Replace Functions

qmake provides functions for processing the contents of variables during the configuration process. These functions are called *replace functions*. Typically, they return values that you can assign to other variables. You can obtain these values by prefixing a function with the $$ operator. Replace functions can be divided into built-in functions and function libraries.

See also Test Functions.

absolute_path(path[, base])

Returns the absolute path of `path`.

If `base` is not specified, uses the current directory as the base directory. If it is a relative path, it is resolved relative to the current directory before use.

For example, the following call returns the string `"/home/johndoe/myproject/readme.txt"`:

```
message($$absolute_path("readme.txt", "/home/johndoe/myproject"))
```

This function was introduced in Qt 5.0.

See also clean_path(), relative_path().

basename(variablename)

Returns the basename of the file specified in `variablename`.

For example:

```
FILE = /etc/passwd
FILENAME = $$basename(FILE) #passwd
```

cat(filename[, mode])

Returns the contents of `filename`. You can specify the following options for `mode`:

- `blob` returns the entire contents of the file as one value
- `lines` returns each line as a separate value (without line endings)
- `true` (default value) and `false` return file contents as separate values, split according to qmake value list splitting rules (as in variable assignments). If `mode` is `false`, values that contain only a newline character are inserted into the list to indicate where line breaks were in the file.

clean_path(path)

Returns `path` with directory separators normalized (converted to "/") and redundant ones removed, and "."s and ".."s resolved (as far as possible). This function is a wrapper around QDir::cleanPath.

This function was introduced in Qt 5.0.

See also absolute_path(), relative_path(), shell_path(), system_path().

dirname(file)

Returns the directory name part of the specified file. For example:

```
FILE = /etc/X11R6/XF86Config
DIRNAME = $$dirname(FILE) #/etc/X11R6
```

enumerate_vars

Returns a list of all defined variable names.

This function was introduced in Qt 5.0.

escape_expand(arg1 [, arg2 ..., argn])

Accepts an arbitrary number of arguments. It expands the escape sequences `\n`, `\r`, `\t` for each argument and returns the arguments as a list.

> **Note:** If you specify the string to expand literally, you need to escape the backslashes, as illustrated by the following code snippet:

```
message("First line$$escape_expand(\\n)Second line")
```

find(variablename, substr)

Returns all the values in `variablename` that match the regular expression `substr`.

```
MY_VAR3 = $$member(MY_VAR, 2) $$find(MY_VAR, t.*)
```

MY_VAR2 will contain '-Lone -Ltwo -Lthree -Lfour -Lfive', and MY_VAR3 will contain 'three two three'.

files(pattern[, recursive=false])

Expands the specified wildcard pattern and returns a list of filenames. If `recursive` is true, this function descends into subdirectories.

first(variablename)

Returns the first value of `variablename`.

For example, the following call returns `firstname`:

```
CONTACT = firstname middlename surname phone
message($$first(CONTACT))
```

See also take_first(), last().

format_number(number[, options...])

Returns `number` in the format specified by `options`. You can specify the following options:

- `ibase=n` sets the base of the input to n
- `obase=n` sets the base of the output to n
- `width=n` sets the minimum width of the output to n. If the output is shorter than `width`, it is padded with spaces
- `zeropad` pads the output with zeroes instead of spaces
- `padsign` prepends a space to positive values in the output
- `alwayssign` prepends a plus sign to positive values in the output
- `leftalign` places the padding to the right of the value in the output

Floating-point numbers are currently not supported.

For example, the following call converts the hexadecimal number BAD to 002989:

```
message($$format_number(BAD, ibase=16 width=6 zeropad))
```

This function was introduced in Qt 5.0.

fromfile(filename, variablename)

Evaluates `filename` as a qmake project file and returns the value assigned to `variablename`.

getenv(variablename)

Returns the value of the environment variable `variablename`. This is mostly equivalent to the `$$(variablename)` syntax. The `getenv` function, however, supports environment variables with parentheses in their name.

