customCommands

1.0.0

Generated by Doxygen 1.13.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

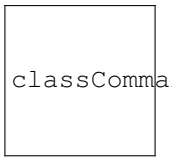
Chapter 4

Class Documentation

4.1 Command Class Reference

`#include <Command.h>`

Inheritance diagram for Command:



Public Member Functions

- Command (const std::string &name, const std::vector< std::string > &aliases, size_t nbOfArguments, const std::string &description, bool isMandatory, bool activateImmediately)
- virtual ~Command ()=default
- virtual void setArguments (const std::vector< std::string > &args)=0
- virtual void execute ()=0
- const std::string & name () const
- std::string description () const
- const std::vector< std::string > & aliases () const
- bool isMandatoryCommand () const
- bool executesNow () const
- std::size_t nbArguments () const

Protected Attributes

- std::string c_name
- std::vector< std::string > c_aliases
- size_t c_nbOfArguments
- std::string c_description
- bool c_isMandatory
- bool c_activateImmediately

4.1.1 Detailed Description

Class representing a Command of the parser

Authors

Paul Caillé, Oregan Hardy

Version

1.0.0

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Command()

```
Command::Command (
            const std::string & name,
            const std::vector< std::string > & aliases,
            size_t nbOfArguments,
            const std::string & description,
            bool isMandatory,
            bool activateImmediately)
```

Constructor of a Command

Parameters

name	name of the Command
aliases	vector of string contained in c_aliases
nbOfArguments	number of argument taken by the Command
description	description of the class, displayed by help
isMandatory	if the command is mandatory
activateImmediately	if the command can be executed directly when parsed

4.1.2.2 ~Command()

```
virtual Command::~Command ()  [virtual], [default]
```

Destructor of a Command, used by the desctructor of Parsing

4.1.3 Member Function Documentation

4.1.3.1 aliases()

```
const std::vector< std::string > & Command::aliases () const
```

Getter for c_aliases

Returns

std::vector<std::string>

4.1.3.2 description()

```
std::string Command::description () const
```

Getter for c_destruction

Returns

std::string

4.1.3.3 execute()

```
virtual void Command::execute ()  [pure virtual]
```

Method launched by Parser, main execution of the command

Implemented in HelpCommand.

4.1.3.4 executesNow()

```
bool Command::executesNow () const
```

Getter for c_activateImmediately

Returns

bool

4.1.3.5 isMandatoryCommand()

```
bool Command::isMandatoryCommand () const
```

Getter for c_isMandatory

Returns

bool

4.1.3.6 name()

```
const std::string & Command::name () const
```

Getter for c_name

Returns

std::string

4.1.3.7 nbArguments()

```
std::size_t Command::nbArguments () const
```

Getter for c_nbOfArguments

Returns

std::size_t

4.1.3.8 setArguments()

```
virtual void Command::setArguments (
            const std::vector< std::string > & args)  [pure virtual]
```

Set arguments parsed by Parser in the Command

Parameters

args	

Implemented in HelpCommand.

4.1.4 Member Data Documentation

4.1.4.1 c_activateImmediately

```
bool Command::c_activateImmediately  [protected]
```

If the command can be executed immediately after being parsed Command using Targets object must be at false

4.1.4.2 c_aliases

```
std::vector<std::string> Command::c_aliases  [protected]
```

Vector of the aliases for the command, must start with "-" or "--" for long name

4.1.4.3 c_description

```
std::string Command::c_description  [protected]
```

Description of the command displayed by help

4.1.4.4 c_isMandatory

```
bool Command::c_isMandatory  [protected]
```

Boolean if the command is mandatory or not

4.1.4.5 c_name

```
std::string Command::c_name  [protected]
```

Name of the command

4.1.4.6 c_nbOfArguments

```
size_t Command::c_nbOfArguments  [protected]
```

Number of argument the command takes

The documentation for this class was generated from the following files:

- commands/Command.h
- commands/Command.cpp

4.2 HelpCommand Class Reference

```
#include <Command.h>
```

Inheritance diagram for HelpCommand:

Public Member Functions

- HelpCommand (const Parsing &parser)
- void setArguments (const std::vector< std::string > &args) override
- void execute () override

Public Member Functions inherited from Command

- Command (const std::string &name, const std::vector< std::string > &aliases, size_t nbOfArguments, const std::string &description, bool isMandatory, bool activateImmediately)
- virtual ~Command ()=default
- const std::string & name () const
- std::string description () const
- const std::vector< std::string > & aliases () const
- bool isMandatoryCommand () const
- bool executesNow () const
- std::size_t nbArguments () const

Additional Inherited Members

Protected Attributes inherited from Command

- std::string c_name
- std::vector< std::string > c_aliases
- size_t c_nbOfArguments
- std::string c_description
- bool c_isMandatory
- bool c_activateImmediately

4.2.1 Detailed Description

Class representing Help Command

Authors

Paul Caillé, Oregan Hardy

Version

1.0.0

4.2.2 Constructor & Destructor Documentation

4.2.2.1 HelpCommand()

```
HelpCommand::HelpCommand (
            const Parsing & parser)  [explicit]
```

Constructor for HelpCommand

Parameters

parser	

4.2.3 Member Function Documentation

4.2.3.1 execute()

```
void HelpCommand::execute ()  [override], [virtual]
```

Display the help message of the commands inside of Parser

Implements Command.

