Command Line Processor

Generado por Doxygen 1.9.2

Capítulo 1

README

This is a minimal example of a book based on R Markdown and **bookdown** (https://github.↩ com/rstudio/bookdown). Please see the page "Get Started" at https://bookdown.org/home/about/ for how to compile this example.

Capítulo 2

An holistic proposal and tool to parse command line input

2.1. disclaimer

Aqui el disclaimer

2.2. Motivacion

Cualquier programa susceptible de ser ejecutado a traves de una linea de comandos necesita una serie de parámetros de entrada de lo contario siempre generaria el mismo resultado.

Ahora bien, dependiendo del sistema operativo, la forma habitual de indicar estos parámetros varía; es mas, cada programador podría definir un nuevo sistema de entrada de parámetros. Esto obliga, o al menos debería, a que un programa multiplataforma debería adecuarse a cada sistema para aceptar el estilo de cada uno de ellos.

2.2.1. sistemas nix

Tipicamente estos sistemas utilizan los simbolos '-' o '–' para indicar un parámetro, donde:

- La primera forma se aplica a parámetros definidos por un solo caracter y permite además concatenarlos
- La segunda forma se entiende como parámetro extendido definido por una palabra

Ejemplos:

`command -h` `command --help` `command -af` `command -a -f` `command -o output_file`

2.2.2. sistemas windows

`command /h` `command /-Y` `command /D:value` `command /o output_file`

y actualmente con Powershell tambien se soportan las formas '-' y '–' y en algunos casos combinadas con ':'

2.2.3. ZOS y otros

Otros sistemas operativos soportan otros metodos. Por ejemplo zOS usa /_characters_, donde *characters* es el mínimo numero de caracteres necesario para identificar un comando.

Por ejemplo `/pre pattern` y `/preffix pattern` son equivalentes en el sentido de que no existe mas que un comando que empiece por la secuencia `pre`

2.3. Definiciones

Introduzcamos algunas definiciones:

Parametro o **argumento** son sinonimos y es cualquier elemento que aparece despues del comando en si mismo

Ahora bien, estos parámetros pueden ser de tres tipos:

1. Datos de entrada
2. Opciones
3. Flags

Y cada programa puede aceptar un determinado número de opciones, incluida ninguna, y de flags, incluidos ninguno, pero en el caso de que exista al menos uno, este, por definición debe asumir un valor por defecto cuando no es especificado en la linea de comandos.

2.3.1. Datos de entrada

Es cualquier elemento o secuencia de ellos que debe ser procesado por el programa.

Por ejemplo: `cat fichero.txt` mostrara por pantalla el contenido del fichero fichero.txt

2.3.2. Opciones

Es cualquier elemento o secuencia de ellos que modifica o afecta al comportamiento del programa aportándole algun tipo de información

Por ejemplo: `grep -e _pattern_` o `grep --regexp _pattern_` indica a grep que patrón usar

2.3.3. Flags

Es cualquier elemento que activa o desactiva alguna característica o funcionalidad del programa. El caso mas típico seria el flag: `-h` o `--help` que indica al programa que muestre su ayuda

Otros ejemplos podrían ser:

`command -v` o `command --verbose` que suele usarse para generar información de progreso `command -s` o `command --silent` que suele usarse para evitar que se genere esa información

Notese que estos flags son complementarios; es decir, afectan a las misma funcionalidad.

2.4. Incoherencias

Con estos sistemas se plantean los siguientes problemas:

Dado un parametro P o Parameter:

- ¿Como podemos saber si es un flag o una opcion?
- Si es un flag, ¿Activa o desactiva esa funcionalidad? (Vease el caso -verbose y -silent)
- ¿Por que tengo que elegir como usuario, y controlar como desarrollador, las dos opciones -P y –Parameter?
- En el caso de que no exista una version corta del parámetro, ¿Por que necesito escribir Parameter si con la primera letra ya seria suficiente?
- Cuanto mas caracteres tenga que escribir mas posible es que introduzca un error de tipografia

2.5. Propuesta

El siguiente esquema propone resolver estas "deficiencias" y clarificar de cara al usuario los significados de cada parámetro:

1. Un flag se prefija siempre con los signos + o - y su significado es **siempre** el mismo: + activa la funcionalidad y - la desactiva.
2. Dado que los flags no necesitan información asociada se pueden concatenar en una secuencia de caracteres con la unica condicion de que ninguno de ellos empiece por el mismo caracter
3. Una opcion se prefija **siempre** con la barra / y su valor asociado se establece en la siguiente palabra
4. Cualquier parámetro queda definido en el momento en el que no existe duda acerca de la intención del usuario; es decir, no es necesario introducir el nombre completo del parámetro

Por ultimo, y como consejo, el manual de usuario o la ayuda en linea, deberia indicar los valores por defecto de todos los flags y opciones soportadas por el programa

Capítulo 3

Indice de módulos

3.1. Módulos

Lista de todos los módulos:

Capítulo 4

Indice de namespaces

4.1. Lista de 'namespaces'

Lista de toda la documentación de los 'namespaces', con una breve descripción:

Capítulo 5

Indice jerárquico

5.1. Jerarquía de la clase

Esta lista de herencias esta ordenada aproximadamente por orden alfabético:

Capítulo 6

Índice de clases

6.1. Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

Capítulo 7

Indice de archivos

7.1. Lista de archivos

Lista de todos los archivos documentados y con descripciones breves:

Capítulo 8

Documentación de módulos

8.1. Tipos definidos para C++

Enumeraciones

- enum class Type {
  STRING , CHAR , BOOL , NUMBER ,
  DECIMAL , LONGNUMBER , LONGDECIMAL , DATE ,
  TIME , DATETIME , TMS , DIR ,
  DIR_EXISTS , FILE , FILE_EXISTS , **FLAG** }

  Define el tipo de dato esperado del parametro.
- enum class Source {
  DEFAULT , ENV , **CMDLINE** , CODE ,
  AUTO }

  Indicates the origin of thelast value of parameter.

8.1.1. Descripción detallada

Conjunto de tipos para simplificar la codificacion

8.1.2. Documentación de las enumeraciones

8.1.2.1. Source

```
enum class Source  [strong]
```

Indicates the origin of thelast value of parameter.

Valores de enumeraciones

DEFAULT	Default value
ENV	Value has been set in the environment
CODE	Value has been update programatically
AUTO	**Reserved** Hide parameter from queries

Definición en la línea 19 del archivo types.hpp.

8.1.2.2. Type

```
enum class Type  [strong]
```

Define el tipo de dato esperado del parametro.

Define the expected type of the string passed as parameter.

It will be used to check the validity of the input

Valores de enumeraciones

STRING	string (valor por defecto)
CHAR	const char *
BOOL	boolean
NUMBER	Whole number (long/int)
DECIMAL	Decimal number (double/float)
LONGNUMBER	Whole number (long long/long int)
LONGDECIMAL	Whole number (long double/long float)
DATE	A date formatted as locale. Example: `dd/mm/yy`
TIME	A time in format `hh:mm:ss`
DATETIME	An string like `yyyy/mm/dd[ -]hh:mm:ss` Note that using space requires quote the parameter
TMS	An string like `yyyy/mm/dd-hh:mm:ss.nnnnnn` Note that spaces is not allowed
DIR	A possibly correct path
DIR_EXISTS	An existing path before execution
FILE	A possibly correct file name
FILE_EXISTS	An existing file before execution

Definición en la línea 13 del archivo types.hpp.

Capítulo 9

Documentación de namespaces

9.1. Referencia del Namespace _cmdline

Private namespace.

Clases

- class Argument
- class CommandLine
- class Define
- class Group
- class ParameterTree

Funciones

- void add2tree (ParameterTree *root[ ], const char *word)
- char * **makeChar** (std::string str)
- bool makeBoolean (const char *value)
- bool **makeBoolean** (std::string value)
- char * makeUpper (const char *str)
- char * strUpper (const char *str)
- vector< std::string > splitArgument (const char *parm)
- vector< std::string > tokenize (const char *src, const char *pat)
- vector< int > tokenizeNumber (const char *src, const char *pat)
- void defaultDate (char *aux)
- void validateEntry (const char *parm, const char *prev)
- void **validateValue** (const char *value, cmdline::Type type)
- long validateNumber (const char *value)
- double validateDecimal (const char *value)
- long long validateLongNumber (const char *value)
- long double validateLongDecimal (const char *value)
- filesystem::path validateDir (const char *value)
- filesystem::path validateDirExist (const char *value)
- filesystem::path validateFile (const char *value)
- filesystem::path validateFileExist (const char *value)
- struct tm * validateTime (const char *value)

- struct tm ∗ validateDate (const char ∗value, int fmt=-1)
- struct tm ∗ validateDateTime (const char ∗value)
- char ∗ validateTimestamp (const char ∗value)
- int makeInteger (const char ∗value)
- float makeFloat (const char ∗value)
- struct tm makeTm (const char ∗value)
- bool valMakeBoolean (const char ∗value)
- bool valMakeBoolean (std::string value)
- ParameterTree ∗ createTree (const char ∗word)
- ParameterTree ∗ joinTree (ParameterTree ∗root, const char ∗word)
- char ∗ makeChar (string str)
- bool makeBoolean (string value)
- struct tm ∗ makeDateTime (char ∗sdate, char ∗stime)
- int isLeap (int year)
- double validateDecimal2 (const char ∗value)
- long double validateLongDecimal2 (const char ∗value)
- void validateDateValue (const char ∗value, vector< int > dt)
- void validateValue (const char ∗value, Type type)
- template<typename T >
  T getValue (const char ∗value, Type type)
- template<typename T >
  void checkValue (auto value)

Variables

- ParameterTree ∗ rootOptions [128]
- ParameterTree ∗ rootFlags [128]
- char _upper [64] = ""

9.1.1. Descripción detallada

Private namespace.