This function was introduced in Qt 5.12.

join(variablename, glue, before, after)

Joins the value of `variablename` with `glue`. If this value is not empty, this function prefixes the value with `before` and suffixes it with `after`. `variablename` is the only required field, the others default to empty strings. If you need to encode spaces in `glue`, `before`, or `after`, you must quote them.

last(variablename)

Returns the last value of `variablename`.

For example, the following call returns phone:

```
CONTACT = firstname middlename surname phone
message($$last(CONTACT))
```

See also take_last(), first().

list(arg1 [, arg2 ..., argn])

Takes an arbitrary number of arguments. It creates a uniquely named variable that contains a list of the arguments, and returns the name of that variable. You can use the variable to write a loop as illustrated by the following code snippet

```
for(var, $$list(foo bar baz)) {
    ...
}
```

instead of:

```
values = foo bar baz
for(var, values) {
    ...
}
```

lower(arg1 [, arg2 ..., argn])

Takes an arbitrary number of arguments and converts them to lower case.

See also upper().

member(variablename [, start [, end]])

Returns the slice of the list value of `variablename` with the zero-based element indices between `start` and `end` (inclusive).

If `start` is not given, it defaults to zero. This usage is equivalent to `$$first(variablename)`.

If `end` is not given, it defaults to `start`. This usage represents simple array indexing, as exactly one element will be returned.

It is also possible to specify start and end in a single argument, with the numbers separated by two periods.

Negative numbers represent indices starting from the end of the list, with -1 being the last element.

If either index is out of range, an empty list is returned.

If `end` is smaller than `start`, the elements are returned in reverse order.

> **Note:** The fact that the end index is inclusive and unordered implies that an empty list will be returned only when an index is invalid (which is implied by the input variable being empty).

See also str_member().

num_add(arg1 [, arg2 ..., argn])

Takes an arbitrary number of numeric arguments and adds them up, returning the sum.

Subtraction is implicitly supported due to the possibility to simply prepend a minus sign to a numeric value to negate it:

```
sum = $$num_add($$first, -$$second)
```

If the operand may be already negative, another step is necessary to normalize the number:

```
second_neg = -$$second
second_neg ~= s/^--//
sum = $$num_add($$first, $$second_neg)
```

This function was introduced in Qt 5.8.

prompt(question [, decorate])

Displays the specified `question`, and returns a value read from stdin.

If `decorate` is *true* (the default), the question gets a generic prefix and suffix identifying it as a prompt.

quote(string)

Converts a whole `string` into a single entity and returns the result. This is just a fancy way of enclosing the string into double quotes.

Returns the `string` with every special regular expression character escaped with a backslash. This function is a wrapper around QRegularExpression::escape.

read_registry(tree, key[, flag])

Returns the value of registry key `key` inside the tree `tree`.

Only the trees HKEY_CURRENT_USER (HKCU) and HKEY_LOCAL_MACHINE (HKLM) are supported.

The `flag` may be WOW64_32KEY (32) or WOW64_64KEY (64).

> **Note:** This function is available only on Windows hosts.

This function was introduced in Qt 5.12.1.

relative_path(filePath[, base])

Returns the path to `filePath` relative to `base`.

If `base` is not specified, it is the current project directory. If it is relative, it is resolved relative to the current project directory before use.

If `filePath` is relative, it is first resolved against the base directory; in that case, this function effectively acts as $$clean_path().

This function was introduced in Qt 5.0.

See also absolute_path(), clean_path().

replace(string, old_string, new_string)

Replaces each instance of `old_string` with `new_string` in the contents of the variable supplied as `string`. For example, the code

```
MESSAGE = This is a tent.
message($$replace(MESSAGE, tent, test))
```

prints the message:

```
This is a test.
```

resolve_depends(variablename, prefix)

This is an internal function that you will typically not need.

This function was introduced in Qt 5.0.

This function was introduced in Qt 5.0.

section(variablename, separator, begin, end)

Returns a section of the value of `variablename`. This function is a wrapper around QString::section.

For example, the following call outputs `surname`:

```
CONTACT = firstname:middlename:surname:phone
message($$section(CONTACT, :, 2, 2))
```

shadowed(path)

Maps the path from the project source directory to the build directory. This function returns `path` for in-source builds. It returns an empty string if `path` points outside of the source tree.

