                                                                                        αt
                                                                                                                                                                                                                                  le
                                          CH/NA
                                                                                                                           as numbers: == !=
                                                                                                                                                                                        as strings: eq
                                                             Equality Operators:
                                                                                                                                                                                                                       ne
                                                                                                                                                                                                                                   cmp
Note: or The
                                          left.
                                                             Bitwise And:
Bitwise String Operators
                                          left
                                                             Bitwise Or and Exclusive Or:
                                                                                                                                  1.
                                          left
are:
                                                             C-style Logical And:
                                                                                                                          &&
                                                             Logical Defined-Or:
        ۶.
                                          left.
                                                                                                                           П
                    .= ^.=
                                          NA
        & .=
                                                             Range Operators:
                                          right
                                                                                                                           ?:
                                                             Conditional Operator:
                                          right
                                                             Assignment Operators:
                                                                                                                                                                              \_=
                                                                                                                                                                                                                                                goto last next redo dump
                                          left
                                                             Comma, fat-comma Operators:
                                                                                                                           =>
                                          NA
                                                             <u>list operators (rightward)</u>
                                          right
                                                             Logical Not:
                                          left
                                                                                                                        and
                                                             Logical And:
                                          left
                                                             Logical or and Exclusive or:
                                                                                                                        or xor
                                                                                                                                                                                                                           -+- is - - with a + to put them together. The 0+ is the same, but -+- has higher precedence.
                                                             Converts a string that starts with digits into a number.
                                                                                                                                                             print -+- '22les poulets!';
trick operators
                                                                                                                                                             # prints 22
                                          0+
Do not use in
 production code!
                                          =()=
                                                             Called the 'qoatse' operator. It causes the right side
                                                                                                                                                            my $str = "A 22 before 33 does not make 9, it is 44!";
But understanding how
                                                              expression to be evaluated in array context. Used to assign
                                                                                                                                                            my $digit_count =()= $str =~
print "$digit_count";
                                                                                                                                                                                                                          '\d/g;
these work does help
                                                                                                                                                                                                                        # prints '7',the number of digits in $str
                                                             the array/list size to a scalar.
understand Perl.
                                                                                                              "@{[something]}" is join $", something
These are not real Perl
                                                             Interpolate an array in a string:
                                                                                                                                                            print "these people @{[get_names()]} get promoted"
                                          @{[]}
operators; they are
                                                             the same as:
concatenation of other
operators that achieve a
                                                            Force scalar context.
                                                                                                                     In scalar context localtime returns human readable time.
                                                                                                                                                                                                                            $ perl -le 'print ~~localtime'
                                                                                                                      but in list context it returns a 9-tuple with date elements.
                                                                                                                                                                                                                              on Nov 30 09:06:13 2009
specific effect.

    Negation of a true value by "!" or "not"

Truth and falsehood

    False in a boolean

                                                                                                                                                            So the following scalar values are
                                                                                                                                                                                                                           All other scalar values are true, such as:
                                                                                              returns a special false value.
                                                                                                                                                            considered false:
                                                                                                                                                                                                                               1 any non-0 number
                                             context:
                                                                                                                                                                                                                           ' 'the string with a space in it'00' two or more 0 characters in a string
                                                                                              When evaluated as a string it is
                                                                                                                                                             · undef - the undefined value

 the number 0.

 Remember that the
                                                                                              treated as ", but as a number, it is treated as 0.
                                                  the strings '0' and '',
                                                                                                                                                            • 0 the number 0, even if you write it
strings '0' and " mean
                                              • the empty list (),
                                                                                                                                                                as 000 or 0.0
                                                                                                                                                                                                                               "0\n" a 0 followed by a newline
false. The output of
                                                                                                                                                                " the empty string.
glob() may return a file
                                                  "undef
                                                                                                                                                            • '0', a single 0 in the string.

    'false' . Even the string 'false' evaluates to true.

    All other values are true.

named '0'!
 🔔 a bareword false has
                                                                                                                                                                                                                  use constant { true => 1, false => 0 };

one way to define valid true and false constant symbols that can be used in assignments (but see ←):

in the constant of the constant symbols is the constant symbols.

It is a constant to the constant symbols that can be used in assignments.

It is a constant symbol to the constant symbols that can be used in assignments.

It is a constant symbol to the consta
 a truth value of true!!!
File test operators
See filetest -X
                                          File tests can be \underline{\text{stacked}} (-r -w -e \underline{\text{sfname}}) or combined as in the following example \underline{\underline{\sigma}}:
                                                                                                                                                                                                                  if (-e $fname && -f _ && -r _ ) {
  print("$fname exists, is readable\n"); }
                                              Notice the underscore in the example: it's the virtual filehandle _ accessing the last stat or lstat result :
```

The operators check if is readable by effective uid/gid exists is a block special file. is writable by effective uid/gid is a character special file. is empty. -s -f handle is opened to a tty. See also: -x is executable by effective uid/gid has nonzero size (returns size in bytes). -t File Tests <u>o</u> -о -R is owned by effective uid has setuid bit set. -g -k is readable by real uid/gid -d is a directory. has setaid bit set. · File test operators @ -N -W -X -O -M is writable by real uid/gid has sticky bit set perl tutorial is a symbolic link is a named pipe (FIFO) or Filehandle is a pipe. is an ASCII text file (heuristic quess). is executable by real uid/aid -T qis a "binary" file (opposite of -T).

Days between start time and node change time (in file is owned by **real** uid. -8 is a socket -R · localtime Days between start time and file access time Days between start time and file modification time • IO::Interactive 1

### Perl 5 Constants and Variables

```
Perl Constants
                              • Perl pragma to declare constants. ... But be aware that these are still not read-only, that they inject sub-routines and have several limitations. Read the doc!!
                                <u>CPAN modules for defining constants by Neil Bowers</u>. Of particular interest: <u>Const::Fast</u> and <u>Attribute::Constant</u> for efficient read-only constants.
Perl Variables Names
                                                                                                   Array Naming Conventions
                                                                                                                                                    All: 1st char: underscore or letter. Never use ALLCAPS

    Module names are MixedCaseNoUnderscores
    Constants are UPPERCASE_WITH_UNDERSCORES

