cv::CommandLineParser Class Reference

Core functionality » Utility and system functions and macros

```
Designed for command line parsing. More...
#include "utility.hpp"
Public Member Functions
                           CommandLineParser (int argc, const char *const argv[], const String &keys)
                           Constructor. More...
                           CommandLineParser (const CommandLineParser &parser)
                           Copy constructor. More...
                           ~CommandLineParser ()
                           Destructor More
                     void about (const String &message)
                           Set the about message. More...
                     bool check () const
                           Check for parsing errors. More...
 template<typename T >
                       T get (const String &name, bool space_delete=true) const
                           Access arguments by name. More...
 template<typename T >
                        T get (int index, bool space_delete=true) const
                           Access positional arguments by index. More...
                   String getPathToApplication () const
                           Returns application path. More...
                     bool has (const String &name) const
                           Check if field was provided in the command line. More...
 CommandLineParser & operator= (const CommandLineParser &parser)
                           Assignment operator. More...
                     void printErrors () const
                           Print list of errors occured. More...
                     void printMessage () const
                           Print help message. More...
Protected Member Functions
 void getByIndex (int index, bool space_delete, int type, void *dst) const
 void getByName (const String &name, bool space delete, int type, void *dst) const
Protected Attributes
 Impl * impl
Detailed Description
Designed for command line parsing.
The sample below demonstrates how to use CommandLineParser:
CommandLineParser parser(argc, argv, keys);
parser.about("Application name v1.0.0");
if (parser.has("help"))
{
     parser.printMessage();
return 0;
int N = parser.get<int>("N");
double fps = parser.get<double>("fps");
String path = parser.get<String>("path");
use_time_stamp = parser.has("timestamp");
String img1 = parser.get<String>(0);
String img2 = parser.get<String>(1);
int repeat = parser.get<int>(2);
```

```
if (!parser.check())
{
     parser.printErrors();
return 0;
```

Keys syntax

The keys parameter is a string containing several blocks, each one is enclosed in curley braces and describes one argument. Each argument contains three parts separated by the | symbol:

- 1. argument names is a space-separated list of option synonyms (to mark argument as positional, prefix it with the @ symbol)
- 2. default value will be used if the argument was not provided (can be empty)
- 3. help message (can be empty)

For example:

```
const String keys =
   "{help h usage ?
   "{@image1
   "{@image2
   "{@repeat
   "{path
   "{fps
                                                                                     print this message
imagel for compare
image2 for compare
number
                                                              <none>
                                                                                     number
path to file
fps for output video
count of objects
use time stamp
                                                              -1.0
100
                 N count
                ts timestamp
```

Note that there are no default values for help and timestamp so we can check their presence using the has() method. Arguments with default values are considered to be always present. Use the get() method in these cases to check their actual value instead.

String keys like get<String>("\@image1") return the empty string "" by default - even with an empty default value. Use the special <none> default value to enforce that the returned string must not be empty. (like in get<String>("\@image2"))

Usage

For the described keys:

```
# Good call (3 positional parameters: image1, image2 and repeat; N is 200, ts is true) \frac{1}{2} ./app -N=200 1.png 2.jpg 19 -ts
```

Examples:

contours2.cpp, convexhull.cpp, cout_mat.cpp, demhist.cpp, distrans.cpp, edge.cpp, ffilldemo.cpp, filestorage.cpp, fitellipse.cpp, grabcut.cpp, houghcircles.cpp, laplace.cpp, lsd lines.cpp, morphology2.cpp, pca.cpp, polar transforms.cpp, segment objects.cpp, and

Constructor & Destructor Documentation

```
cv::CommandLineParser::CommandLineParser ( int
                                              const char *const argv[],
                                              const String &
                                                                keys
Constructor.
```

Initializes command line parser object

Parameters

argc number of command line arguments (from main())