This function was introduced in Qt 5.0.

shell_path(path)

Converts all directory separators within `path` to separators that are compatible with the shell that is used while building the project (that is, the shell that is invoked by the make tool). For example, slashes are converted to backslashes when the Windows shell is used.

This function was introduced in Qt 5.0.

See also system_path().

shell_quote(arg)

Quotes `arg` for the shell that is used while building the project.

This function was introduced in Qt 5.0.

See also system_quote().

size(variablename)

Returns the number of values of `variablename`.

See also str_size().

sort_depends(variablename, prefix)

This is an internal function that you will typically not need.

This function was introduced in Qt 5.0.

sorted(variablename)

Returns the list of values in `variablename` with entries sorted in ascending ASCII order.

This function was introduced in Qt 5.8.

split(variablename, separator)

Splits the value of `variablename` into separate values, and returns them as a list. This function is a wrapper around QString::split.

For example:

```
CONTACT = firstname:middlename:surname:phone
message($$split(CONTACT, :))
```

sprintf(string, arguments...)

Replaces %1-%9 in `string` with the arguments passed in the comma-separated list of function `arguments` and returns the processed string.

str_member(arg [, start [, end]])

This function is identical to member(), except that it operates on a string value instead of a list variable, and consequently the indices refer to character positions.

This function can be used to implement many common string slicing operations:

```
# $$left(VAR, len)
left = $$str_member(VAR, 0, $$num_add($$len, -1))

# $$right(VAR, len)
right = $$str_member(VAR, -$$num, -1)

# $$mid(VAR, off, len)
mid = $$str_member(VAR, $$off, $$num_add($$off, $$len, -1))

# $$mid(VAR, off)
mid = $$str_member(VAR, $$off, -1)

# $$reverse(VAR)
reverse = $$str_member(VAR, -1, 0)
```

Note: In these implementations, a zero `len` argument needs to be handled separately.

See also member(), num_add().

This function was introduced in Qt 5.8.

str_size(arg)

Returns the number of characters in the argument

This function was introduced in Qt 5.8.

system(command[, mode[, stsvar]])

You can use this variant of the `system` function to obtain stdout from the command and assign it to a variable.

For example:

```
UNAME = $$system(uname -s)
contains( UNAME, [lL]inux ):message( This looks like Linux ($$UNAME) to me )
```

Like $$cat(), the *mode* argument takes `blob`, `lines`, `true`, and `false` as value. However, the legacy word splitting rules (i.e. empty or `true`, and `false`) differ subtly.

If you pass `stsvar`, the command's exit status will be stored in that variable. If the command crashes, the status will be -1, otherwise a non-negative exit code of the command's choosing. Usually, comparing the status with zero (success) is sufficient.

See also the test variant of system().

system_path(path)

Converts all directory separators within `path` to separators that are compatible with the shell that is used by the `system()` functions to invoke commands. For example, slashes are converted to backslashes for the Windows shell.

This function was introduced in Qt 5.0.

See also shell_path().

system_quote(arg)

Quotes `arg` for the shell that is used by the `system()` functions.

This function was introduced in Qt 5.0.

See also shell_quote().

take_first(variablename)

Returns the first value of `variablename` and removes it from the source variable.

This provides convenience for implementing queues, for example.

This function was introduced in Qt 5.8.

See also take_last(), first().

take_last(variablename)

Returns the last value of `variablename` and removes it from the source variable.

This provides convenience for implementing stacks, for example.

See also take_first(), last().

unique(variablename)

Returns the list of values in `variablename` with duplicate entries removed. For example:

```
ARGS = 1 2 3 2 5 1
ARGS = $$unique(ARGS) #1 2 3 5
```

upper(arg1 [, arg2 ..., argn])

Takes an arbitrary number of arguments and converts them to upper case.

See also lower().

val_escape(variablename)

Escapes the values of `variablename` in a way that enables parsing them as qmake code.

This function was introduced in Qt 5.0.

‹ Variables

Test Functions ›