9.1.2. Documentación de las funciones

9.1.2.1. add2tree()

```
void add2tree (
            ParameterTree * root[],
            const char * word )
```

Definición en la línea 54 del archivo tools.cpp.

9.1.2.2. checkValue()

```
void _cmdline::checkValue (
             auto value )
```

Definición en la línea 356 del archivo validations.cpp.

9.1.2.3. createTree()

```
ParameterTree * _cmdline::createTree (
             const char * word )
```

Definición en la línea 17 del archivo tools.cpp.

9.1.2.4. defaultDate()

```
void defaultDate (
             char * aux )
```

Definición en la línea 172 del archivo tools.cpp.

9.1.2.5. getValue()

```
T _cmdline::getValue (
             const char * value,
             Type type )
```

Definición en la línea 339 del archivo validations.cpp.

9.1.2.6. isLeap()

```
int _cmdline::isLeap (
             int year )
```

Definición en la línea 65 del archivo validations.cpp.

9.1.2.7. joinTree()

```
ParameterTree * _cmdline::joinTree (
            ParameterTree * root,
            const char * word )
```

Definición en la línea 28 del archivo tools.cpp.

9.1.2.8. makeBoolean() [1/2]

```
bool makeBoolean (
            const char * value )
```

Definición en la línea 76 del archivo tools.cpp.

9.1.2.9. makeBoolean() [2/2]

```
bool _cmdline::makeBoolean (
            string value )
```

Definición en la línea 83 del archivo tools.cpp.

9.1.2.10. makeChar()

```
char * _cmdline::makeChar (
            string str )
```

Definición en la línea 70 del archivo tools.cpp.

9.1.2.11. makeDateTime()

```
struct tm * _cmdline::makeDateTime (
            char * sdate,
            char * stime )
```

Definición en la línea 35 del archivo validations.cpp.

9.1.2.12. makeFloat()

```
float makeFloat (
            const char * value )
```

Definición en la línea 366 del archivo validations.cpp.

9.1.2.13. makeInteger()

```
int makeInteger (
            const char * value )
```

Definición en la línea 361 del archivo validations.cpp.

9.1.2.14. makeTm()

```
struct tm makeTm (
            const char * value )
```

Definición en la línea 371 del archivo validations.cpp.

9.1.2.15. makeUpper()

```
char * makeUpper (
            const char * str )
```

Definición en la línea 86 del archivo tools.cpp.

9.1.2.16. splitArgument()

```
vector< string > splitArgument (
            const char * parm )
```

Definición en la línea 94 del archivo tools.cpp.

9.1.2.17. strUpper()

```
char * strUpper (
            const char * str )
```

Definición en la línea 152 del archivo tools.cpp.

9.1.2.18. tokenize()

```
vector< string > tokenize (
            const char * src,
            const char * pat )
```

Definición en la línea 136 del archivo tools.cpp.

9.1.2.19. tokenizeNumber()

```
vector< int > tokenizeNumber (
            const char * src,
            const char * pat )
```

Definición en la línea 145 del archivo tools.cpp.

9.1.2.20. validateDate()

```
struct tm * validateDate (
            const char * value,
            int fmt = -1 )
```

Definición en la línea 198 del archivo validations.cpp.

9.1.2.21. validateDateTime()

```
struct tm * validateDateTime (
            const char * value )
```

Definición en la línea 228 del archivo validations.cpp.

9.1.2.22. validateDateValue()

```
void _cmdline::validateDateValue (
            const char * value,
            vector< int > dt )
```

Definición en la línea 189 del archivo validations.cpp.

9.1.2.23. validateDecimal()

```
double validateDecimal (
            const char * value )
```

Definición en la línea 126 del archivo validations.cpp.

9.1.2.24. validateDecimal2()

```
double _cmdline::validateDecimal2 (
            const char * value )
```

Definición en la línea 71 del archivo validations.cpp.

9.1.2.25. validateDir()

```
filesystem::path validateDir (
            const char * value )
```

Definición en la línea 273 del archivo validations.cpp.

9.1.2.26. validateDirExist()

```
filesystem::path validateDirExist (
            const char * value )
```

Definición en la línea 295 del archivo validations.cpp.

9.1.2.27. validateEntry()

```
void validateEntry (
            const char * parm,
            const char * prev )
```

Definición en la línea 98 del archivo validations.cpp.

9.1.2.28. validateFile()

```
filesystem::path validateFile (
            const char * value )
```

Definición en la línea 306 del archivo validations.cpp.

9.1.2.29. validateFileExist()

```
filesystem::path validateFileExist (
            const char * value )
```

Definición en la línea 314 del archivo validations.cpp.

9.1.2.30. validateLongDecimal()

```
long double validateLongDecimal (
            const char * value )
```

Definición en la línea 151 del archivo validations.cpp.

9.1.2.31. validateLongDecimal2()

```
long double _cmdline::validateLongDecimal2 (
            const char * value )
```

Definición en la línea 84 del archivo validations.cpp.

9.1.2.32. validateLongNumber()

```
long long validateLongNumber (
            const char * value )
```

Definición en la línea 114 del archivo validations.cpp.

9.1.2.33. validateNumber()

```
long validateNumber (
            const char * value )
```

Definición en la línea 102 del archivo validations.cpp.

9.1.2.34. validateTime()

```
struct tm * validateTime (
            const char * value )
```

Definición en la línea 177 del archivo validations.cpp.

9.1.2.35. validateTimestamp()

```
char * validateTimestamp (
            const char * value )
```

Definición en la línea 248 del archivo validations.cpp.

9.1.2.36. validateValue()

```
void _cmdline::validateValue (
            const char * value,
            Type type )
```

Definición en la línea 321 del archivo validations.cpp.

9.1.2.37. valMakeBoolean() [1/2]

```
bool valMakeBoolean (
            const char * value )
```

Definición en la línea 400 del archivo validations.cpp.

9.1.2.38. valMakeBoolean() [2/2]

```
bool valMakeBoolean (
            std::string value )
```

Definición en la línea 403 del archivo validations.cpp.

9.1.3. Documentación de las variables

9.1.3.1. _upper

```
char _upper[64] = ""
```

Definición en la línea 16 del archivo tools.cpp.

9.1.3.2. rootFlags

```
ParameterTree* rootFlags[128]
```

Definición en la línea 16 del archivo commandline.cpp.

9.1.3.3. rootOptions

```
ParameterTree* rootOptions[128]
```

Definición en la línea 15 del archivo commandline.cpp.

9.2. Referencia del Namespace cmdline

Public namespace to avoid naming mangling.

Clases

- class CmdLine
- class CmdLineDuplicateArgumentException
- class CmdLineException
- class CmdLineI
- class CmdLineInvalidTypeException
- class CmdLineIS
- class CmdLineNotFoundException
- class CmdLineParameterException
- class CmdLineS
- class CmdLineValueException
- class HelpDetailedRequested
- class HelpException
- class HelpRequested
- class Parm
- class ParmFlag
- class ParmOption

typedefs

- typedef std::vector< Parm > Parameters
- using TYPE_STRING = std::string
- using TYPE_BOOL = bool
- using TYPE_NUMBER = long
- using TYPE_LONG = long
- using TYPE_INT = int
- using TYPE_DECIMAL = float
- using TYPE_FLOAT = float
- using TYPE_DOUBLE = double
- using TYPE_LONGNUMBER = long long
- using TYPE_LONGDECIMAL = float
- using TYPE_DATE = std::tm
- using TYPE_TIME = std::tm
- using TYPE_DATETIME = std::tm
- using TYPE_TMS = char *
- using TYPE_DIR = filesystem::path
- using TYPE_FILE = filesystem::path
- using Flags = std::unordered_map< std::string, bool >
- using Options = std::unordered_map< std::string, vector< std::string > >

Enumeraciones

- enum class Type {
  STRING , CHAR , BOOL , NUMBER ,
  DECIMAL , LONGNUMBER , LONGDECIMAL , DATE ,
  TIME , DATETIME , TMS , DIR ,
  DIR_EXISTS , FILE , FILE_EXISTS , **FLAG** }

  Define el tipo de dato esperado del parametro.
- enum class Source {
  DEFAULT , ENV , **CMDLINE** , CODE ,
  AUTO }

  Indicates the origin of thelast value of parameter.

Funciones

- template<typename T >
  const vector< T > getOptionValuesAs (const char *name)
- void _cleanClass ()

Variables

- CmdLine * singleton_ = nullptr
- _cmdline::CommandLine * _commandLine
- char ** _argv = nullptr
- int _nargc
- Parameters _parms

9.2.1. Descripción detallada

Public namespace to avoid naming mangling.

Base class for cmdline exceptions

As library process argument from command line it inherits from invalid_argument

Parámetros

msg	Text to show

9.2.2. Documentación de los 'typedefs'

9.2.2.1. Flags

```
using Flags = std::unordered_map<std::string, bool>
```

Definición en la línea 38 del archivo types.hpp.

9.2.2.2. Options

```
using Options = std::unordered_map<std::string, vector<std::string> >
```

Definición en la línea 39 del archivo types.hpp.

9.2.2.3. Parameters

```
typedef std::vector<Parm> Parameters
```

Definición en la línea 77 del archivo cmdline.hpp.

9.2.2.4. TYPE_BOOL

```
using TYPE_BOOL = bool
```

Definición en la línea 22 del archivo types.hpp.

9.2.2.5. TYPE_DATE

```
using TYPE_DATE = std::tm
```

Definición en la línea 31 del archivo types.hpp.

9.2.2.6. TYPE_DATETIME

```
using TYPE_DATETIME = std::tm
```

Definición en la línea 33 del archivo types.hpp.

9.2.2.7. TYPE_DECIMAL

```
using TYPE_DECIMAL = float
```

Definición en la línea 26 del archivo types.hpp.

9.2.2.8. TYPE_DIR

```
using TYPE_DIR = filesystem::path
```

Definición en la línea 35 del archivo types.hpp.

9.2.2.9. TYPE_DOUBLE

```
using TYPE_DOUBLE = double
```

Definición en la línea 28 del archivo types.hpp.