Case sensitive. ASCII by default, <u>UTF-8</u> if the <u>utf8</u>
                                All variables: words_with_underscores
                                                                                   Same, but array names should be plural
                                Local variables: $lowercase
                                                                                     @locals
                                                                                     @Global_Array
pragma is used.
                                Global variables:
                                                    $Title_Case
                                                                                                                                                      Package wide vars are Mixed_Case_With_Underscores
                                                                                                                                                      Functions/methods are lowercase_with_underscores
                                                                                     @CONSTANT_ARRAYS
                                Constants:
                                                    SUPPER CASE
                                                                                                              $#days
                                                   Simple scalar value
Perl types
                          $
                             Sfoo
                                                                                                                                    Last index of array @days.
                                                                                                              $days->[28]
Scalar
                              $days[28]
                                                   29th element of array @days
                                                                                                                                    29th element of array pointed to by reference $days.
                                                   Value associated with the Feb key of hash %days
                                                                                                              $days[0][2]
$d{99}{'Feb'}
                              $days{'Feb'}
                                                                                                                                    Multi-dimensional array
                                                   Same as $days, use before alphanumumerics.
                                                                                                                                    Multi-dimensional hash
                              ${days}
                                                   The $days variable inside the Dog package.
                              $Dog::days
                                                                                                                        'Feb'
                                                                                                                                    Multi-dimensional hash emulation
                                                                                                              $d{99,
                                                   Same as above. Archaic use of single quote.
                              $Dog'days
                                                                                                                A list is an ordered collection of scalars (of any type).
list and Array
                              @days
                                                Array containing ($days[0], $days[1], ... #days[$#days])
                              @days[3,4,5] Array <u>slices</u> containing ($days[3], $days[4], $days[5])
@days[3..5] Array <u>slices</u> containing ($days[3], $days[4], $days[5])

    0-based indexed (first)
                                                                                                                 An array is a variable that contains a list
  index is 0).
                                                                                                                Reading beyond the end of array returns undef
   Last index of array
                              • Negative indices used in read access from the end: -1 is last item.
  @name is $#name
                                Use these negative indices to access from the end. Do not compute index with $#name -3, if the list size is 2, this will give invalid results.
                                                                                                              my @extracted = (6, 2, 8, 4);

    array slices LPo

                                Use a slice to select multiple elements from a list, array, or hash.
                                                                                                                                                          my @digits = (0..9);

Don't use a slice when you know you need exactly one element.
An Ivalue slice imposes list context on the righthand side.

                                                                                                              my @choices = @digits[@extracted]
my $mod_time = (state $filename)[9];
                                                                                                                                                          my @one2five = @digits[1..5];
my @premiers = @digit[1, 2, 3, 5, 7];
    Simple explanation

    Assign to array slice to update several values. ➤

                                                                                                              @extracted[1, 3] = (7, 9);

    Anonymous array := a type of array reference. Use it to build nested data structures.
    Array reference allows Perl to treat the array as a single item.

                                What are the advantages of anonymous array? @ StackOverflow

    Anonymous arrays

                                Perlref @ Perldoc, Perl reference tutorial @ Perldoc
                                                                Associative array (hash): keys-value pairs. Can be initialized as:
                                                                                                                                                    Initialize a hash slice with array context:
Hash/associative array
                              용
                                           %days
                                                                  my %days = (Jan => 31, Feb => $leap? 29 : 28, ...)
my %days = ("Jan", 31, 'Feb', $leap? 29 : 28, ...
                                                                                                                                                   @char_to_num{'A' .. 'Z'} = 1 .. 26;
my %rating = (ron =>20, al => 50, steve=80);
Hashes @ Perl Maven
                                                                    Multiple values of a hash can be changed with the following construct:
                                                                                                                                                    my @names = (ron, al);
                                                                                                                                                    @rating{ @names } = (25, 35);
hash slice LPo
                                           @days{'J',F'}
                                                                Hash slice returning a list containing ($days{'J'}, $days{'F'})
                                                               my scores = @rating{ @names }; @rating { @names } = (45, 55);
key-value slices LPor ■
                                        extract/write values:
                                                                & is needed to create reference to subroutine
                                           *foo
                                                                                                              See: Advanced Perl Programming, 1st Edition Section 3.2
Typeglob

    hash variables
    subroutine nam

    format names (See write and select)
    how to format output in Perl?, Perl-Formats

                                                                                                                                                                              6. file handles7. directory handles
7 kinds of package
                                 scalar variables $
variables types
                                                                   subroutine name
                              2. array variables
                              A reference is a scalar variable whose value is a pointer to another Perl variable. Use it to build more complex data types. Make reference with \. Stringize it with ref
Perl references intro
                             mv @arrav = gw(a, b, c):
                                                               mv $arrav ref = ['a' .'b'. "c\n"]:
                                                                                                              mv %hash = (a=>1, b=>2, c=>3):
                                                                                                                                                          my $hash_ref = {a=>1, b=>2, c=>3};
Perl reference tutorial
                                                               print ${$array_ref}[1]; # b
print $array_ref->[1]; # b, arrow notation
                              print $array[1]. # b
                                                                                                                                                          print ${$hash_ref}{c}; #3
print $hash_ref->{c}; #3 with arrow notation
                                                                                                              print $hash{c}; #3
Reference purpose
                                                                                                              ← arrow notation is shorter/cleaner ➡
                              Store a ref to an array or hash into an array: push @array \%hash
                                                                                                              Pass array or hash to subroutine: fct(\@a, \%h): Return from sub: return (\@a, \%h):
Scalar values
                                                                   \label{literals} \textbf{literals} \ \textbf{examples:} \quad \textbf{Note: leading 0 work only for literals, not for string-to-number conversions.}
                                                                                                                                                                              Useful related builtin functions
                                                                                          my $x = 12345;
my $x = 12345.67;
                                                                                                                             # integer
# floating point
                                integer: using the system's native format.
· numeric:

    oct - supports binary, octal,

                                   bigint - transparent big integer support
                                                                                                                                                                                 hex
Note: underlines can be
                                                                                                                             # scientific notation
# power² exponent: Perl >= v5.22
                                                                                                                                                                                 hex
POSIX::ceil
                                   bignum - transparent big number support.
                                                                                              x = 6.02e23;
used inside decimal,
                                floating-point: using the system's native format.
                                                                                                      0x1f.0p3;
                                                                                          my
                                                                                              $x
                                                                                          my $x = 4_294_967_296;
my $x = 0x1234_5678;
                                                                                                                               underline for legibility underline in hex is also OK
hexadecimal and binary
                                · bigrat - transparent big rational number support.
                                                                                                                             #
                                                                                                                                                                                 POSIX::floor
                                                                                                                                                                              • <u>abs</u>
literals.
                             A variable holding an integer can be converted to floating-point if the operation done to it requires it
                                                                                          my
                                                                                              5x = 0377
                                                                                                                             # octal
                                                                                                      00377;
                                                                                                                                octal also
                                                                                               $x
                                                                                          my
                                                                                          my $x = 0b1100_0010;
                                                                                                                             # binary with underlines
                              (such as dividing 1 by 2).

    string

                                double-quoted strings: perform backslash and variable interpolation of expression that begin with $ (a scalar) or @ (an array). Hashes cannot be interpolated.
                                single-quote strings: only perform \' and \' substitution (to ' and \' respectively), nothing else.
                                Single quote and double quote strings can spread multiple lines: it embeds the newline character on each new line.

