14. 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 sub-expressions matched by the parentheses in pattern, i.e. **($1,$2,$3,**…**)**.

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

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, otherwise it returns **false**.

Optional modifiers: **g** replaces all occurrences of the pattern; **e** evaluates the replacement string as a Perl expression; for the other modifiers, see /PATTERN/ matching. Almost any delimiter may replace the slashes; if single quotes are used, no interpolation is done on the strings between the delimiters, otherwise they 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†]

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

15. File test operators

These unary operators take 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 info of the preceding test or **stat** call.

-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 section 'Special variables'.

16. 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.

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. If SUBROUTINE is specified, gives the name of a subroutine that returns 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]]]
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 on 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.

ioctl FILEHANDLE, FUNCTION, $VAR
performs *ioctl*(2) on the file. This function has non-standard return values. See the manual for details.

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.
"|CMD"	opens a pipe to command CMD. If CMD is '-', forks.
"CMD|"	opens a pipe from command CMD. If CMD is '-', forks.

FILE may be &FILEHND, in which case the new file handle 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†]
Prints the elements of LIST, converting them to strings if needed. If FILEHANDLE is omitted, prints by default to standard output (or to the last selected output channel, see **select**).

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 1 upon success, 0 otherwise.

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 FILEHANDLE is omitted, assumes the file last read.

10. Structure conversion

pack TEMPLATE, LIST
: Packs the values into a binary structure using TEMPLATE.

unpack TEMPLATE, EXPR
: Unpacks the structure EXPR into an array, using TEMPLATE.

TEMPLATE is a sequence of characters as follows:

a	/ **A**	ASCII string, null / space padded
b	/ **B**	Bit string in ascending / descending order
c	/ **C**	Native / unsigned char value
f	/ **d**	Single / double float in native format
h	/ **H**	Hex string, low / high nybble first.
i	/ **I**	Signed / unsigned integer value
l	/ **L**	Signed / unsigned long value
n	/ **N**	Short / long in network (big endian) byte order
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 which will be used as a repeat count, '*****' specifies all remaining arguments.
If the format is preceded with **%**N, **unpack** returns an N-bit checksum instead.
Spaces may be included in the template for readability purposes.

11. String functions

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 lower case version of EXPR.

lcfirst EXPR
: Returns EXPR with the first character in lower case.

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]
: 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 child pid 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 (numeric, or its name as a string).

setpgrp PID, PGRP
: Sets the process group for the PID (0 = current process).

setpriority WHICH, WHO, PRIO
: Sets the current priority for a process, process group, or a user.

sleep [EXPR]
: Causes the script 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 pid of the deceased process (-1 if none). The status is returned in **$?**.

'**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 [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.

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 [LIST]
: Imports semantics from the named module into the current package.

7. 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

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.

23. Miscellaneous

defined EXPR
: Tests whether the lvalue EXPR has an actual value.

do FILENAME
: Executes FILENAME as a Perl script. See also **require** in section 'Subroutines, packages and modules'.

dump [LABEL]
: Immediate core dump. When reincarnated, starts at LABEL.

eval{EXPR**;** …**}**
: Executes the code between **{** and **}**. Traps run-time errors as described with **eval(**EXPR**)**, section 'String functions'.

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

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

ref EXPR†
: 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.

@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.
$var{'a',1,...} emulates a multi-dimensional array.
('a'..'z')[4,7,9] a slice of an array literal.
PKG**::**VAR a variable from a package, e.g. **$pkg::var**, **@pkg::ary**.
****OBJECT reference to an object, e.g. **\$var**, **\%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}**.

4. 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 have an associated assignment operator, e.g. '**.=**'.

-> Dereference operator.
**** 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.
<=> cmp Numeric (string) compare. Returns -1, 0 or 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, xor.

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.
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 side of the operator or at the right side of it. Only the operators **not**, **and**, **or** and **xor**, have lower precedence.
Parentheses can be added around the parameter lists to avoid precedence problems.

25. 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 multi-character.
$, The output field separator for the print operator.
$" The separator which 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 multi-dimensional 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 number of the Perl interpreter running this script. 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 this Perl interpreter 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.

Conventions

fixed	denotes literal text.
THIS	means variable text, i.e. things you must fill in.
THIS†	means that THIS will default to **$_** if omitted.
word	is a keyword, i.e. a word with a special meaning.
RET	denotes pressing a keyboard key.
[…]	denotes an optional part.

1. 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
to enter a single line of script. Multiple **-e** commands may be given to build up a multi-line script.

-F REGEXP
specifies a regular expression to split on if **-a** is in effect.

-i EXT
files processed by the **<>** construct are to be edited in-place.

-I DIR
with **-P**: tells the C preprocessor where to look for include files. The directory is prepended to **@INC**.

-l [OCTNUM]
enables automatic line ending processing, e.g. **-l013**.

-n assumes an input loop around the script. Lines are not printed.

-p assumes an input loop around the script. Lines are printed.

-P runs the C preprocessor on the script before compilation by Perl.

-s interprets '**-xxx**' on the command line as switches and sets the corresponding variables **$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* 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.

-x [DIR]
extracts Perl program from the input stream. If DIR is specified, switches to this directory before running the program.

-0 VAL
(that's the number zero) designates an initial value for the record separator **$/**. See also **-l**.

28. 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.

RET 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 omitted, lists next window.

- Lists previous window.

w Lists window around current line.

f FILE Switches to FILE and start listing it.

l SUB Lists the named SUBroutine.

S List the names of all subroutines.

/PATTERN**/** Searches forwards for PATTERN.

?PATTERN**?** Searches backwards for PATTERN.

b [LINE [CONDITION]]
Sets breakpoint at LINE, default: current line.

b SUBNAME [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 the 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 character.

COMMAND Executes COMMAND as a Perl statement.