替换函数

qmake 提供了在配置过程中处理变量内容的函数。这些函数称为*替换函数*。通常，它们返回可以分配给其他变量的值。可以通过在函数前面加上运算符前缀来获取这些值。替换函数可分为内置函数和函数库。$$

另请参见测试函数。

内置替换函数

基本替换函数作为内置函数实现。

absolute_path（路径[，基数]）

返回 的绝对路径。path

如果未指定，则使用当前目录作为基目录。如果是相对路径，则在使用前相对于当前目录解析。base

例如，以下调用返回字符串："/home/johndoe/myproject/readme.txt"

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

此功能在 Qt 5.0 中引入。

另请参见clean_path（）、relative_path（）。

基名（变量名）

返回 中指定的文件的基名。variablename

例如：

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

猫（文件名[，模式]）

返回 的内容。您可以为 指定以下选项：filenamemode

- true（默认值）并返回文件内容作为单独的值，根据qmake值列表拆分规则进行拆分（如在变量赋值中）。如果 is，则将仅包含换行符的值插入到列表中，以指示换行符在文件中的位置。falsemodefalse

clean_path（路径）

返回的目录分隔符规范化（转换为"/"）并删除冗余分隔符，以及 "。"和"..."已解决（尽可能）。此函数是围绕QDir：：清洁路径的包装器。path

此功能在 Qt 5.0 中引入。

另见absolute_path（）、relative_path（）、shell_path（）、system_path（）。

目录名称（文件）

返回指定文件的目录名称部分。例如：

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

enumerate_vars

返回所有已定义变量名称的列表。

此功能在 Qt 5.0 中引入。

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

接受任意数量的参数。它扩展了每个参数的转义序列，并将参数作为列表返回。\n\r\t

注意： 如果指定要按字面扩展的字符串，则需要对反斜杠进行转义，如以下代码段所示：

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

find(variablename, substr)

Returns all the values in that match the regular expression .variablenamesubstr

```
MY_VAR = one two three four
MY_VAR2 = $$join(MY_VAR, " -L", -L) -Lfive
MY_VAR3 = $$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'.

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

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

- `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 . If the output is shorter than , it is padded with spaces`nwidth`
- `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 to :`BAD002989`

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

This function was introduced in Qt 5.0.

fromfile(filename, variablename)

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

See also infile().

getenv(variablename)

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

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

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 with the zero-based element indices between and (inclusive).`variablenamestartend`

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

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 is smaller than , the elements are returned in reverse order.`endstart`

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 , and returns a value read from stdin.`question`

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

quote(string)

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

re_escape(string)

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

read_registry(tree, key[, flag])

Only the trees () and () are supported.HKEY_CURRENT_USERHKEY_LOCAL_MACHINE

The may be () or ().`flagWOW64_32KEY32WOW64_64KEY64`

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 relative to .`filePathbase`

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

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

This function was introduced in Qt 5.0.

See also absolute_path(), clean_path().

replace(string, old_string, new_string)

Replaces each instance of with in the contents of the variable supplied as . For example, the `codeold_stringnew_stringstring`

```
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.

reverse(variablename)

Returns the values of in reverse order.`variablename`

This function was introduced in Qt 5.0.

section(variablename, separator, begin, end)

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

```
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 for in-source builds. It returns an empty string if points outside of the source tree.`pathpath`

This function was introduced in Qt 5.0.

shell_path(path)

Converts all directory separators within 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.`path`

This function was introduced in Qt 5.0.

See also system_path().

shell_quote(arg)

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

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 with entries sorted in ascending ASCII order.`variablename`

Numerical sorting can be accomplished by zero-padding the values to a fixed length with the help of the format_number() function.

This function was introduced in Qt 5.8.

split(variablename, separator)

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

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

sprintf(string, arguments...)

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

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 argument needs to be handled separately.`len`

See also member(), num_add().

This function was introduced in Qt 5.8.

str_size(arg)

Returns the number of characters in the argument.

See also size().

This function was introduced in Qt 5.8.

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

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

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

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

If you pass , 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.`stsvar`

See also the test variant of system().

system_path(path)

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

This function was introduced in Qt 5.0.

See also shell_path().

system_quote(arg)

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

This function was introduced in Qt 5.0.

See also shell_quote().

take_first(variablename)

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

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 and removes it from the source variable.`variablename`

This provides convenience for implementing stacks, for example.

This function was introduced in Qt 5.8.

See also take_first(), last().

unique(variablename)

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

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 in a way that enables parsing them as qmake code.`variablename`

This function was introduced in Qt 5.0.

< Variables Test Functions >

DOCUMENTATION

Contribute to Qt
Forum
Wiki
Downloads
Marketplace

Feedback Sign In