    But \n is only expanded in double quoted strings! In single quote string it is treated as two characters; no substitution is done (as explained above).

  • Unicode support
                             Use Unicode literally in a program; add the utf8 pragma: use utf8;
                                                                                                              See: Perl Unicode Tutorial, Perl Unicode Introduction, Perl Unicode Support @ perldoc
   · Quote constructs
                                           Generic
                                                                Meaning
                                                                                          Interpolates?
                                                                                                              Notes
                                                                Literal string
                                                                                                              • Not all characters can be used as the / separator. { }, ( ) and < > can also be
                                                                                          No
                                           q//
                                                                Literal string
                                           qq//

    Strings in Perl:

                                                                                          Yes
                                           qx//
                                                                Command execution
                                                                                                                 You can use whitespace between the quote specifier and its initial bracketing character:
           quoted,
                                                                                          Yes
           interpolated
                                          qw//
m//
                                                                                                                       my $chuck_of_code = q {
    if ($condition) {
                             ()
//
                                                                World list
                                                                                          No
                                                                Pattern match
                                                                                          Yes
           and escaped
                                                                                                                                 print "Salut!
                              s///
                                           s///
                                                                Pattern substitution
                                                                                          Yes
                             tr///
                                                                Character translation
                                           ar//
                                                                Regular expression
                                                                                          Yes
                              • It's also possible to write: s<foo>(bar) and tr(a-f)[A-F] as well as separating them on 2 lines:

    Array variables are interpolated by joining all elements with the separator specified by the <u>$" special variable ($LIST_SEPARATOR)</u>.

                                                                                                                                                                                                     [A-F1:
     Character escapes
                                           Alert (bell)
                                                                                                              FSC character
                                                                                                                                                    Any Unicode code point, by name
                              \b
                                           Backspace
                                                                                   \033
                                                                                                              ESC in octal
                                                                                  \o{33}
\x7f
                                                                                                                                                    \N{LATIN SMALL LETTER E WITH ACUTE}
\N{ U+E9 }
      double auoted
                              \e
                                           ESC character
                                                                                                              ESC in octal
                              \f
                                           Form feed
                                                                                                              DEL in hexadecimal
      strings)
                             \n
                                           Newline (usually LF)
                                                                                   \x{263a}
                                                                                                              Character number 0x263A
                                                                                                              Control-C
                                           Carriage return (Usually CR)
                              ١t
                                           Horizontal tab

    translation

                                           Force next character to titlecase
                                                                                   \U
                                                                                          Force all following characters to uppercase. Ends at \E
                                                                                                                                                                              ١E
                                                                                                                                                                                           Fnds \U \L \F or \Q
                              ۱u
                                                                                          Force all following characters to lowercase. Ends at \E
      escapes
                                           Force next character to lowercase
                                                                                   \L
 (inside double auoted
                                                                                   ۱F
                                                                                          Force all following characters to Unicode fold case. Ends at \E
                                                                                   \Q
                                                                                          Backslash all following non alphanumeric characters. Ends at \E
      strings)
  • bareword
                              In Perl, a bareword refers to a sequence of characters suitable for an identifier. It's not quoted. By default Perl allows barewords to behave like strings.
                               This is not allowed when any of use strict; or use strict "subs"; or use v5.12; is specified.
                              Perl here-documents are a form of line oriented quoting. There are several forms of here documents, where the identifier (like EOF used below, but can be any word)

    Here documents

                             must be placed at the beginning of the terminating line:
        Here docs @ Perl
        maven
Perl here doc
                                Default:
                                                  << FOF
                                                                  Supports variable interpolation
                                Double quotes: <<"EOF";
                                                                   Supports variable interpolation. Can also be written with whitespace as in << "EOF
                                                                   Does not support interpolation. Can also be written with whitespace as in << 'EOF';
Execute commands in a shell and return text printed on stdout. Can also be written with whitespace as in << 'EOF';
        @Wikipedia
                                Single quotes:
                                                   <<'EOF':
                                                   <<`EOF`;
                                backticks:
                                                   <<~FOF:
                                                                   Allows indenting the here-doc string. Can also use the ~ with the other forms: <<~\EOF, <<~"EOF", <<~"EOF", <<~"EOF"
                                indented:
                                They can also be stacked and text can be transformed. See the documentation.
   • Perl Regexp
                              Regexp Tutorial, Learn PCRE in X minutes, PCRE cheatsheet,
                                                                                                                    Debuggex regexp tester, regex101, RegEx Pal
                                                             $last_slash = <u>rindex("/usr/bin/ls", "/");</u>
                                                                                                              part = substr(stext, pos, len) \mid A value of -1 in pos identifies last character.
   · index/substr
                              $pos = index($page, $line):
                              mv $pref = "I like awk and erlang"
                                                                                                              substr($pref. -15) =~ s/Perl/Perl5/q: # replace text inside a restricted portion of the string.