9.2.2.10. TYPE_FILE

```
using TYPE_FILE = filesystem::path
```

Definición en la línea 36 del archivo types.hpp.

9.2.2.11. TYPE_FLOAT

```
using TYPE_FLOAT = float
```

Definición en la línea 27 del archivo types.hpp.

9.2.2.12. TYPE_INT

```
using TYPE_INT = int
```

Definición en la línea 25 del archivo types.hpp.

9.2.2.13. TYPE_LONG

```
using TYPE_LONG = long
```

Definición en la línea 24 del archivo types.hpp.

9.2.2.14. TYPE_LONGDECIMAL

```
using TYPE_LONGDECIMAL = float
```

Definición en la línea 30 del archivo types.hpp.

9.2.2.15. TYPE_LONGNUMBER

```
using TYPE_LONGNUMBER = long long
```

Definición en la línea 29 del archivo types.hpp.

9.2.2.16. TYPE_NUMBER

```
using TYPE_NUMBER = long
```

Definición en la línea 23 del archivo types.hpp.

9.2.2.17. TYPE_STRING

```
using TYPE_STRING = std::string
```

Definición en la línea 21 del archivo types.hpp.

9.2.2.18. TYPE_TIME

```
using TYPE_TIME = std::tm
```

Definición en la línea 32 del archivo types.hpp.

9.2.2.19. TYPE_TMS

```
using TYPE_TMS = char*
```

Definición en la línea 34 del archivo types.hpp.

9.2.3. Documentación de las funciones

9.2.3.1. _cleanClass()

```
void cmdline::_cleanClass ( )
```

Definición en la línea 14 del archivo cmdline_flavors.cpp.

9.2.3.2. getOptionValuesAs()

```
const vector< T > cmdline::getOptionValuesAs (
            const char * name )
```

Definición en la línea 63 del archivo cmdline.cpp.

9.2.4. Documentación de las variables

9.2.4.1. _argv

```
char** _argv = nullptr
```

Definición en la línea 10 del archivo cmdline_flavors.cpp.

9.2.4.2. _commandLine

```
_cmdline::CommandLine* _commandLine
```

Definición en la línea 15 del archivo cmdline.cpp.

9.2.4.3. _nargc

```
int _nargc
```

Definición en la línea 11 del archivo cmdline_flavors.cpp.

9.2.4.4. _parms

```
Parameters _parms
```

Definición en la línea 12 del archivo cmdline_flavors.cpp.

9.2.4.5. singleton_

```
CmdLine* singleton_ = nullptr
```

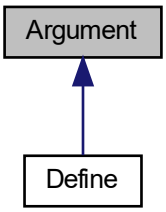
Definición en la línea 12 del archivo cmdline.cpp.

Capítulo 10

Documentación de las clases

10.1. Referencia de la Clase Argument

Diagrama de herencias de Argument



Métodos públicos

- Argument (Argument ∗arg)
- Argument (Parm ∗parm)
- Argument (const char ∗name, const char ∗value)
- Argument (const char ∗name, const char ∗value, Source source)
- Argument (const char ∗name, const char ∗value, Type type)
- Argument & setFromEnv (const char ∗value)
- Argument & setValue (bool value)
- Argument & setValue (const char ∗value)
- Argument & setValue (std::string value)
- Argument & addValue (std::string value)
- Argument & addValues (vector< string > values)
- const char ∗ getValue ()
- vector< const char ∗ > getValues ()
- vector< string > getStringValues ()
- bool getBoolean ()
- Argument & initValues (vector< string > values)
- Argument & makeUpper ()

Atributos públicos

- string name
- Source source = Source::DEFAULT
- Type type = Type::STRING
- bool multiple = false
- string defValue
- set< string > values

10.1.1. Descripción detallada

Definición en la línea 11 del archivo argument.hpp.

10.1.2. Documentación del constructor y destructor

10.1.2.1. ~Argument()

```
~Argument ( )
```

Definición en la línea 60 del archivo argument.cpp.

10.1.2.2. Argument() [1/5]

```
Argument (
            Argument * arg )
```

Definición en la línea 27 del archivo argument.cpp.

10.1.2.3. Argument() [2/5]

```
Argument (
            Parm * parm )
```

Definición en la línea 19 del archivo argument.cpp.

10.1.2.4. Argument() [3/5]

```
Argument (
            const char * name,
            const char * value )
```

Definición en la línea 39 del archivo argument.cpp.

10.1.2.5. Argument() [4/5]

```
Argument (
            const char * name,
            const char * value,
            Source source )
```

Definición en la línea 44 del archivo argument.cpp.

10.1.2.6. Argument() [5/5]

```
Argument (
            const char * name,
            const char * value,
            Type type )
```

Definición en la línea 56 del archivo argument.cpp.

10.1.3. Documentación de las funciones miembro

10.1.3.1. addValue()

```
Argument & addValue (
            std::string value )
```

Definición en la línea 124 del archivo argument.cpp.

10.1.3.2. addValues()

```
Argument & addValues (
            vector< string > values )
```

Definición en la línea 129 del archivo argument.cpp.

10.1.3.3. getBoolean()

```
bool getBoolean ( )
```

Definición en la línea 81 del archivo argument.cpp.

10.1.3.4. getStringValues()

```
vector< string > getStringValues ( )
```

Definición en la línea 75 del archivo argument.cpp.

10.1.3.5. getValue()

```
const char * getValue ( )
```

Definición en la línea 64 del archivo argument.cpp.

10.1.3.6. getValues()

```
vector< const char * > getValues ( )
```

Definición en la línea 67 del archivo argument.cpp.

10.1.3.7. initValues()

```
Argument & initValues (
            vector< string > values )
```

Definición en la línea 95 del archivo argument.cpp.

10.1.3.8. makeUpper()

```
Argument & makeUpper ( )
```

Definición en la línea 134 del archivo argument.cpp.

10.1.3.9. setFromEnv()

```
Argument & setFromEnv (
            const char * value )
```

Definición en la línea 102 del archivo argument.cpp.

10.1.3.10. setValue() [1/3]

```
Argument & setValue (
            bool value )
```

Definición en la línea 113 del archivo argument.cpp.

10.1.3.11. setValue() [2/3]

```
Argument & setValue (
            const char * value )
```

Definición en la línea 108 del archivo argument.cpp.

10.1.3.12. setValue() [3/3]

```
Argument & setValue (
            std::string value )
```

Definición en la línea 119 del archivo argument.cpp.

10.1.4. Documentación de los datos miembro

10.1.4.1. defValue

```
string defValue
```

Definición en la línea 17 del archivo argument.hpp.

10.1.4.2. multiple

```
bool multiple = false
```

Definición en la línea 16 del archivo argument.hpp.

10.1.4.3. name

```
string name
```

Definición en la línea 13 del archivo argument.hpp.

10.1.4.4. source

```
Source source = Source::DEFAULT
```

Definición en la línea 14 del archivo argument.hpp.

10.1.4.5. type

```
Type type = Type::STRING
```

Definición en la línea 15 del archivo argument.hpp.

10.1.4.6. values

```
set<string> values
```

Definición en la línea 18 del archivo argument.hpp.

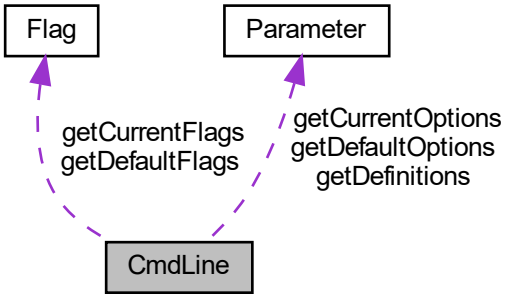
La documentación para esta clase fue generada a partir de los siguientes ficheros:

- argument.hpp
- argument.cpp

10.2. Referencia de la Clase CmdLine

Singleton encargado de mantener la informacion pasada por la linea de comandos.

Diagrama de colaboración para CmdLine:



Atributos públicos

- logical(∗ hasFlag)(const char ∗name)
- logical(∗ hasOption)(const char ∗name)
- logical(∗ hasDefine)(const char ∗name)
- logical(∗ isOptionMultiple)(const char ∗name)
- logical(∗ isDefineMultiple)(const char ∗name)
- logical(∗ getFlag)(const char ∗name)
- int(∗ getOptionNumValues)(const char ∗name)
- int(∗ getDefineNumValues)(const char ∗name)
- const char ∗(∗ getOption)(const char ∗name)
- const char ∗(∗ getDefine)(const char ∗name)
- const char ∗∗(∗ getOptionValues)(const char ∗name)
- const char ∗∗(∗ getDefineValues)(const char ∗name)
- Flag ∗∗(∗ getDefaultFlags)(logical active)
- Flag ∗∗(∗ getCurrentFlags)(logical active)
- Parameter ∗∗(∗ getDefaultOptions)()
- Parameter ∗∗(∗ getCurrentOptions)()
- Parameter ∗∗(∗ getDefinitions)()

10.2.1. Descripción detallada

Singleton encargado de mantener la informacion pasada por la linea de comandos.

Definición en la línea 4 del archivo cmdline.h.

10.2.2. Documentación de los datos miembro

10.2.2.1. getCurrentFlags

```
Flag **(* getCurrentFlags) (logical active)
```

Definición en la línea 18 del archivo cmdline.h.

10.2.2.2. getCurrentOptions

```
Parameter **(* getCurrentOptions) ()
```

Definición en la línea 20 del archivo cmdline.h.

10.2.2.3. getDefaultFlags

```
Flag **(* getDefaultFlags) (logical active)
```

Definición en la línea 17 del archivo cmdline.h.

10.2.2.4. getDefaultOptions

```
Parameter **(* getDefaultOptions) ()
```

Definición en la línea 19 del archivo cmdline.h.

10.2.2.5. getDefine

```
const char *(* getDefine) (const char *name)
```

Definición en la línea 14 del archivo cmdline.h.

