paramxx

1.0.8

Generated by Doxygen 1.8.15

Chapter 1

The C++ parameter interface library

1.1 Copyright

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

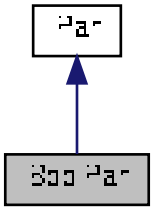
Chapter 4

Class Documentation

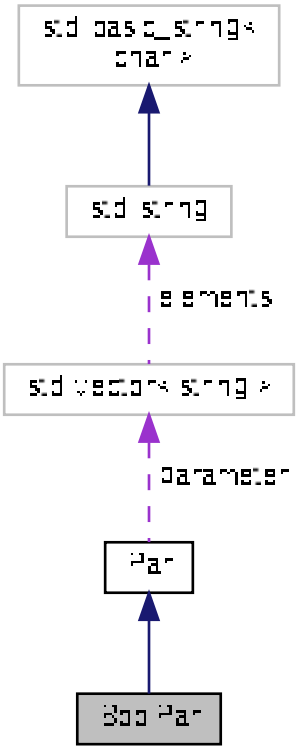
4.1 BoolPar Class Reference

```
#include <BoolPar.h>
```

Inheritance diagram for BoolPar:



Collaboration diagram for BoolPar:



Public Member Functions

- BoolPar ()
- BoolPar (const BoolPar &par)
- **BoolPar** (ParTxt &par)
- bool pgetb (void) const
- void **set_val** (const string &str)

Additional Inherited Members

4.1.1 Detailed Description

A boolean parameter. The class is reponsible to make sure that the parameter can only take a boolean value. The class currently does not check whether min $>$ max.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 BoolPar() [1/2]

```
BoolPar::BoolPar ( )  [inline]
```

The default constructor.

4.1.2.2 BoolPar() [2/2]

```
BoolPar::BoolPar (
            const BoolPar & par )  [inline]
```

The copy constructor.

4.1.3 Member Function Documentation

4.1.3.1 pgetb()

```
bool BoolPar::pgetb (
            void  ) const  [virtual]
```

The user is trying to get the boolean value of a parameter which is anything but a boolean.

Reimplemented from Par.

References Par::PARNAME, and Par::PARVALUE.

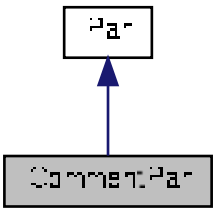
The documentation for this class was generated from the following files:

- BoolPar.h
- BoolPar.cc

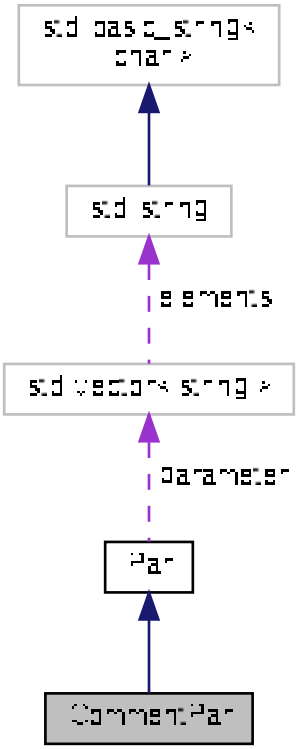
4.2 CommentPar Class Reference

`#include <CommentPar.h>`

Inheritance diagram for CommentPar:



Collaboration diagram for CommentPar:



Public Member Functions

- CommentPar (const CommentPar &par)
- **CommentPar** (ParTxt &par)
- void plist (ostream &os) const
- void **set_val** (const string &str)

Additional Inherited Members

4.2.1 Detailed Description

A class to hold a comment line from a parameter file. A comment line can either begin with with a '#' character or the line can be blank.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 CommentPar()

```
CommentPar::CommentPar (
            const CommentPar & par )  [inline]
```

The copy constructor.

4.2.3 Member Function Documentation

4.2.3.1 plist()

```
void CommentPar::plist (
            ostream & os ) const  [inline], [virtual]
```

Since commentPar does not have the 'name = val prompt' format.

Reimplemented from Par.

References Par::print().