    Replacement
```

substr(\$pref, index(\$pref, "awk"), length("awk")) = "Perl"; substr(\$pref, 0, 0) = "Sally and"; # insert text anywhere

# insert text anywhere

with substr LPo

Page 12   Page							
Contract securities   SANG				e use the <b>peridoc -v</b> command.			
Security and protein in Security 2015 - S. S. Security 2015 -		\$# \$* \$[ \${^E	ENCODING} \${^WIN32 SLOPP	PY_STAT}			
Sendor Services  Selection and Selection	General variables						
AND THE CONTROL OF TH				subroutine parameters			
Accessed and services and servi	list separator				• \$SUBSEP		
STOP STOP STOP STOP STOP STOP STOP STOP	·	_				NAME	
Special variables in each of the provision of function uses good variables for and control of the provision of function uses good variables for and control of the provision of function uses good variables for and control of the provision of function uses good variables for and control of the provision of function uses good variables and so and control of the provision of the pr	Perl process ID	• \$PID	Process real GID	• \$GID	Process effective GID	D • \$EGID	
Service devicements   Service   Serv	Process real UID	• \$UIG		Process effective UID	• \$EUID	ER_ID\$	
- See Port Interprete medicals, and all intervience of the control	Special variables in sort				on that uses the <=> equ	ality operator to force numerical	
version and subversion   syl   some subversion   syl	<u>Current environment</u>	%ENV				ays.	
## SPECUAD DIRECTORS					_	1	
Include-edit extension state extension state extension state in STA	Maximum file descriptor				<b>@</b> F		
Section   Sect	Include Directories	@INC	Included filenames	%INC	Hook localization (?)	\$INC	
Second Mark				@ISA		\$^M	
Pet sex built Regest Potriables String matched Regest Potriables String matched String matched String preceding match string preceding preceding match preceding precedin	Maximum block nesting	\${^MAX_NESTED_EVAL	_BEGIN_BLOCKS}			T-11-0-11-11-1	
Scring matched  String pescading match  String pescading match (completed (APOSTMATCH) pescading match (completed (APOSTMATCH) pescading pescading match (completed (String pescading pescadin		1	Signal handlers	%SIG		%{^HOOK}	
Since matched  String percenting match String percenting perc	Regexp Variables						
String preceding match  String preceding match  Spread of the spread of	captured sub-patterns	\$ <digit>(\$1,\$2,)</digit>		Capture buffer content	@{^CAPTURE}		
String following match  String following match  Spostmatch  Spostmatch  Spostmatch  String following match foompiled  Spostmatch  Spostmatch  Spostmatch  String following match foompiled  String following match foompiled  Spostmatch  Spottmatch  Spostmatch  Spottmatch  Spostmatch	String matched	l '			\${^MATCH}		
Last Capture group  SLAST_PAREN_MATCH String Fragment (Compute Rev) SLAST_PAREN_MATCH String Fragment (Compute Rev) SLAST_PAREN_MATCH String Fragment (Compute Rev) String Fragment (Compu	String preceding match				\${^PREMATCH}		
Match capture key values	String following match				{^POSTMATCH}		
## Watch start offsets  ## WELAST_MATCH_START	Last capture group		H		_	CH_RESULT	
Last successful pattern  \$\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \		%LAST_PAREN_MATC	Н	Maximum regexp nested group	\${^RE_COMPILE_RECURSION_LIMIT}		
regex debug flag  \${ARE_DEBUG_FLAG}\$  regex internal optimization/memorx  \${ARE_TRIE_MAXBUF}\$  **Format_Variables  **Current_value of the writel accumulator for formatil_lines.  **Form_tead_format_commat.  **Great_TRIE_MAXBUF}  **SACCUMULATOR  **SACCUMULES-format_lines_left(EXPR)  **SACMAT_LINE_BREAK_CHARACTERS  **SACCMAT_LINE_BREAK_CHARACTERS  **SEOMMAT_LINE_Seromat_lines_per_page(EXPR)  **SACMAT_LINE_Seromat_lines_per_page(EXPR)  **SACCMMAT_LINES_per_page  **SACMAT_LINES_per_page  **SACMAT_LINES_per_page  **SACMAT_LINES_per_page  **SACCMMAT_LINES_per_page  **SACCMMAT_LINES_p	Match start offsets		Match ends offsets				
• Format Variables  Current value of the writed accumulator for format Unions.  Form feed format. defaults to M  • SPORMAT_FORMFEED  • SPORMAT_FORMFEED  • SPORMAT_FORMFEED  • SPORMAT_LINES_LIFT  • S	Last successful pattern	\${^LAST_SUCESSFUL_PA	ATTERN}			_CODE_RESULT	
Current value of the write) accumulator for tormat() lines.  Form feed format. defaults to M	regexp debug flag	\${^RE_DEBUG_FLAG}		regexp internal optimization/mem	ory \${^RE_TRIE_N	MAXBUF}	
writed accumulator for formatd, lines. Format_forms(EXPR)  Form feed format, defaults to M  Number of lines left on the page on currently selected output channel  Name of current top-page format of output channel  Form Variables  The variables \$\%, \$\\$, \$\\$, \$\\$, and \$\\$? contain information about different types of error conditions that may appear during execution of a Perl program. They correspond to errors detected by the Perl interpreter, C library, operating system, or an external program, respectively.  Selected error  OS detected error  Variables  Variables  OS detected error  Selected format.  Ino: Handle->format_line_break_characters EXPR string may be broken to fill continuation fields  Current page length of current output channel  Variables  HANDLE->format_lines_per_page(EXPR)  SFORMAT_LINES_PER_PAGE  SFORMAT_LINES_PER_PAG	Format Variables						
SFORMAT_FORMFEED   String may be broken to fill   Continuation fields   SFORMAT_LINE_BREAK_CHARACTERS   String may be broken to fill   Continuation fields   SFORMAT_LINE_BREAK_CHARACTERS   String may be broken to fill   String format in the string form and to current output output output channel   SFORMAT_LINE_Sper_page (EXPR)   SFORMAT_LINE	write() accumulator for						
the page on currently selected output channel  SFORMAT_LINES_LEFT  SHANDLE->format_top_name(EXPR)  SFORMAT_TOP_NAME  SFORMAT_TOP_NAME  SFORMAT_TOP_NAME  SFORMAT_TOP_NAME  SFORMAT_NAME  SFORMAT_LINES_PER_PAGE  SFORMAT_LINES_P	_	<ul> <li>\$FORMAT_FORMFEED</li> </ul>		string may be broken to fill	• \$FORMAT_LINE		
- \$FORMAT_TOP_NAME - \$^ - \$FORMAT_NAME - \$^ - \$^ - \$FORMAT_NAME - \$^ - \$^ - \$FORMAT_NAME - \$^ - \$^ - \$Contain information about different types of error conditions that may appear during execution of a Perl program.  They correspond to errors detected by the Perl interpreter, C library, operating system, or an external program, respectively.  Perl error from the last eval operator - \$EVAL_ERROR - \$@ - \$Current value of C error integer variable - \$OS_ERROR - \$! returns the system variable error when used in a numeric context, but returns the string from perror() when used in string context.  OS detected error - \$EXTENDED_OS_ERROR - \$^! - \$CHILD_ERROR - \$^ - \$CHILD_ERROR - \$^ - \$CHILD_ERROR_NATIVE} - \$^ - \$CHILD_ERROR_NATIVE} - \$^ - \$CHILD_ERROR_NATIVE} - \$^ - \$^ - \$CHILD_ERROR_NATIVE} - \$^ - \$^ - \$^ - \$CHILD_ERROR_NATIVE} - \$^ - \$^ - \$^ - \$^ - \$CHILD_ERROR_NATIVE} - \$^ - \$^ - \$^ - \$^ - \$^ - \$^ - \$^ - \$^	the page on currently	• \$FORMAT_LINES_LEF			• \$FORMAT_LINES_PER_PAGE		