10.2.2.6. getDefineNumValues

```
int(* getDefineNumValues) (const char *name)
```

Definición en la línea 12 del archivo cmdline.h.

10.2.2.7. getDefineValues

```
const char **(* getDefineValues) (const char *name)
```

Definición en la línea 16 del archivo cmdline.h.

10.2.2.8. getDefinitions

```
Parameter **(* getDefinitions) ()
```

Definición en la línea 21 del archivo cmdline.h.

10.2.2.9. getFlag

```
logical(* getFlag) (const char *name)
```

Definición en la línea 10 del archivo cmdline.h.

10.2.2.10. getOption

```
const char *(* getOption) (const char *name)
```

Definición en la línea 13 del archivo cmdline.h.

10.2.2.11. getOptionNumValues

```
int(* getOptionNumValues) (const char *name)
```

Definición en la línea 11 del archivo cmdline.h.

10.2.2.12. getOptionValues

```
const char **(* getOptionValues) (const char *name)
```

Definición en la línea 15 del archivo cmdline.h.

10.2.2.13. hasDefine

```
logical(* hasDefine) (const char *name)
```

Definición en la línea 7 del archivo cmdline.h.

10.2.2.14. hasFlag

```
logical(* hasFlag) (const char *name)
```

Definición en la línea 5 del archivo cmdline.h.

10.2.2.15. hasOption

```
logical(* hasOption) (const char *name)
```

Definición en la línea 6 del archivo cmdline.h.

10.2.2.16. isDefineMultiple

```
logical(* isDefineMultiple) (const char *name)
```

Definición en la línea 9 del archivo cmdline.h.

10.2.2.17. isOptionMultiple

```
logical(* isOptionMultiple) (const char *name)
```

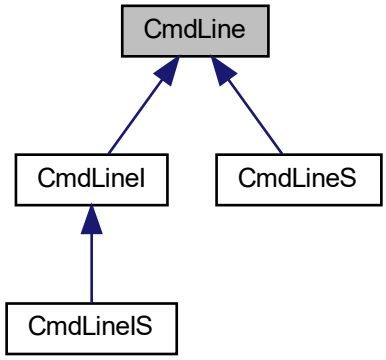
Definición en la línea 8 del archivo cmdline.h.

La documentación para esta clase fue generada a partir del siguiente fichero:

- cmdline.h

10.3. Referencia de la Clase CmdLine

Diagrama de herencias de CmdLine



Métodos públicos

- CmdLine (int argc, char **argv, Parameters parms)
- vector< const char * > args ()
- bool hasFlag (const char *name)
- bool hasFlag (string name)
- Flags getDefaultFlags (bool all=true)
- Flags getCurrentFlags (bool all=true)
- bool hasOption (const char *name)
- bool hasOption (string name)
- bool isOptionMultiple (const char *name)
- bool isOptionMultiple (string name)
- const char * getOption (const char *name)
- const char * getOption (string name)
- vector< const char * > getOptionValues (const char *name)
- vector< const char * > getOptionValues (string name)
- int getOptionNumValues (const char *name)
- int getOptionNumValues (string name)
- Options getDefaultOptions ()
- Options getCurrentOptions ()
- template<typename T >
  const T getOptionAs (T t, const char *name)
- template<typename T >
  const T getOptionAs (const char *name)
- template<typename T >
  const T getOptionAs (string name)
- template<typename T >
  const vector< T > getOptionValuesAs (string name)
- template<typename T >
  const vector< T > **getOptionValuesAs** (const char *name)
- bool hasDefinition (const char *def)

- bool hasDefinition (string def)
- bool isDefinitionMultiple (const char *name)
- bool isDefinitionMultiple (string name)
- const char * getDefinition (const char *name)
- const char * getDefinition (string name)
- Options getDefinitions ()
- int getDefinitionNumValues (const char *name)
- int getDefinitionNumValues (string name)
- vector< const char * > getDefinitionValues (const char *name)
- vector< const char * > getDefinitionValues (string name)

Métodos públicos estáticos

- static CmdLine getInstance (int argc, char **argv, Parameters parms=Parameters())
- static CmdLine getInstance (Parameters parms, int argc, char **argv)
- static void freeInstance ()

Métodos protegidos

- CmdLine (int argc, char **argv, Parameters parms, bool sensitive, bool strict)
- CmdLine (int argc, char **argv, Parameters parms, bool sensitive)

Métodos protegidos estáticos

- static CmdLine pGetInstance (int argc, char **argv, Parameters parms, bool sensitive=false, bool strict=false)

10.3.1. Descripción detallada

Definición en la línea 79 del archivo cmdline.hpp.

10.3.2. Documentación del constructor y destructor

10.3.2.1. CmdLine() [1/4]

```
CmdLine ( )  [inline]
```

Definición en la línea 81 del archivo cmdline.hpp.

10.3.2.2. CmdLine() [2/4]

```
CmdLine (
            int argc,
            char ** argv,
            Parameters parms )
```

Definición en la línea 17 del archivo cmdline.cpp.

10.3.2.3. ~CmdLine()

```
~CmdLine ( )
```

Definición en la línea 23 del archivo cmdline.cpp.

10.3.2.4. CmdLine() [3/4]

```
CmdLine (
            int argc,
            char ** argv,
            Parameters parms,
            bool sensitive,
            bool strict )  [protected]
```

Definición en la línea 20 del archivo cmdline.cpp.

10.3.2.5. CmdLine() [4/4]

```
CmdLine (
            int argc,
            char ** argv,
            Parameters parms,
            bool sensitive )  [inline], [protected]
```

Definición en la línea 148 del archivo cmdline.hpp.

10.3.3. Documentación de las funciones miembro

10.3.3.1. args()

```
vector< const char * > args ( )
```

Definición en la línea 42 del archivo cmdline.cpp.

10.3.3.2. freeInstance()

```
void freeInstance ( )  [static]
```

Definición en la línea 14 del archivo cmdline.cpp.

10.3.3.3. getCurrentFlags()

```
Flags getCurrentFlags (
            bool all = true )
```

Definición en la línea 47 del archivo cmdline.cpp.

10.3.3.4. getCurrentOptions()

```
Options getCurrentOptions ( )
```

Definición en la línea 61 del archivo cmdline.cpp.

10.3.3.5. getDefaultFlags()

```
Flags getDefaultFlags (
            bool all = true )
```

Definición en la línea 46 del archivo cmdline.cpp.

10.3.3.6. getDefaultOptions()

```
Options getDefaultOptions ( )
```

Definición en la línea 60 del archivo cmdline.cpp.

10.3.3.7. getDefinition() [1/2]

```
const char * getDefinition (
            const char * name )
```

Definición en la línea 76 del archivo cmdline.cpp.

10.3.3.8. getDefinition() [2/2]

```
const char * getDefinition (
            string name )
```

Definición en la línea 77 del archivo cmdline.cpp.

10.3.3.9. getDefinitionNumValues() [1/2]

```
int getDefinitionNumValues (
            const char * name )
```

Definición en la línea 78 del archivo cmdline.cpp.

10.3.3.10. getDefinitionNumValues() [2/2]

```
int getDefinitionNumValues (
            string name )
```

Definición en la línea 79 del archivo cmdline.cpp.

10.3.3.11. getDefinitions()

```
Options getDefinitions ( )
```

Definición en la línea 80 del archivo cmdline.cpp.

10.3.3.12. getDefinitionValues() [1/2]

```
vector< const char * > getDefinitionValues (
            const char * name )
```

Definición en la línea 83 del archivo cmdline.cpp.

10.3.3.13. getDefinitionValues() [2/2]

```
vector< const char * > getDefinitionValues (
            string name )
```

Definición en la línea 84 del archivo cmdline.cpp.

10.3.3.14. getInstance() [1/2]

```
CmdLine getInstance (
            int argc,
            char ** argv,
            Parameters parms = Parameters() )  [static]
```

Definición en la línea 31 del archivo cmdline.cpp.

10.3.3.15. getInstance() [2/2]

```
CmdLine getInstance (
            Parameters parms,
            int argc,
            char ** argv )  [static]
```

Definición en la línea 26 del archivo cmdline.cpp.

10.3.3.16. getOption() [1/2]

```
const char * getOption (
            const char * name )
```

Definición en la línea 53 del archivo cmdline.cpp.

10.3.3.17. getOption() [2/2]

```
const char * getOption (
            string name )  [inline]
```

Definición en la línea 102 del archivo cmdline.hpp.

10.3.3.18. getOptionAs() [1/3]

```
const T getOptionAs (
            const char * name )  [inline]
```

Definición en la línea 114 del archivo cmdline.hpp.

10.3.3.19. getOptionAs() [2/3]

```
const T getOptionAs (
            string name )  [inline]
```

Definición en la línea 119 del archivo cmdline.hpp.

10.3.3.20. getOptionAs() [3/3]

```
const T getOptionAs (
            T t,
            const char * name )  [inline]
```

Definición en la línea 111 del archivo cmdline.hpp.

10.3.3.21. getOptionNumValues() [1/2]

```
int getOptionNumValues (
            const char * name )
```

Definición en la línea 58 del archivo cmdline.cpp.

10.3.3.22. getOptionNumValues() [2/2]

```
int getOptionNumValues (
            string name )
```

Definición en la línea 59 del archivo cmdline.cpp.

10.3.3.23. getOptionValues() [1/2]

```
vector< const char * > getOptionValues (
            const char * name )
```

Definición en la línea 55 del archivo cmdline.cpp.

10.3.3.24. getOptionValues() [2/2]

```
vector< const char * > getOptionValues (
            string name )
```

Definición en la línea 56 del archivo cmdline.cpp.

10.3.3.25. getOptionValuesAs()

```
const vector< T > getOptionValuesAs (
            string name )  [inline]
```

Definición en la línea 120 del archivo cmdline.hpp.

10.3.3.26. hasDefinition() [1/2]

```
bool hasDefinition (
            const char * def )
```

Definición en la línea 72 del archivo cmdline.cpp.