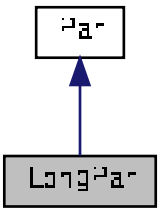
The documentation for this class was generated from the following files:

- CommentPar.h
- CommentPar.cc

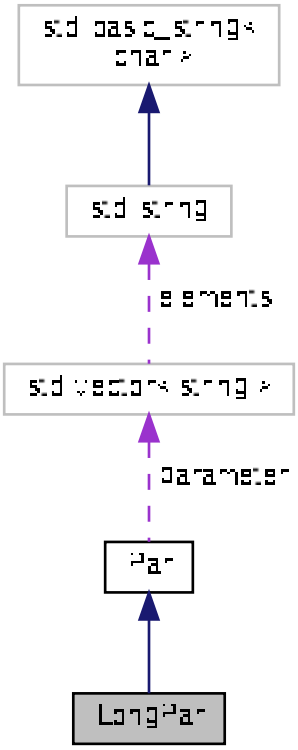
4.3 LongPar Class Reference

`#include <LongPar.h>`

Inheritance diagram for LongPar:



Collaboration diagram for LongPar:



Public Member Functions

- LongPar (const LongPar &par)
- **LongPar** (ParTxt &par)
- int pgeti () const

  only defined in RealPar
- long pgetl () const

  only defined in LongPar
- void **set_val** (const string &str)

Additional Inherited Members

4.3.1 Detailed Description

An long parameter. The class is reponsible to make sure that the parameter can only take a long value.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 LongPar()

```
LongPar::LongPar (
            const LongPar & par )  [inline]
```

The copy constructor.

4.3.3 Member Function Documentation

4.3.3.1 pgeti()

```
int LongPar::pgeti (
            void  ) const  [virtual]
```

only defined in RealPar

Reimplemented from Par.

References Par::PARNAME, and Par::PARVALUE.

4.3.3.2 pgetl()

```
long LongPar::pgetl (
            void  ) const  [inline], [virtual]
```

only defined in LongPar

Reimplemented from Par.

References Par::PARVALUE.

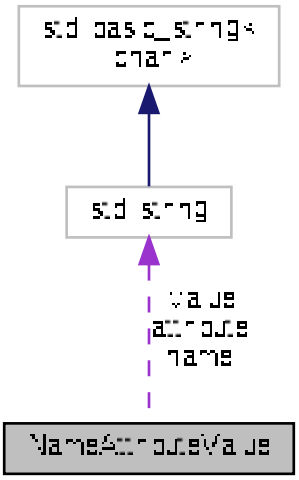
The documentation for this class was generated from the following files:

- LongPar.h
- LongPar.cc

4.4 NameAttributeValue Class Reference

```
#include <NameAttributeValue.h>
```

Collaboration diagram for NameAttributeValue:



Public Member Functions

- virtual ~NameAttributeValue ()

  Destructor.
- NameAttributeValue ()

  Default constructor, all members are initialize to an empty string.
- NameAttributeValue (const char *str)
- string get_attribute (void)

  return the attribute of the object
- string get_name (void) const

  return the name of the object
- string get_value (void) const

  return the pointer to the value of the object
- void init_NameAttributeValue (const char *str)

  Set the name, attribute and value.
- virtual void print (ostream &os=cerr)

  Print the content of the object to a stream.
- void set_attribute (const char *a)

  set the attribute of the object.
- void set_name (const char *n)

  set the name of the object.
- void set_value (const char *v)

  set the value of the object.

Protected Attributes

- string **name**
- string **attribute**
- string **value**

Friends

- ostream & operator<< (ostream &os, NameAttributeValue &a)
- ostream & operator<< (ostream &os, NameAttributeValue *a)

4.4.1 Detailed Description

Parse the name, attribute (if any) and value from a string.

The class parses a string of the form : name.attribute=value, name+, name-, –name for the name ‘name’, attribute ‘attribute’ and value ‘value’.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 ~NameAttributeValue()

```
virtual NameAttributeValue::~NameAttributeValue ( )  [inline], [virtual]
```

Destructor.

4.4.2.2 NameAttributeValue() [1/2]

```
NameAttributeValue::NameAttributeValue ( )  [inline]
```

Default constructor, all members are initialize to an empty string.

4.4.2.3 NameAttributeValue() [2/2]

```
NameAttributeValue::NameAttributeValue (
            const char * str )  [inline]
```

Parse a string, str, into name, attribute and value. Equivalent to calling the default constructor then initialize_Name↩ AttributeValue.

4.4.3 Member Function Documentation

4.4.3.1 get_attribute()

```
string NameAttributeValue::get_attribute (
            void  )  [inline]
```

return the attribute of the object

4.4.3.2 get_name()

```
string NameAttributeValue::get_name (
            void  ) const  [inline]
```

return the name of the object

4.4.3.3 get_value()

```
string NameAttributeValue::get_value (
            void  ) const  [inline]
```

return the pointer to the value of the object

4.4.3.4 init_NameAttributeValue()

```
void NameAttributeValue::init_NameAttributeValue (
            const char * str )
```

Set the name, attribute and value.