* Error Variables  The variables \$@, \$1, \$^E, and \$? contain information about different types of error conditions that may appear during execution of a Perl program. They correspond to errors detected by the Perl interpreter, C library, operating system, or an external program, respectively.  Perl error from the last eval operator  * \$EVAL_ERROR  * \$0 \$EXCEPTIONS_BEING_CAUGHT  * \$^S\$  Current value of C error on the last eval operator  * \$0 \$ERROR  * \$1 returns the system variable error when used in a numeric context, but returns the string from perror() when used in string context.  * \$EXTENDED_OS_ERROR  * \$CHILD_ERROR  * \$CHILD_ERROR_NATIVE}	page format of output	\$FORMAT_TOP_NAME		· ·	HANDLE->format_name(EXPR)     \$FORMAT_NAME		
Perl error from the last eval operator  • \$EVAL_ERROR • \$@  • \$OS_ERROR • \$! returns the system variable erroo when used in a numeric context, but returns the string from perror() when used in string context.  • \$EXCEPTIONS_BEING_CAUGHT • \$AS  • \$CUrrent value of C erroo when used in a numeric context, but returns the string from perror() when used in string context.  • \$CUrrent state of interpreter • \$EXCEPTIONS_BEING_CAUGHT • \$AS  • \$COS_ERROR • \$CERROR • \$CER	Error Variables	The variables \$@, \$!, \$^E, and \$? contain information about different types of error conditions that may appear during execution of a Perl program.					
integer variable  • \$ERRNO • \$! when used in a numeric context, but returns the string from perror() when used in string context.  • \$ERRNO • \$! set to 1 if current error is this error.  • \$ERRNO • \$!  • \$ERRNO • \$!		• \$EVAL_ERROR	2011		• \$EXCEPTIONS_E	BEING_CAUGHT	
Status returned by last pipe close, backtick command, wait, waited.      SEXTENDED_OS_ERROR      *SCHILD_ERROR      native status returned by last pipe close, backtick command, wait, waited.      *SCHILD_ERROR_NATIVE}      *SCHILD_ERROR_NATIVE}		• \$OS_ERROR • \$ERRNO	when used in a numeric context, but returns the string from <b>perror()</b> when	set to 1 if current error is this	• %OS_ERROR • %ERRNO		
Status returned by last pipe close, backtick command, wait, waited,  * \$CHILD_ERROR  native status returned by last pipe close, backtick command, wait, waited,  * \$?  * \$CHILD_ERROR_NATIVE}  pipe close, backtick command, wait() or system() call	OS detected error						
	pipe close, backtick command, wait, waited,	• \$CHILD_ERROR		pipe close, backtick command,	\${^CHILD_ERROR_	_NATIVE}	

Current value of warning switch	• \$WARNING • \$^W			Current set of warning checks enabled by the use warnings pragma	\${^WARNING_BITS	}
Variables related to the interpreter state	These variables provide inform	ation about the cu	urrent interpreter state.			
Flag associated with the -c switch	• \$COMPILING • \$^C			The current value of the debugging flags	• \$DEBUGGING • \$^D	
Current phase of the perl interpreter	\${^GLOBAL_PHASE}			Debugging support. Internal variable.	• \$PERLDB • \$^P	
Compile-time hints for the perl interpreter. Internal use only	\$^H			Values of compiled statements	%^H	
Taint mode	\${^TAINT}			Safe locale operations availability	\${^SAFE_LOCALES	3}
Input/Output Layers. Internal use by PerllO only.	\${^OPEN}			Unicode Settings of Perl	\${^UNICODE}	
Internal UTF-8 offset caching code state	\${^UTF8CACHE}			State of UTF-8 locale detected by perl at startup.	\${^UTF8LOCALE}	
File handle Variables	See also: Perl File Handles		The following variables a	are used in the Input/Output handling as well as program arguments.		
Name of current file read from <>	\$ARGV		rguments of the script nd operator <>.  ➡	@ARGV	Number of arguments minus one	\$#ARGV
Special file handle that iterates over command-line filenames in @ARGV	ARGV	Special file hand currently open o edit-in-place pro	utput file when doing	ARGVOUT		
Output field separator for the print operator	<ul> <li>IO::Handle-&gt;output_field_separator( EXPR )</li> <li>\$OUTPUT_FIELD_SEPARATOR</li> <li>\$OFS</li> <li>\$,</li> </ul>			Current line number for the last file handled accessed	<ul><li>HANDLE-&gt;input_</li><li>\$INPUT_LINE_N</li><li>\$NR</li><li>\$.</li></ul>	
Input record separator (newline by default)	<ul> <li>IO::Handle-&gt;input_record_separator( EXPR )</li> <li>\$INPUT_RECORD_SEPARATOR</li> <li>\$RS</li> <li>\$/</li> </ul>			Output record separator	• IO::Handle->outpu • \$OUTPUT_RECO • \$ORS • \$\	t_record_separator( EXPR ) RD_SEPARATOR
Auto-flush control  order of output @ Perl Maven  Suffering from Buffering?	• HANDLE->autoflush( EX • \$OUTPUT_AUTOFLUSH • \$I		Perl activates file buffering by default. Assign 1 to \$  to activate auto-flush.	Last read file handle	\${^LAST_FH}	

# Perl 5 Input/Output

References	Writing to	en @ perldoc browser  • Stupid open() tricks @Perl.com:  • print to a string  • read lines from a string  en file in-memory @ stackOverflow  • create an anonymous temporary file							
print, printf, sprintf		print, printf, sprintf (which describes the format). Note: print, a list operator, is more efficient than printf.  print output to stdout by default, but accept a file handle as the first argument if it is NOT followed by a separating comma! (a ',' puts it in the list to print!)							
diamond operator <>		oth <> and <<>> operators read the content of files listed on the command line via @ARGV. Nothing or - on the command line identifies stdin.  ne <> operator supports shell redirection and pipe operations which <<>> does not allow (for security reasons).							
The double diamond, a more secure <> (Perl >=	print <>	·;	← Simple implementat	ion of /bin/cat	print <<>>;	← safer one	Redirection cannot be forced via file names embedding them		
v5.22)	print so	ort <>;	← Simple implementat	ion of /bin/sort	<pre>print sort &lt;&lt;&gt;&gt;;</pre>	← safer one	with. the <<>> operator.		
In-place-editing of The <> operator tries to duplicate the original file's permission and ownership.	change the In a while renames opens a r prints into	\$^I to a backup file extension (such as Emacs "~" or ".bak") to nge the behaviour of the <> and <<>> operators and print.  while (<>) {} loop, when \$^I is not undef (its default), Perl: mannes currently processed file with the specified extension added, ones a new file with the original name rints into the new file.  ny modification goes into the new file: in-place-editing it!  use strict;  \$^I = "~"; # rename old file: add '~' to it's name (Emacs-style backup)  while (<>) {     s/something/Something else/; # perform any substitution     print; }							
perl -i cmdline option	It's also pos	's also possible to do this on the command line! For example: perl _p _i ~ _w _e 's/something/Something else/g' data*.dat							