10.3.3.27. hasDefinition() [2/2]

```
bool hasDefinition (
            string def )
```

Definición en la línea 73 del archivo cmdline.cpp.

10.3.3.28. hasFlag() [1/2]

```
bool hasFlag (
            const char * name )
```

Definición en la línea 44 del archivo cmdline.cpp.

10.3.3.29. hasFlag() [2/2]

```
bool hasFlag (
            string name )
```

Definición en la línea 45 del archivo cmdline.cpp.

10.3.3.30. hasOption() [1/2]

```
bool hasOption (
            const char * name )
```

Definición en la línea 49 del archivo cmdline.cpp.

10.3.3.31. hasOption() [2/2]

```
bool hasOption (
            string name )
```

Definición en la línea 50 del archivo cmdline.cpp.

10.3.3.32. isDefinitionMultiple() [1/2]

```
bool isDefinitionMultiple (
            const char * name )
```

Definición en la línea 74 del archivo cmdline.cpp.

10.3.3.33. isDefinitionMultiple() [2/2]

```
bool isDefinitionMultiple (
            string name )
```

Definición en la línea 75 del archivo cmdline.cpp.

10.3.3.34. isOptionMultiple() [1/2]

```
bool isOptionMultiple (
            const char * name )
```

Definición en la línea 51 del archivo cmdline.cpp.

10.3.3.35. isOptionMultiple() [2/2]

```
bool isOptionMultiple (
            string name )
```

Definición en la línea 52 del archivo cmdline.cpp.

10.3.3.36. pGetInstance()

```
CmdLine pGetInstance (
            int argc,
            char ** argv,
            Parameters parms,
            bool sensitive = false,
            bool strict = false )  [static], [protected]
```

Definición en la línea 36 del archivo cmdline.cpp.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- cmdline.hpp
- cmdline.cpp

10.4. Referencia de la Clase CmdLineDuplicateArgumentException

Diagrama de herencias de CmdLineDuplicateArgumentException

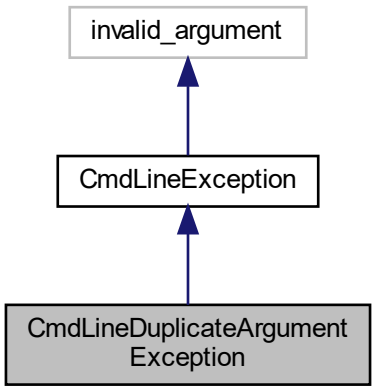
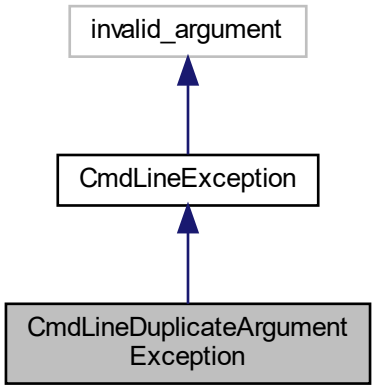


Diagrama de colaboración para CmdLineDuplicateArgumentException:



Métodos públicos

- CmdLineDuplicateArgumentException (const char ∗fmt,...)

10.4.1. Descripción detallada

Definición en la línea 47 del archivo cmdline_exceptions.hpp.

10.4.2. Documentación del constructor y destructor

10.4.2.1. CmdLineDuplicateArgumentException()

```
CmdLineDuplicateArgumentException (
            const char * fmt,
             ... )
```

Definición en la línea 40 del archivo cmdline_exceptions.cpp.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- cmdline_exceptions.hpp
- cmdline_exceptions.cpp

10.5. Referencia de la Clase CmdLineException

Diagrama de herencias de CmdLineException

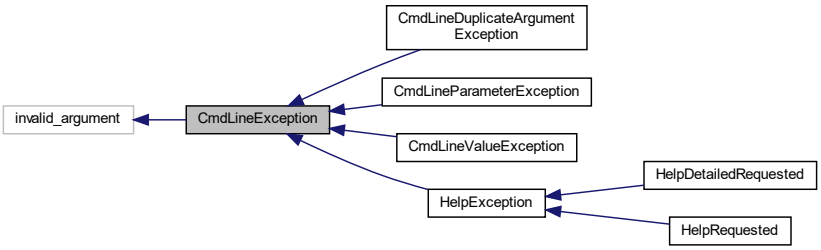
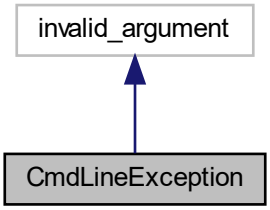


Diagrama de colaboración para CmdLineException:



Métodos públicos

- CmdLineException (const char ∗fmt,...)
- CmdLineException (char ∗txt)

10.5.1. Descripción detallada

Definición en la línea 27 del archivo cmdline_exceptions.hpp.

10.5.2. Documentación del constructor y destructor

10.5.2.1. CmdLineException() [1/3]

```
CmdLineException ( )  [inline]
```

Definición en la línea 29 del archivo cmdline_exceptions.hpp.

10.5.2.2. ~CmdLineException()

```
~CmdLineException ( )  [inline]
```

Definición en la línea 30 del archivo cmdline_exceptions.hpp.

10.5.2.3. CmdLineException() [2/3]

```
CmdLineException (
            const char * fmt,
             ... )
```

Definición en la línea 19 del archivo cmdline_exceptions.cpp.

10.5.2.4. CmdLineException() [3/3]

```
CmdLineException (
            char * txt )
```

Definición en la línea 18 del archivo cmdline_exceptions.cpp.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- cmdline_exceptions.hpp
- cmdline_exceptions.cpp

10.6. Referencia de la Clase CmdLineI

Diagrama de herencias de CmdLineI

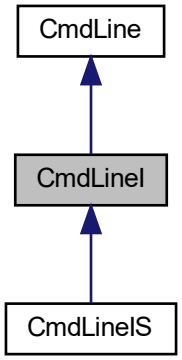
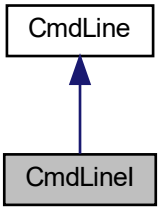


Diagrama de colaboración para CmdLineI:



Métodos públicos

- CmdLineI (int argc, char ∗∗argv, Parameters parms)
- CmdLineI (int argc, char ∗∗argv, cmdline::Parameters parms, bool strict)

Métodos públicos estáticos

- static CmdLine getInstance (int argc, char ∗∗argv, Parameters parms)

Otros miembros heredados

10.6.1. Descripción detallada

Definición en la línea 153 del archivo cmdline.hpp.

10.6.2. Documentación del constructor y destructor

10.6.2.1. CmdLinel() [1/2]

```
CmdLineI (
            int argc,
            char ** argv,
            cmdline::Parameters parms )
```

Definición en la línea 21 del archivo cmdline_flavors.cpp.

10.6.2.2. CmdLinel() [2/2]

```
CmdLineI (
            int argc,
            char ** argv,
            cmdline::Parameters parms,
            bool strict )
```

Definición en la línea 22 del archivo cmdline_flavors.cpp.

10.6.2.3. ~CmdLinel()

```
~CmdLineI ( )
```

Definición en la línea 18 del archivo cmdline_flavors.cpp.

10.6.3. Documentación de las funciones miembro

10.6.3.1. getInstance()

```
CmdLine getInstance (
            int argc,
            char ** argv,
            cmdline::Parameters parms )  [static]
```

Definición en la línea 23 del archivo cmdline_flavors.cpp.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- cmdline.hpp
- cmdline_flavors.cpp

10.7. Referencia de la Clase CmdLineInvalidTypeException

Diagrama de herencias de CmdLineInvalidTypeException

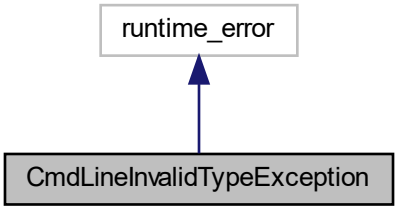
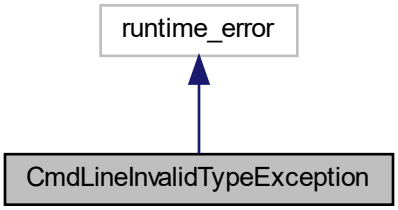


Diagrama de colaboración para CmdLineInvalidTypeException:



Métodos públicos

- CmdLineInvalidTypeException (const char ∗fmt,...)

10.7.1. Descripción detallada

Definición en la línea 21 del archivo cmdline_exceptions.hpp.

10.7.2. Documentación del constructor y destructor

10.7.2.1. CmdLineInvalidTypeException()

```
CmdLineInvalidTypeException (
            const char * fmt,
             ... )
```

Definición en la línea 54 del archivo cmdline_exceptions.cpp.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- cmdline_exceptions.hpp
- cmdline_exceptions.cpp

10.8. Referencia de la Clase CmdLineIS

Diagrama de herencias de CmdLineIS

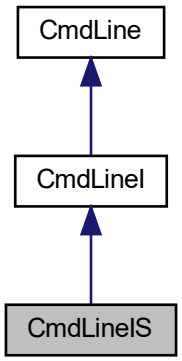
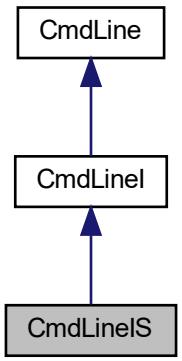


Diagrama de colaboración para CmdLineIS:



Métodos públicos

- CmdLineIS (int argc, char **argv, Parameters parms)

Métodos públicos estáticos

- static CmdLine getInstance (int argc, char **argv, Parameters parms)

Otros miembros heredados

10.8.1. Descripción detallada

Definición en la línea 165 del archivo cmdline.hpp.

10.8.2. Documentación del constructor y destructor

10.8.2.1. CmdLineIS()

```
CmdLineIS (
            int argc,
            char ** argv,
            cmdline::Parameters parms )
```

Definición en la línea 27 del archivo cmdline_flavors.cpp.