4.2.3.2 setArguments()

```
void HelpCommand::setArguments (
            const std::vector< std::string > & args)  [override], [virtual]
```

Implementation of setArguments method

Exceptions

std::runtime_error	if the vector is not empty

Parameters

args	std::vector<std::string>>

Implements Command.

The documentation for this class was generated from the following files:

- commands/Command.h
- commands/Command.cpp

4.3 Parsing Class Reference

`#include <Parsing.h>`

Public Member Functions

- Parsing (Targets &targets)
- void addCommand (Command *command)
- void parseInput (int argc, const char *argv[]) const
- std::vector< std::string > allDescriptions () const
- bool hasCommand (const std::string &name) const
- const Targets & targets () const
- ~Parsing ()

Public Attributes

- std::vector< Command * > p_commandsToParse
- std::string exename

4.3.1 Detailed Description

Class representing the command parser

Authors

Paul Caillé, Oregan Hardy

Version

1.0.0

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Parsing()

```
Parsing::Parsing (
            Targets & targets)  [explicit]
```

Constructor for a parser

Parameters

targets	

4.3.2.2 ~Parsing()

```
Parsing::~Parsing ()
```

Destructor for Parsing

4.3.3 Member Function Documentation

4.3.3.1 addCommand()

```
void Parsing::addCommand (
            Command * command)
```

Add a command to the parser

Parameters

command	

4.3.3.2 allDescriptions()

```
std::vector< std::string > Parsing::allDescriptions () const
```

Return the description of all the commands added to the parser

Returns

std::vector<std::string>

4.3.3.3 hasCommand()

```
bool Parsing::hasCommand (
            const std::string & name) const
```

Check if the given command name is present in p_commandToParse

Parameters

name	

Returns

boolean, if the command exist in the parser

4.3.3.4 parseInput()

```
void Parsing::parseInput (
            int argc,
            const char * argv[]) const
```

Main method to parse command

Parameters

argc	
argv	

4.3.3.5 targets()

```
const Targets & Parsing::targets () const
```

Getter for p_targets

Returns

Target

4.3.4 Member Data Documentation

4.3.4.1 exename

```
std::string Parsing::exename  [mutable]
```

Name of the executable

4.3.4.2 p_commandsToParse

```
std::vector<Command *> Parsing::p_commandsToParse
```

Vector of all commands added to the parser

The documentation for this class was generated from the following files:

- parsing/Parsing.h
- parsing/Parsing.cpp

4.4 Targets Class Reference

```
#include <Targets.h>
```

Public Types

- using const_iterator = std::vector<std::string>::const_iterator

Public Member Functions

- Targets (bool canBeEmpty, const std::string &description)
- bool canBeEmpty () const
- const std::vector< std::string > & targets () const
- void addTarget (const std::string &targ)
- bool empty () const
- const_iterator begin () const
- const_iterator end () const

Friends

- std::ostream & operator<< (std::ostream &os, const Targets &targets)

4.4.1 Detailed Description

Class representing the targets of the program, such as files

Authors

Paul Caillé, Oregan Hardy

Version

1.0.0

4.4.2 Member Typedef Documentation

4.4.2.1 const_iterator

```
using Targets::const_iterator = std::vector<std::string>::const_iterator
```

4.4.3 Constructor & Destructor Documentation

4.4.3.1 Targets()

```
Targets::Targets (
            bool canBeEmpty,
            const std::string & description)
```

Constructor for Targets

Parameters

canBeEmpty	
description	

4.4.4 Member Function Documentation

4.4.4.1 addTarget()

```
void Targets::addTarget (
            const std::string & targ)
```

Add a target to the targets vector

Parameters

targ	the target to add

4.4.4.2 begin()

```
Targets::const_iterator Targets::begin () const
```

4.4.4.3 canBeEmpty()

```
bool Targets::canBeEmpty () const
```

Getter for t_canBeEmpty

Returns

boolean

4.4.4.4 empty()

```
bool Targets::empty () const
```

Indicate if the list is empty

Returns

boolean

4.4.4.5 end()

```
Targets::const_iterator Targets::end () const
```

4.4.4.6 targets()

```
const std::vector< std::string > & Targets::targets () const
```

Getter for t_targs

Returns

std::vector<std::string>

4.4.5 Friends And Related Symbol Documentation

4.4.5.1 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const Targets & targets)  [friend]
```

The documentation for this class was generated from the following files:

- target/Targets.h
- target/Targets.cpp

Chapter 5

File Documentation

5.1 commands/Command.cpp File Reference

```
#include "Command.h"
#include "../parsing/Parsing.h"
#include <iostream>
```

5.2 commands/Command.h File Reference

```
#include <string>
#include <vector>
```

Classes

- class Command
- class HelpCommand

5.3 Command.h

Go to the documentation of this file.