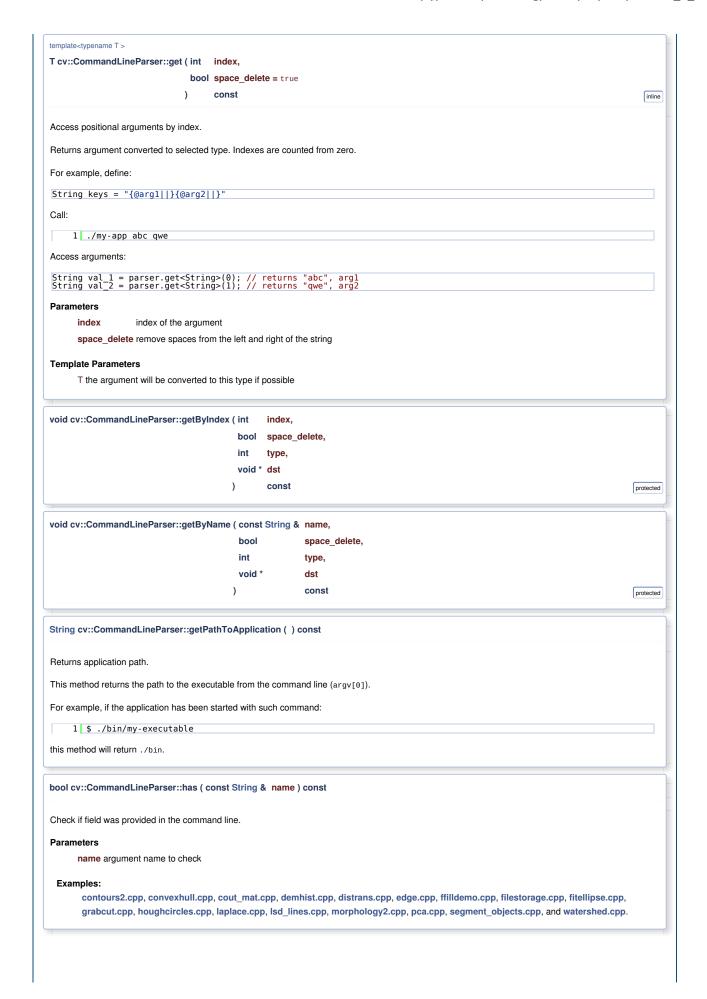
argv array of command line arguments (from main())

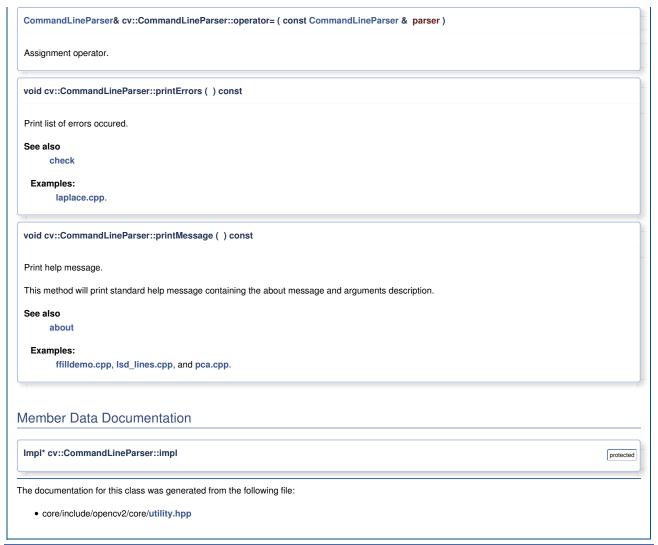
keys string describing acceptable command line parameters (see class description for syntax)

cv::CommandLineParser::CommandLineParser (const CommandLineParser & parser)

Copy constructor.

cv::CommandLineParser::~CommandLineParser() Destructor Member Function Documentation void cv::CommandLineParser::about (const String & message) Set the about message. The about message will be shown when printMessage is called, right before arguments table. bool cv::CommandLineParser::check () const Check for parsing errors. Returns true if error occured while accessing the parameters (bad conversion, missing arguments, etc.). Call printErrors to print error messages list. laplace.cpp. template<typename T > T cv::CommandLineParser::get (const String & name, bool space_delete = true const inline Access arguments by name. Returns argument converted to selected type. If the argument is not known or can not be converted to selected type, the error flag is set (can be checked with check). For example, define: String keys = "{N count||}"; Call: 1 \$./my-app -N=20 2 # or 3 \$./my-app --count=20 int N = parser.get<int>("N"); **Parameters** name of the argument name space_delete remove spaces from the left and right of the string **Template Parameters** T the argument will be converted to this type if possible Note You can access positional arguments by their @-prefixed name: parser.get<String>("@image"); demhist.cpp, distrans.cpp, edge.cpp, filldemo.cpp, filestorage.cpp, fitellipse.cpp, grabcut.cpp, houghcircles.cpp, laplace.cpp, lsd_lines.cpp, morphology2.cpp, pca.cpp, polar_transforms.cpp, segment_objects.cpp, and watershed.cpp.





Generated on Mon Sep 19 2016 02:52:59 for OpenCV by (10) 1.8.12