10.8.2.2. ~CmdLineIS()

```
~CmdLineIS ( )
```

Definición en la línea 34 del archivo cmdline_flavors.cpp.

10.8.3. Documentación de las funciones miembro

10.8.3.1. getInstance()

```
CmdLine getInstance (
            int argc,
            char ** argv,
            cmdline::Parameters parms )  [static]
```

Definición en la línea 31 del archivo cmdline_flavors.cpp.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- cmdline.hpp
- cmdline_flavors.cpp

10.9. Referencia de la Clase CmdLineNotFoundException

Diagrama de herencias de CmdLineNotFoundException

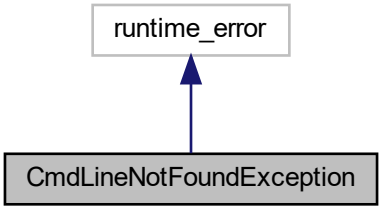
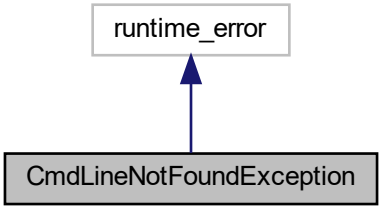


Diagrama de colaboración para CmdLineNotFoundException:



Métodos públicos

- CmdLineNotFoundException (const char ∗fmt,...)

10.9.1. Descripción detallada

Definición en la línea 16 del archivo cmdline_exceptions.hpp.

10.9.2. Documentación del constructor y destructor

10.9.2.1. CmdLineNotFoundException()

```
CmdLineNotFoundException (
            const char * fmt,
             ... )
```

Definición en la línea 47 del archivo cmdline_exceptions.cpp.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- cmdline_exceptions.hpp
- cmdline_exceptions.cpp

10.10. Referencia de la Clase CmdLineParameterException

Diagrama de herencias de CmdLineParameterException

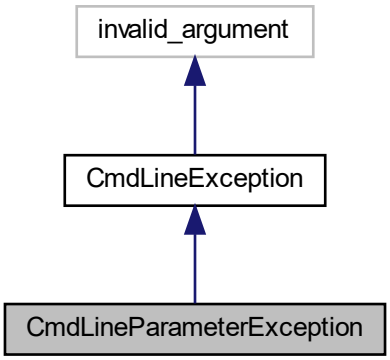
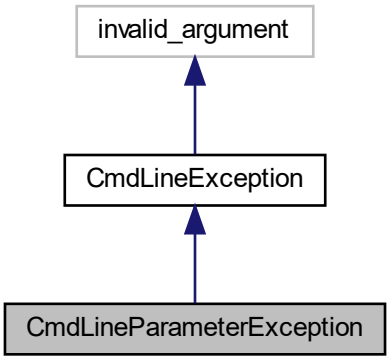


Diagrama de colaboración para CmdLineParameterException:



Métodos públicos

- CmdLineParameterException (const char ∗fmt,...)

10.10.1. Descripción detallada

Definición en la línea 37 del archivo cmdline_exceptions.hpp.

10.10.2. Documentación del constructor y destructor

10.10.2.1. CmdLineParameterException()

```
CmdLineParameterException (
            const char * fmt,
             ... )
```

Definición en la línea 26 del archivo cmdline_exceptions.cpp.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- cmdline_exceptions.hpp
- cmdline_exceptions.cpp

10.11. Referencia de la Clase CmdLineS

Diagrama de herencias de CmdLineS

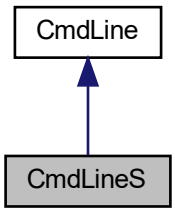
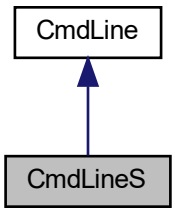


Diagrama de colaboración para CmdLineS:



Métodos públicos

- CmdLineS (int argc, char ∗∗argv, Parameters parms)

Métodos públicos estáticos

- static CmdLine getInstance (int argc, char ∗∗argv, Parameters parms)

Otros miembros heredados

10.11.1. Descripción detallada

Definición en la línea 160 del archivo cmdline.hpp.

10.11.2. Documentación del constructor y destructor

10.11.2.1. CmdLineS()

```
CmdLineS (
            int argc,
            char ** argv,
            cmdline::Parameters parms )
```

Definición en la línea 26 del archivo cmdline_flavors.cpp.

10.11.3. Documentación de las funciones miembro

10.11.3.1. getInstance()

```
CmdLine getInstance (
            int argc,
            char ** argv,
            cmdline::Parameters parms )  [static]
```

Definición en la línea 28 del archivo cmdline_flavors.cpp.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- cmdline.hpp
- cmdline_flavors.cpp

10.12. Referencia de la Clase CmdLineValueException

Diagrama de herencias de CmdLineValueException

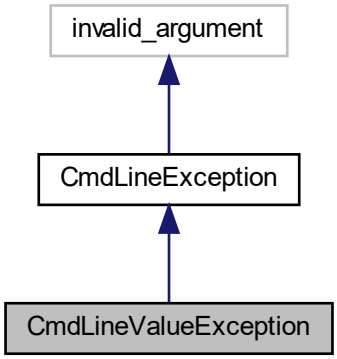
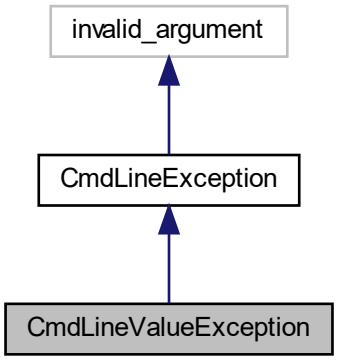


Diagrama de colaboración para CmdLineValueException:



Métodos públicos

- CmdLineValueException (const char ∗fmt,...)

10.12.1. Descripción detallada

Definición en la línea 42 del archivo cmdline_exceptions.hpp.

10.12.2. Documentación del constructor y destructor

10.12.2.1. CmdLineValueException()

```
CmdLineValueException (
            const char * fmt,
             ... )
```

Definición en la línea 33 del archivo cmdline_exceptions.cpp.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- cmdline_exceptions.hpp
- cmdline_exceptions.cpp

10.13. Referencia de la Clase CommandLine

Métodos públicos

- CommandLine (int argc, char **argv, Parameters parms, bool sensitive, bool strict)
- vector< const char * > getArgs ()
- bool hasFlag (const char *flag)
- Flags getDefaultFlags (bool all)
- Flags getCurrentFlags (bool all)
- int getOptionNumValues (const char *name)
- bool hasOption (const char *name)
- bool isOptionMultiple (const char *name)
- Options getDefaultOptions ()
- Options getCurrentOptions ()
- const char * getOption (const char *name)
- vector< const char * > getOptionValues (const char *name)
- bool hasDefinition (const char *def)
- bool isDefinitionMultiple (const char *name)
- int getDefinitionNumValues (const char *name)
- const char * getDefinition (const char *name)
- vector< const char * > getDefinitionValues (const char *name)
- Options getDefinitions ()

10.13.1. Descripción detallada

Definición en la línea 14 del archivo commandline.hpp.

10.13.2. Documentación del constructor y destructor

10.13.2.1. CommandLine()

```
CommandLine (
            int argc,
            char ** argv,
            Parameters parms,
            bool sensitive,
            bool strict )
```

Definición en la línea 18 del archivo commandline.cpp.

10.13.2.2. ~CommandLine()

```
~CommandLine ( )
```

Definición en la línea 27 del archivo commandline.cpp.

10.13.3. Documentación de las funciones miembro

10.13.3.1. getArgs()

```
vector< const char * > getArgs ( )
```

Definición en la línea 30 del archivo commandline.cpp.

10.13.3.2. getCurrentFlags()

```
Flags getCurrentFlags (
            bool all )
```

Definición en la línea 40 del archivo commandline.cpp.

10.13.3.3. getCurrentOptions()

```
Options getCurrentOptions ( )
```

Definición en la línea 66 del archivo commandline.cpp.

10.13.3.4. getDefaultFlags()

```
Flags getDefaultFlags (
            bool all )
```

Definición en la línea 37 del archivo commandline.cpp.

10.13.3.5. getDefaultOptions()

```
Options getDefaultOptions ( )
```

Definición en la línea 63 del archivo commandline.cpp.

10.13.3.6. getDefinition()

```
const char * getDefinition (
            const char * name )  [inline]
```

Definición en la línea 39 del archivo commandline.hpp.

10.13.3.7. getDefinitionNumValues()

```
int getDefinitionNumValues (
            const char * name )
```

Definición en la línea 81 del archivo commandline.cpp.

10.13.3.8. getDefinitions()

```
Options getDefinitions ( )
```

Definición en la línea 85 del archivo commandline.cpp.

10.13.3.9. getDefinitionValues()

```
vector< const char * > getDefinitionValues (
            const char * name )
```

Definición en la línea 77 del archivo commandline.cpp.

10.13.3.10. getOption()

```
const char * getOption (
            const char * name )
```

Definición en la línea 59 del archivo commandline.cpp.

10.13.3.11. getOptionNumValues()

```
int getOptionNumValues (
            const char * name )
```

Definición en la línea 55 del archivo commandline.cpp.

10.13.3.12. getOptionValues()

```
vector< const char * > getOptionValues (
            const char * name )
```

Definición en la línea 51 del archivo commandline.cpp.

10.13.3.13. hasDefinition()

```
bool hasDefinition (
            const char * def )
```

Definición en la línea 69 del archivo commandline.cpp.

10.13.3.14. hasFlag()

```
bool hasFlag (
            const char * flag )
```

Definición en la línea 33 del archivo commandline.cpp.

10.13.3.15. hasOption()

```
bool hasOption (
            const char * name )
```

Definición en la línea 43 del archivo commandline.cpp.