Special filehandle names	ARGV	The special filehan	dle that iterates over co	mmand-line filenar	nes in @ARGV. Usually writt	en as the null filehandle in the	angle operator <> (or <<>>)		
Also See: • File handle Variables	ARGVOUT	The special filehandle that points to the currently open output file when doing edit-in-place processing with <u>-i</u> .  • Useful when you have to do a lot of inserting and don't want to keep modifying \$							
section above.	STDIN	Each time <std inc<="" string="" th="" the=""><th colspan="5">**STDIN&gt;: line input operator for the STDIN filehandle (for the standard input).  Each time <stdin> is used in scalar context, Perl reads 1 complete line of the standard input and uses it as the value of <stdin>.  The string includes a line termination character. Use the chomp built-in function to strip it off the variable.  If <stdin> is read in list context, it returns all lines inside a list! For example, foreach (<stdin>) { } reads the entire stdin in 1 step: \$_ holds it all!</stdin></stdin></stdin></stdin></th></std>	**STDIN>: line input operator for the STDIN filehandle (for the standard input).  Each time <stdin> is used in scalar context, Perl reads 1 complete line of the standard input and uses it as the value of <stdin>.  The string includes a line termination character. Use the chomp built-in function to strip it off the variable.  If <stdin> is read in list context, it returns all lines inside a list! For example, foreach (<stdin>) { } reads the entire stdin in 1 step: \$_ holds it all!</stdin></stdin></stdin></stdin>						
		<pre>while (<stdin>     print; }</stdin></pre>	<pre># print all # lines of # stdin</pre>	<pre>while (define    print \$_; }</pre>	ed(\$_ = <stdin>)) {</stdin>	equivalent to the cod stored in the default	nost cell is the shortest form. It is le beside it; each line of stdin is variable \$_ and the loop stops on STDIN> returns undef.		
	STDOUT	standard output							
	STDERR	Note: generally STDERR is not buffered, while STDOUT is buffered by default. Text sent on STDERR may show up before STDOUT.  • Print a new line on STDOUT to help flushing it or assign 1 to \$   to activate auto-flush.							
	DATA								
say	• say	use fea	ture qw(say);	or use v5.	10; (or higher). Like	print, but implicitly appends a	newline at the end of the list.		
<u>open</u>									

## Perl 5 Statements

Loop control	See <b>perlsyn</b> for more informati	ee perlsyn for more information on Perl syntax which includes declarations, blocks, loops, labels, subroutines, etc					
Use the <u>last</u> and <u>redo</u> inside a naked block of code to control looping.	loop control keywords:   last of exits the loop.   next of starts the next iteration of the loop.   redo of restarts the loop block without evaluating the condition again.		The last, next, and redo loop control keywords work in the following constructs:  • while (condition) { }  • until (condition) { }  • for (init; condition; continue) { }  • foreach array { }  • naked block: { }	Notes:  • The while and foreach loops may have a continue block: executed before evaluating condition again, which corresponds to the 3rd part of a for loop statement. See this @ stackOverflow.  • Blocks can be labelled of as targets to last, next, and redo			
Statement modifiers	• if EXPR • unless EXPR • while EXPR • until EXPR • for LIST • foreach LIST • when EXPR • do block	processed. The has "END" forea	each statements impose a list context; the complete list is refore a loop like the following trying to stop on a line that on it will not work since it reads all of STDIN:  ch ( <stdin>) {  if ?END/;</stdin>	The while statement imposes a scalar context; it takes one line at a time from <stdin> and the following code works properly:  while (<stdin>) {     last if /_END/;    ;   }</stdin></stdin>			
Conditional statements							

### Perl 5 Subroutines

Perl subroutines								
subroutine &		teach the subroutine		ne? @ StackOverflow	Another point of view: Subroutine	es and Ampersands		
Subroutine Prototypes	An older Po	erl feature. Clashes v	vith subroutine si	gnatures as of Perl v5.20	). In $Perl >= v5.20$ put the :protot	ype attribute before sub	proutine prototype	parenthesis.
Subroutine signatures	Exactly zer	o arguments		()	Zero or 1 argument, no default, ur	nnamed:	(\$=)	
<ul><li>Perl &gt;=5.36: Stable</li><li>Perl &gt;= 5.20:</li></ul>	Zero or 1 a	rgument, no default,	named	(\$val=)	Zero or 1 argument, named, with	default	(\$val=1)	
Experimental See: <u>Use v5.20</u> subroutine signatures	exactly 1 n	amed argument:		(\$val)	Exactly 2 arguments		(\$v1, \$v2)	
	2, 3 or 4 arguments no defaults: (\$v1,		\$v2, \$=, \$=)	2,3 or 4 arguments, 1 default:		(\$v1, \$v2, \$ \$=)	\$v3='a',	
	Two or more, any number of arguments.		(\$v1, \$v2, @)	Two or more arguments, remainders into a named array:		(\$v1, \$v2, (	Prest)	
	Two or more arguments: an even number		(\$v1, \$v2, %)	Two or more arguments, remainders into a named hash:		(\$v1, \$v2,	rest)	
	Class met	<u>hod</u>		(\$class,)	Object method		( \$self,)	
Variables in subroutines	global by	default						
	<u>my</u>	local, lexical scope	, non persistent					
	<u>state</u>	Local, lexical scop	e, persistent	Perl >= v5.10	Restriction: in Perl < v5.28: array a	and hashes state cannot	be initialized in list	context.
	our	creates a lexical so	oped alias to a p	ackage variable				
	local Localizes an existing package variable to the current scope. It's not a declaration. The variable previous value is restored when leaving the scope.							
Returned value	The retu     The subi	<ul> <li>The result of the last evaluated expression is implicitly returned</li> <li>The return operator can be used but it's not required unless used to change execution flow (return immediately from the subroutine).</li> <li>The subroutine can return a scalar in scalar context or a list if called in list context.</li> <li>Inside the subroutine, use the <u>wantarray</u> function to determine the context of the subroutine call.</li> </ul>						

## Perl 5 Built-in Functions

Perl Functions Perl syntax	To get information about a Perl function from the command line use the <b>perldoc -f</b> command.  To get information about <b>print</b> use: <b>perldoc -f print</b>
!Cautionary notes	
each keyword is broken     Use Var::Pairs instead.	Do NOT use the built-in each. It is broken, as described by <u>Damian Conway</u> in his <u>Modern Perl Best Practice O'Reilly course</u> , section control structure.  • each is not re-entrant:  • nested loops of each over the same hash does not work as expected and will create infinite loop since the nested loop each juts iterates from where the first loop each left it.  • Exiting the loop leaves the state of the each internal pointer at the current location.  • If you use each on the same hash later it will resume from where it left, it will not start form the beginning.