```
00001 //
00002 // Created on 29/11/2024.
00003 // CAILLE
00004 // PAUL
00005 // M1 - CL
00006 //
00007
00008 #ifndef COMMAND_H
00009 #define COMMAND_H
00010 #include <string>
00011 #include <vector>
00012
00013
00014 class Parsing;
00015
00021 class Command {
00022 protected:
```

```
00026     std::string c_name;
00030     std::vector<std::string> c_aliases;
00034     size_t c_nbOfArguments;
00038     std::string c_description;
00042     bool c_isMandatory;
00047     bool c_activateImmediately;
00048
00049 public:
00059     Command(const std::string &name, const std::vector<std::string> &aliases, size_t nbOfArguments,
00060             const std::string &description, bool isMandatory, bool activateImmediately);
00061
00065     virtual ~Command() = default;
00066
00071     virtual void setArguments(const std::vector<std::string> &args) = 0;
00072
00076     virtual void execute() = 0;
00077
00082     const std::string &name() const;
00083
00088     std::string description() const;
00089
00094     const std::vector<std::string> &aliases() const;
00095
00100     bool isMandatoryCommand() const;
00101
00106     bool executesNow() const;
00107
00112     std::size_t nbArguments() const;
00113 };
00114
00120 class HelpCommand final : public Command {
00124  const Parsing &parser;
00125
00126 public:
00131  explicit HelpCommand(const Parsing &parser);
00132
00138  void setArguments(const std::vector<std::string> &args) override;
00139
00143  void execute() override;
00144 };
00145
00146 #endif //COMMAND_H
```

5.4 parsing/Parsing.cpp File Reference

```
#include "Parsing.h"
#include <iostream>
#include <ostream>
```

5.5 parsing/Parsing.h File Reference

```
#include <vector>
#include "../commands/Command.h"
#include "../target/Targets.h"
```

Classes

- class Parsing

5.6 Parsing.h

Go to the documentation of this file.

```
00001 //
00002 // Created on 29/11/2024.
00003 // CAILLE
00004 // PAUL
00005 // M1 - CL
00006 //
00007
00008 #ifndef PARSING_H
00009 #define PARSING_H
00010 #include <vector>
00011
00012 #include "../commands/Command.h"
00013 #include "../target/Targets.h"
00014
00020 class Parsing {
00024     Targets &p_targets;
00025
00031     Command *findCommand(const std::string &name) const;
00032
00037     void executeAll() const;
00038
00044     bool checkMissingMandatory(const std::vector<std::string> &inputParts) const;
00045
00046 public:
00050     std::vector<Command *> p_commandsToParse;
00051
00055     mutable std::string exename ;
00056
00061     explicit Parsing(Targets &targets);
00062
00067     void addCommand(Command *command);
00068
00074     void parseInput(int argc, const char *argv[]) const;
00075
00080     std::vector<std::string> allDescriptions() const;
00081
00087     bool hasCommand(const std::string &name) const;
00088
00093     const Targets &targets() const;
00094
00098     ~Parsing();
00099 };
00100
00101
00102 #endif //PARSING_H
```

5.7 target/Targets.cpp File Reference

```
#include <ostream>
#include "Targets.h"
```

Functions

- std::ostream & operator<< (std::ostream &os, const Targets &targets)

5.7.1 Function Documentation

5.7.1.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & os,
            const Targets & targets)
```

5.8 target/Targets.h File Reference

```
#include <string>
#include <vector>
```

Classes

- class Targets

5.9 Targets.h

Go to the documentation of this file.

```
00001 #ifndef TARGETS_H
00002 #define TARGETS_H
00003
00004 #include <string>
00005 #include <vector>
00006
00012 class Targets {
00016     std::string t_description;
00020     std::vector<std::string> t_targs;
00024     bool t_canBeEmpty;
00025
00026 public:
00027     using const_iterator = std::vector<std::string>::const_iterator;
00028
00034     Targets(bool canBeEmpty, const std::string &description);
00035
00040     bool canBeEmpty() const;
00041
00046     const std::vector<std::string> &targets() const;
00047
00052     void addTarget(const std::string &targ);
00053
00058     bool empty() const;
00059
00060     const_iterator begin() const;
00061
00062     const_iterator end() const;
00063
00064     friend std::ostream &operator<<(std::ostream &os, const Targets &targets);
00065 };
00066
00067 #endif // TARGETS_H
```