10.13.3.16. isDefinitionMultiple()

```
bool isDefinitionMultiple (
            const char * name )
```

Definición en la línea 73 del archivo commandline.cpp.

10.13.3.17. isOptionMultiple()

```
bool isOptionMultiple (
            const char * name )
```

Definición en la línea 47 del archivo commandline.cpp.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- commandline.hpp
- commandline.cpp

10.14. Referencia de la Clase Define

Diagrama de herencias de Define

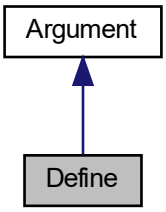
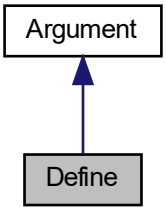


Diagrama de colaboración para Define:



Métodos públicos

- Define (const char ∗name, const char ∗value)
- Define (const char ∗name)
- void initValues (vector< string > values)

Otros miembros heredados

10.14.1. Descripción detallada

Definición en la línea 8 del archivo defines.hpp.

10.14.2. Documentación del constructor y destructor

10.14.2.1. Define() [1/2]

```
Define (
            const char * name,
            const char * value )  [inline]
```

Definición en la línea 11 del archivo defines.hpp.

10.14.2.2. Define() [2/2]

```
Define (
            const char * name )  [inline]
```

Definición en la línea 12 del archivo defines.hpp.

10.14.2.3. ~Define()

```
~Define ( )  [inline]
```

Definición en la línea 16 del archivo defines.hpp.

10.14.3. Documentación de las funciones miembro

10.14.3.1. initValues()

```
void initValues (
            vector< string > values )  [inline]
```

Definición en la línea 19 del archivo defines.hpp.

La documentación para esta clase fue generada a partir del siguiente fichero:

- defines.hpp

10.15. Referencia de la Estructura Flag

Atributos públicos

- const char * name
- logical value

10.15.1. Descripción detallada

Definición en la línea 31 del archivo types.h.

10.15.2. Documentación de los datos miembro

10.15.2.1. name

```
const char* name
```

Definición en la línea 32 del archivo types.h.

10.15.2.2. value

```
logical value
```

Definición en la línea 33 del archivo types.h.

La documentación para esta estructura fue generada a partir del siguiente fichero:

- types.h

10.16. Referencia de la Clase Garbage

10.16.1. Descripción detallada

Definición en la línea 3 del archivo garbage.hpp.

La documentación para esta clase fue generada a partir del siguiente fichero:

- garbage.hpp

10.17. Referencia de la Clase Group

Diagrama de herencias de Group

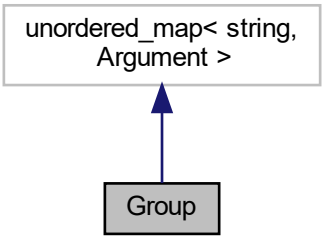
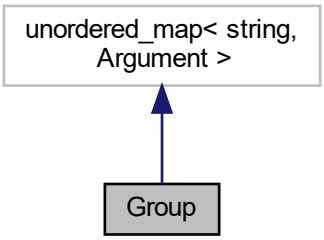


Diagrama de colaboración para Group:



Métodos públicos

- Group & add (const char *name, Argument *arg)
- Group & add (string name, Argument *arg)
- Group & add (Argument *arg)
- Argument * find (string what)
- Argument * operator[ ] (string what)

10.17.1. Descripción detallada

Definición en la línea 10 del archivo group.hpp.

10.17.2. Documentación de las funciones miembro

10.17.2.1. add() [1/3]

```
Group & add (
            Argument * arg )  [inline]
```

Definición en la línea 21 del archivo group.hpp.

10.17.2.2. add() [2/3]

```
Group & add (
            const char * name,
            Argument * arg )  [inline]
```

Definición en la línea 12 del archivo group.hpp.

10.17.2.3. add() [3/3]

```
Group & add (
            string name,
            Argument * arg )  [inline]
```

Definición en la línea 16 del archivo group.hpp.

10.17.2.4. find()

```
Argument * find (
            string what )  [inline]
```

Definición en la línea 26 del archivo group.hpp.

10.17.2.5. operator[]()

```
Argument * operator[] (
            string what )  [inline]
```

Definición en la línea 34 del archivo group.hpp.

La documentación para esta clase fue generada a partir del siguiente fichero:

- group.hpp

10.18. Referencia de la Clase HelpDetailedRequested

Diagrama de herencias de HelpDetailedRequested

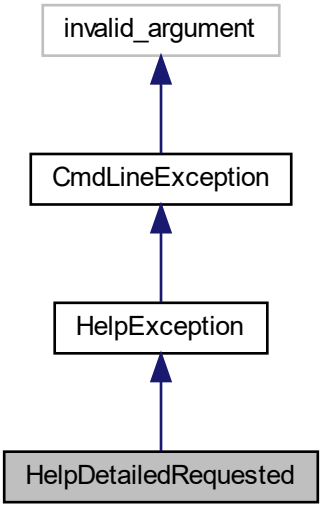
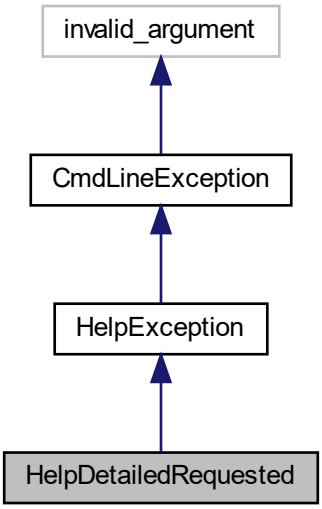


Diagrama de colaboración para HelpDetailedRequested:



Otros miembros heredados

10.18.1. Descripción detallada

Definición en la línea 63 del archivo cmdline_exceptions.hpp.

10.18.2. Documentación del constructor y destructor

10.18.2.1. HelpDetailedRequested()

```
HelpDetailedRequested ( )  [inline]
```

Definición en la línea 65 del archivo cmdline_exceptions.hpp.

La documentación para esta clase fue generada a partir del siguiente fichero:

- cmdline_exceptions.hpp

10.19. Referencia de la Clase HelpException

Diagrama de herencias de HelpException

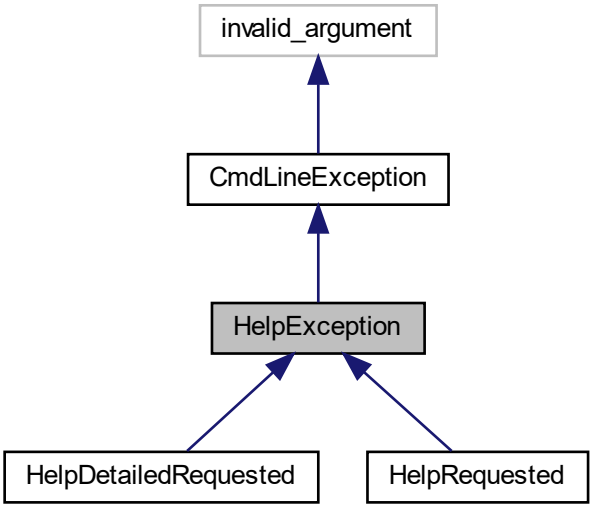
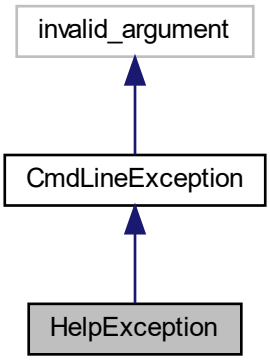


Diagrama de colaboración para HelpException:



Métodos protegidos

- HelpException (const char *txt)

Otros miembros heredados

10.19.1. Descripción detallada

Definición en la línea 55 del archivo cmdline_exceptions.hpp.

10.19.2. Documentación del constructor y destructor

10.19.2.1. HelpException()

```
HelpException (
            const char * txt )  [inline], [protected]
```

Definición en la línea 57 del archivo cmdline_exceptions.hpp.

La documentación para esta clase fue generada a partir del siguiente fichero:

- cmdline_exceptions.hpp

10.20. Referencia de la Clase HelpRequested

Diagrama de herencias de HelpRequested

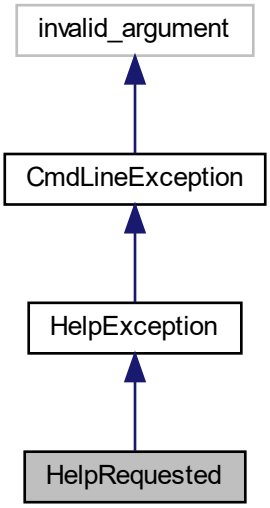
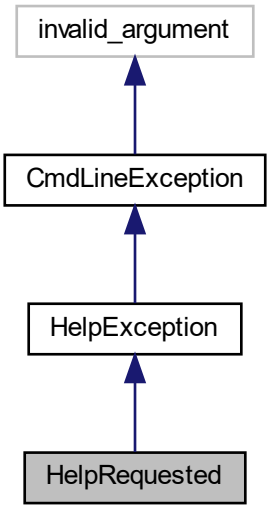


Diagrama de colaboración para HelpRequested:



Otros miembros heredados

10.20.1. Descripción detallada

Definición en la línea 59 del archivo cmdline_exceptions.hpp.

10.20.2. Documentación del constructor y destructor

10.20.2.1. HelpRequested()

```
HelpRequested ( )  [inline]
```

Definición en la línea 61 del archivo cmdline_exceptions.hpp.

La documentación para esta clase fue generada a partir del siguiente fichero:

- cmdline_exceptions.hpp

10.21. Referencia de la Estructura Parameter

Atributos públicos

- const char * name
- const char ** values
- int size

10.21.1. Descripción detallada

Definición en la línea 26 del archivo types.h.

10.21.2. Documentación de los datos miembro

10.21.2.1. name

```
const char* name
```

Definición en la línea 27 del archivo types.h.

10.21.2.2. size

```
int size
```

Definición en la línea 29 del archivo types.h.