	Perl 5 Modules ###					
Perl Modules						
Perl core modules	• How to detect where a module is installed : perldoc -1 Module • How to check if a module is part of Perl core : corelist Module (Perl >= v5.9.2)					
Modules @perltutorial  Modules  Using simple modules o	Looks for the module file by searching the <pre>@INC</pre> path. Performed at run time (and therefore can be done conditionally).  • If Perl finds the file, it places the code inside the calling program and executes it. Otherwise, Perl will skip the do statement silently.  • The "included" code does not have access to the lexical variables from the main program.  • Skip the @INC path lookup if given a file path starting with ./,/, or /					
	Loads the module file once, also searching the <a href="Milestance">QINC</a> path. Performed at run time (and therefore can be done conditionally).  • If the require for the same file appears twice, Perl ignores it. Perl will issue an error message if it cannot find the file (as opposed to <a href="Molestance">do</a> ).  • Skip the <a href="Milestance">QINC</a> path lookup if given a file path starting with ./,/, or /					
The normal way to access Perl modules ➡	Similar to require except that Perl applies it before the program starts: it's done at compile time. Modify it dynamically in a BEGIN block. See IntPo.  • Therefore the use statement cannot be invoked inside conditional statements such as if-else. Used often to include a module in a program. That imports the defaults as defined by the module's code.  Select what to import with one of the two equivalent forms: (See IntPo):  • use Module::Name ('function_a', 'function_b');  • use Module::Name (); # import nothing. All accesses to the module must be done with Module::Name::something					
Error handling for:  Can't locate in @INC  How to fix that  See Also: IntPo  See: show-perl-inc @ USRHOME	For the above statements to work Perl must be able to identify the location of the requested module(s).  Perl looks for a module code inside the directories identified by the <u>@INC</u> array.  if you have. <u>use</u> The::Module; inside your code, Perl looks for a sub-directory named 'The' containing a file named 'Module.pm' inside each <u>@INC</u> directory.  If Perl does not find it, there are <u>multiple ways to solve the problem</u> :  Add the required directory to the list of directories identified in the ':' separated list in the PERL5LIB environment variable. (use ';' as separators in Windows).  Add a use <u>lib</u> 'path/to/the/directory'; statement inside your Perl file to add the required directory when executing a specific piece of Perl code, at compile time.  Run Perl with the <u>-I (capital i) option</u> to run the code with the extra directory added to <u>@INC</u> array.  To List the directories used by Perl from one of the following equivalent command lines:  * perl -e 'print join("\n", @INC), "\n";					
	• perl -le 'print for INC';'  You can also get more information with perl -V					

## Topic: Directory Operations

		Topic. D	rectory Operations (m)		
<b>Directory Operations</b>	In Books: LPo				
Opening Files	All file open operations are relative to the <u>current working</u> relative file names)	ng directory (for	open my \$filehandle, '<:utf8', 'a_relative/path.txt'		
Creating temporary files	File::Temp (Perl >= v5.6.1). <u>Using File::Temp</u> • Also see <u>IO::File</u>				
Built-in Functions	Related Functions/Packages / Descriptions		Notes		
Getting file names by: Globbing: with glob	File::Glob (Perl >= v5.6.0) - provides more control.	Example:	<pre>my @all_files = glob '*'; my @perl_files = glob '*.pm *.pl'; # 2 globs, space-separated</pre>		
with the glob operator <>	The <> operator is identifying:  • a filehandle, when: the item inside <> is a Perl identifier or an indirect file handle read scalar,  • a glob expression otherwise.	Glob examples:	<pre>my @all_files = &lt;'*'&gt;; my @all_files = &lt;*&gt;; # 1 glob: no space, no need for string my @perl_files = &lt;'*.pm *.pl'&gt;; # 2 globs, space-separated</pre>		
	• a glob expression officiwise.		<pre>my \$etc_dir = '/etc'; my @etc_dir_files = &lt;\$etc_dir/* \$etc_dir/.*&gt;;</pre>		
			<pre>my @files = <larry *="">; # a glob</larry></pre>		
	See: readline	Filehandle	<pre>my @his_lines = <larry>;  # a filehandle read</larry></pre>		
		examples:	<pre>my \$name = 'LARRY'; my @his_lines = &lt;\$name&gt;; # indirect filehandle read of LARRY handle my @same_lines = readline LARRY; # another way to write above my @same_lines = readline \$name;</pre>		
<ul> <li>with a directory handle <u>LPo</u></li> </ul>	opendir: open a directory: get a directory handle     readdir: read the directory handle. But see this.     closedir: close the directory handle.     DirHandle (Perl <= 5.5)     File::Spec::Functions (Perl >= v5.5.4)     Path::Class	Example: iterate explicitly over a list of file names extracted from the directory using these 3 functions.	<pre>my \$dir = '/usr/bin'; opendir my \$dh, \$dir or die "Failed opening \$dir: \$!"; foreach \$file (readdir \$dh) {    print "File \$file is inside \$dir\n"; # A no path in name! } closedir \$dh;</pre>		
Creating directory	• mkdir	Example:	<pre>mkdir \$dir_name, oct(\$permissions); # octal for permissions mkdir \$dir_name, 0700; # do not use "0700", it's 700 decimal!</pre>		
Removing directory	rmdir Removes an empty directory.     File::Path remove tree , rmtree remove dir & files (Files remove dir & files (Files remove dir & files (Files remove dir & files remove dir & files (Files remove dir & files remov	Perl >= v5.0.1)			
Removing files	• unlink a list or \$_		<pre>unlink 'file1.txt', 'file2.txt'; unlink qw( file1.txt file2.txt); unlink glob 'file?.txt'</pre>		
Renaming files	Tename an old file name to a new one. The fat comma operator is sometimes used to highlight what is the old and the new name.	As in here:	<pre>rename 'old_name' , 'new_name'; rename old_name =&gt; new_name; # using fat comma (which quotes)</pre>		
Changing permissions	chmod changes file permissions				
Changing ownership	chown changes file ownership				
Creating <u>Hard link</u>	link to create a hard link				
Creating symbolic link	symlink to create a symbolic link				
chdir Change current working directory	Change the current working directory.     Change to user home directory using the \$ENV{HOME} and \$ENV{LOGDIR} environment values if ⚠ they are set. The File::HomeDir module helps in setting them.     The built-in chdir is global ⚠ for the entire program. Use File::chdir facilities for localized operations.				
Modules	Functions Legend: Exported by default, exported on request, W	lin32 specific	Extra Information		
Cwd	• getcwd, cwd, fastcwd, fastgetcwd, getdcwd • abs path, realpath, fast abs path  use Cwd;  my \$curdir = getcwd;  print "cwd is \$curdir\n";				