10.21.2.3. values

```
const char** values
```

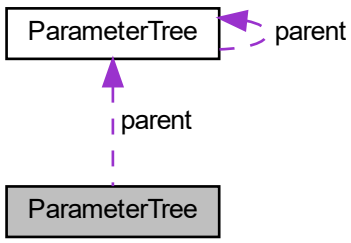
Definición en la línea 28 del archivo types.h.

La documentación para esta estructura fue generada a partir del siguiente fichero:

- types.h

10.22. Referencia de la Clase ParameterTree

Diagrama de colaboración para ParameterTree:



Métodos públicos

- **ParameterTree** (ParameterTree &)=delete
- **ParameterTree** (ParameterTree &&)=delete
- ParameterTree (const char *parm, ParameterTree *parent=nullptr)
- ParameterTree & addBranch (ParameterTree *tree)
- std::string **getName** (const std::string parm, int pos=1)
- char * getWord ()
- char * getReversedWord ()
- ParameterTree * getNext ()
- ParameterTree * getChild (char letter)
- ParameterTree * setParent (ParameterTree *parent)
- ParameterTree & addChild (ParameterTree *child)
- bool hasChild (char letter)
- size_t numChildren ()

Atributos públicos

- char letter
- int branchs = 0
- ParameterTree * parent = nullptr

10.22.1. Descripción detallada

Definición en la línea 7 del archivo parameter_tree.hpp.

10.22.2. Documentación del constructor y destructor

10.22.2.1. ParameterTree()

```
ParameterTree (
            const char * parm,
            ParameterTree * parent = nullptr )  [inline]
```

Definición en la línea 15 del archivo parameter_tree.hpp.

10.22.3. Documentación de las funciones miembro

10.22.3.1. addBranch()

```
ParameterTree & addBranch (
            ParameterTree * tree )
```

Definición en la línea 12 del archivo parameter_tree.cpp.

10.22.3.2. addChild()

```
ParameterTree & addChild (
            ParameterTree * child )
```

Definición en la línea 7 del archivo parameter_tree.cpp.

10.22.3.3. getChild()

```
ParameterTree * getChild (
            char letter )
```

Definición en la línea 67 del archivo parameter_tree.cpp.

10.22.3.4. getNext()

```
ParameterTree * getNext ( )
```

Definición en la línea 59 del archivo parameter_tree.cpp.

10.22.3.5. getReversedWord()

```
char * getReversedWord ( )
```

Definición en la línea 37 del archivo parameter_tree.cpp.

10.22.3.6. getWord()

```
char * getWord ( )
```

Definición en la línea 17 del archivo parameter_tree.cpp.

10.22.3.7. hasChild()

```
bool hasChild (
            char letter )
```

Definición en la línea 63 del archivo parameter_tree.cpp.

10.22.3.8. numChildren()

```
size_t numChildren ( )  [inline]
```

Definición en la línea 25 del archivo parameter_tree.hpp.

10.22.3.9. setParent()

```
ParameterTree * setParent (
            ParameterTree * parent )  [inline]
```

Definición en la línea 22 del archivo parameter_tree.hpp.

10.22.4. Documentación de los datos miembro

10.22.4.1. branchs

```
int branchs = 0
```

Definición en la línea 10 del archivo parameter_tree.hpp.

10.22.4.2. letter

```
char letter
```

Definición en la línea 9 del archivo parameter_tree.hpp.

10.22.4.3. parent

```
ParameterTree* parent = nullptr
```

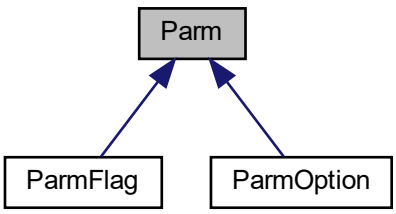
Definición en la línea 11 del archivo parameter_tree.hpp.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- parameter_tree.hpp
- parameter_tree.cpp

10.23. Referencia de la Clase Parm

Diagrama de herencias de Parm



Métodos públicos

- Parm (const char ∗name)
- Parm (const char ∗name, const char ∗value)
- Parm (const char ∗name, const char ∗value, Type type, bool multiple=false)
- Parm (const char ∗name, bool value)
- bool instanceOfFlag ()
- bool instanceOfOption ()

Atributos públicos

- const char ∗ name
- Type type = Type::STRING
- char ∗ value
- bool multiple = false

10.23.1. Descripción detallada

Definición en la línea 31 del archivo cmdline.hpp.

10.23.2. Documentación del constructor y destructor

10.23.2.1. Parm() [1/5]

```
Parm ( )  [inline]
```

Definición en la línea 37 del archivo cmdline.hpp.

10.23.2.2. Parm() [2/5]

```
Parm (
            const char * name )
```

Definición en la línea 5 del archivo parm.cpp.

10.23.2.3. Parm() [3/5]

```
Parm (
            const char * name,
            const char * value )
```

Definición en la línea 12 del archivo parm.cpp.

10.23.2.4. Parm() [4/5]

```
Parm (
            const char * name,
            const char * value,
            Type type,
            bool multiple = false )
```

Definición en la línea 18 del archivo parm.cpp.

10.23.2.5. Parm() [5/5]

```
Parm (
            const char * name,
            bool value )
```

Definición en la línea 25 del archivo parm.cpp.

10.23.3. Documentación de las funciones miembro

10.23.3.1. instanceOfFlag()

```
bool instanceOfFlag ( )
```

Definición en la línea 32 del archivo parm.cpp.

10.23.3.2. instanceOfOption()

```
bool instanceOfOption ( )
```

Definición en la línea 33 del archivo parm.cpp.

10.23.4. Documentación de los datos miembro

10.23.4.1. multiple

```
bool multiple = false
```

Definición en la línea 36 del archivo cmdline.hpp.

10.23.4.2. name

```
const char* name
```

Definición en la línea 33 del archivo cmdline.hpp.

10.23.4.3. type

```
Type type = Type::STRING
```

Definición en la línea 34 del archivo cmdline.hpp.

10.23.4.4. value

```
char* value
```

Definición en la línea 35 del archivo cmdline.hpp.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- cmdline.hpp
- parm.cpp

10.24. Referencia de la Estructura ParmDef

Atributos públicos

- const char * name
- Type type
- char * value
- logical multiple

10.24.1. Descripción detallada

Definición en la línea 19 del archivo types.h.

10.24.2. Documentación de los datos miembro

10.24.2.1. multiple

```
logical multiple
```

Definición en la línea 23 del archivo types.h.

10.24.2.2. name

```
const char* name
```

Definición en la línea 20 del archivo types.h.

10.24.2.3. type

```
Type type
```

Definición en la línea 21 del archivo types.h.

10.24.2.4. value

```
char* value
```

Definición en la línea 22 del archivo types.h.

La documentación para esta estructura fue generada a partir del siguiente fichero:

- types.h

10.25. Referencia de la Clase ParmFlag

`#include <cmdline.hpp>`

Diagrama de herencias de ParmFlag

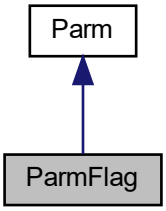
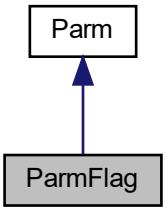


Diagrama de colaboración para ParmFlag:



Métodos públicos

- ParmFlag (const char ∗name)
- ParmFlag (const char ∗name, bool active)

Otros miembros heredados

10.25.1. Descripción detallada

Utility subclass for defining Flags

Parm(const char∗ name) : Name,false Parm(const char∗ name, const char∗ value): name/value (false/true)

Definición en la línea 56 del archivo cmdline.hpp.

10.25.2. Documentación del constructor y destructor

10.25.2.1. ParmFlag() [1/2]

```
ParmFlag (
            const char * name )  [inline]
```

Definición en la línea 59 del archivo cmdline.hpp.

10.25.2.2. ParmFlag() [2/2]

```
ParmFlag (
            const char * name,
            bool active )  [inline]
```

Definición en la línea 60 del archivo cmdline.hpp.

La documentación para esta clase fue generada a partir del siguiente fichero:

- cmdline.hpp

10.26. Referencia de la Clase ParmOption

```
#include <cmdline.hpp>
```

Diagrama de herencias de ParmOption

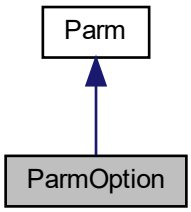
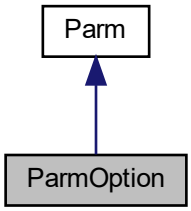


Diagrama de colaboración para ParmOption:



Métodos públicos

- ParmOption (const char ∗name, const char ∗value)
- ParmOption (const char ∗name, Type type)
- ParmOption (const char ∗name, const char ∗value, bool multiple)
- ParmOption (const char ∗name, const char ∗value, Type type, bool multiple=false)

Otros miembros heredados

10.26.1. Descripción detallada

Utility subclass for defining Options

Parm(const char∗ name, const char∗ value): name/value (false/true)

Definición en la línea 69 del archivo cmdline.hpp.

10.26.2. Documentación del constructor y destructor

10.26.2.1. ParmOption() [1/4]

```
ParmOption (
            const char * name,
            const char * value )  [inline]
```

Definición en la línea 72 del archivo cmdline.hpp.

10.26.2.2. ParmOption() [2/4]

```
ParmOption (
            const char * name,
            Type type )  [inline]
```

Definición en la línea 73 del archivo cmdline.hpp.

10.26.2.3. ParmOption() [3/4]

```
ParmOption (
            const char * name,
            const char * value,
            bool multiple )  [inline]
```

Definición en la línea 74 del archivo cmdline.hpp.

10.26.2.4. ParmOption() [4/4]

```
ParmOption (
            const char * name,
            const char * value,
            Type type,
            bool multiple = false )  [inline]
```

Definición en la línea 75 del archivo cmdline.hpp.

La documentación para esta clase fue generada a partir del siguiente fichero:

- cmdline.hpp