File::Basename	fileparse, basename, dirname,				
File::SPec File::Spec::Functions	functional interface to methods: canonpath, catdir, splitpath, splitdir, catpath, abs2rel, rel2abs. All can be		otdir, updir, no upwards, file name is absolute, path. devnul, tmpdir, case tolerant, ng the : ALL tag.		

## Topic: List Operations

			Topic. List Operations w				
List Operators							
Sorting lists	sort	Sort a list	<pre>my @sorted = sort @unsorted_list;</pre>	in place: my @data = sort @data;			
	reverse	Sort a list in reverse order	<pre>my @rsorted = <u>reverse</u> @unsorted_list;</pre>	in place: my @data = <u>reverse</u> @data;			
Filtering list with grep	my @adult_	_ages = <b>grep</b> \$_ > 18, @ages;	my @lucky_ages = grep /7\$/, @ages; # all that end with 7	my @read_ages = <b>grep</b> { \$_ >= 7 && \$_ <= 77 } @ages;			
Counting matches	my \$count	= <b>grep</b> \$_ > 18, @ages;					
		An expression, subroutine or block with trailing boolean can be used as the grep criteria. Each item in the list is identified inside grep by \$					
Transform a list with map							

# Topic: Process control

Process Control	In Books: <u>LPo</u>	Important se	ecurity information: peridoc perisec		
<b>Environment Variables</b>	Inside the <u>%ENV</u> hash.		erl %Config hash: Perl configuration information. For example, whether it support threads, what are path separators, etc  To use it: use Config;		
<b>Built-in Functions</b>	Example		Description/ Notes		
system (2 functions)			Run child process asynchronously using parent's stdin, stdout and stderr, using the OS native command shell.		
<ul><li>using the shell</li><li>security risk?</li></ul>	<pre>system "cd \$project;</pre>	make &";	Use the Unix shell to execute a long running build asynchronously. However: avoid using the shell like this.  • Using the shell to build commands from unvalidated user input data may lead to security issues.		
avoiding the shell	system 'tar', 'cvf', \$tarfile, @directories;		No shell invoked when more than 1 argument is passed to system. No shell interpretation, piping, re-direction done.		
other syntax	system( 'tar', @arguments);		O means success: unless (system 'tar', arguments) { print "tar command success\n"; }		
	<b>system</b> ( { \$prog }, \$arg0, @a	args);			

	Note that if the string contain <b>no</b> shell metacharacters it is executed directly (not through a shell).					
<ul> <li>system return value:</li> <li>A value of 0 usually means all was OK.</li> </ul>	2 bytes: MSByt	MSByte: child program exit code.		my \$retval = system( );		
	informa • 0x80	SByte: system-specific nformation bits: 0x80: set on core dump. 0x7f: signal number		<pre>my \$childp_exitcode = \$retval &gt;&gt; 8; my \$had_core_dump = (\$retval &amp; 0x80) = my signal_number = \$retval &amp; 0x7f;</pre>	<ul> <li>← shift most significant byte</li> <li>= 0x80? 1 : 0;</li> <li>← use least significant byte</li> </ul>	
exec	Unlike system, exec does not return to the parent Perl process. Use: exec 'the_program' or die 'Could not run: \$!"; #or warn or exit					
backquotes``	Use backquotes to <b>capture the stdout</b> of a program. That's the main point of using it.  • The trailing newline is not filtered out; it can be filter by <b>chomp</b> .					
	<ul> <li>The value inside the backquotes is treated like the single double quote string argument of system: it will invoke the shell if there are any shell meta-characters and supports interpolation.</li> <li>The following example builds a dictionary (hash) of topics with the text extracted from perldoc.</li> <li>Note that `` is also written as qx/ /</li> <li>backquote operation in scalar context returns 1 string. In list context it returns a list of strings (1 per line).</li> </ul>					
Modules						
Capture streams	Can be used to capture the stdout and stderr streams for various ways if executing other programs					
Inter-process support	Can also be used to capture streams and provide more inter-process support.     It provides <u>systemx</u> which never uses the shell, along with other useful functions.					
Processes as filehandles	In Books: LPo					
Perl ← program	Launching a process that		open DATE, 'dat	te  ' or die "Cannot pipe from date: \$!";	Use a bare word to define the DATE file handle.	
	pipes into the Perl process		open my \$date_	fh, '- ', 'date' or die "Cannot pipe from date: \$!";	This one and the others define a local file handle variable. The file handle variable can later be used to read, as the	
			open my \$ps_fh	n, '- ', 'ps', 'aux' or die "Cannot pipe from ps: \$!";	above one, but is not global.	
			open my \$find_fh, '- ', 'find', qw(name '*.p[lm]' -print) or die "Cannot pipe from find: \$!";			
Perl ➡ program	Launching a process that the Perl process pipes into.  Open my \$dispather_fh, ' -', 'dispatcher', qw ( '-to-perl-groups' 'Help!' ) or die "Cannot pipe to the dispatcher: \$!";					
Forking	In Books: LPo . See also: Linux fork(2) system call, QA: Why do we need fort to create new processes? Why fork woks the way it does?					
fork with exec and waitpid  See also: Other IPC functions Perl IPC	<ul> <li>fork the process into parent and child.</li> <li>in the child process start the program with exec</li> <li>In the parent process wait for the program termination with waitpid</li> <li>defined(my \$process_id = fork) or die "Fork failed: \$!"; unless (\$process_id) {         # Inside the child process (created by fork)         exec 'long_running_process' or die "Failed starting long_running_process: \$!";         # Inside the parent process, wait for completion of long_running_process.</li> <li>waitpid(\$process_id, 0);</li> </ul>					
<u>Signals</u>	In Books: LPo					
kill	Sends a signal to a list of processes.  • The signal may be identified by number or name (string), which is more portable.  • The *Config{sign name} provides the supported signal names.					
	Note that the fat	comma oper	rator (=>) can be	kill INT => \$pid or die "Can't signal \$pid with SIGINT: \$!";		
	If the signal is 0 of signal to the production.		unless (kill 0, \$process_id) {   warn "Process \$process_id is no longer running!"; }			
	If the signal is a ridentified by the			• <u>kill</u> '-KILL', \$process_group • <u>kill</u> -9, \$process_group		
Signal handlers	• Set the signal handler by setting %SIG for the signal name (with no 'SIG' prefix) to a string holding the name of the subroutine.  \$\frac{\\$SIG}{'INT'}\} = '\text{dispatcher_int_handler';}					

# PerlTidy formatting control

perItidy option	Option	Impact
indentation style	-bl,    opening-brace-on-new-line    brace-left	<ul> <li>Without this option (the default) the code indentation style selected is <u>K&amp;R style</u>.</li> <li>With this option, the indentation style is <u>Allman/BSD style</u>.</li> </ul>