4.4.3.5 print()

```
void NameAttributeValue::print (
            ostream & os = cerr )  [virtual]
```

Print the content of the object to a stream.

4.4.3.6 set_attribute()

```
void NameAttributeValue::set_attribute (
            const char * a )  [inline]
```

set the attribute of the object.

4.4.3.7 set_name()

```
void NameAttributeValue::set_name (
            const char * n )  [inline]
```

set the name of the object.

4.4.3.8 set_value()

```
void NameAttributeValue::set_value (
            const char * v )  [inline]
```

set the value of the object.

4.4.4 Friends And Related Function Documentation

4.4.4.1 operator<< [1/2]

```
ostream& operator<< (
            ostream & os,
            NameAttributeValue & a )  [friend]
```

Output operator. This operator outputs the contents of the object.

4.4.4.2 operator<< [2/2]

```
ostream& operator<< (
            ostream & os,
            NameAttributeValue * a )  [friend]
```

Output operator. This operator outputs the contents of the object.

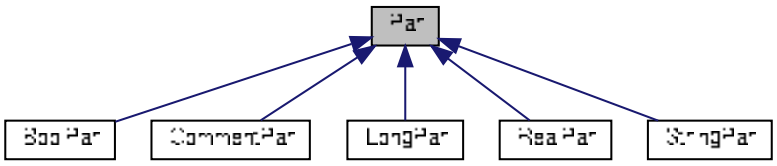
The documentation for this class was generated from the following files:

- NameAttributeValue.h
- NameAttributeValue.cc

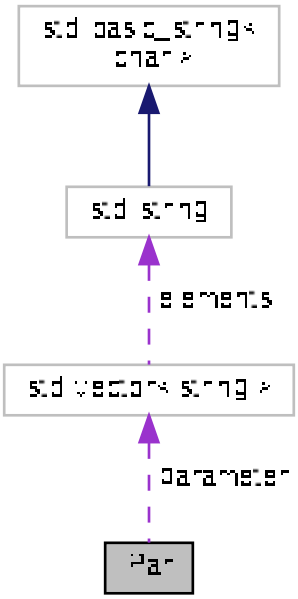
4.5 Par Class Reference

```
#include <Par.h>
```

Inheritance diagram for Par:



Collaboration diagram for Par:



Public Types

- enum ParFileDelimit { **DELIMIT** =',' }
- enum ParType {
  PARNAME, PARTYPE, PARMODE, PARVALUE,
  PARMINIMUM, PARMAXIMUM, PARPROMPT }
- enum **ParamMode** {
  AUTOS ='a', **BATCH** ='b', **hIDDEN** ='h', **HIDDEN** ='H',
  QUERRY ='q', **LEARN** ='l' }
- enum **ParamType** {
  BOOLEAN ='b', **COMMENT** ='c', **INTEGER** ='i', **REAL** ='r',
  STRING ='s' }
- enum **NumTokens** { **NUMTOKENS** =7 }

Public Member Functions

- virtual ~Par (void)
- Par (void)
- **Par** (ParTxt &par)
- int **check_value** (const char *str) const
- virtual void plist (ostream &os) const
- string get_name (void) const

- string get_mode (void) const
- string get_prompt (void) const
- string **get_value** (void) const
- virtual bool pgetb (void) const
- virtual double pgetd (void) const

  only defined in BoolPar
- virtual int pgeti (void) const

  only defined in RealPar
- virtual long pgetl (void) const

  only defined in LongPar
- virtual void pgetstr (char result[ ], size_t size) const

  only defined in LongPar
- virtual string pgetstring (void) const

  only defined in StringPar
- virtual void print (ostream &os) const

  only defined in StringPar
- virtual void **set_val** (const string &str)

Static Public Member Functions

- static int **my_tokenize** (char ∗str, char ∗delimit, char ∗∗∗tokens)
- static void **delete_tokens** (char ∗∗ptr)
- static bool **is_indirrect** (const string &str, char delimit=')')

Protected Member Functions

- void **not_between_limits** (char str[ ], const char ∗left, const string &left_val, const char ∗right, const string &right_val) const

Protected Attributes

- vector< string > **parameter**

Friends

- ostream & operator<< (ostream &os, Par &par)
- ostream & operator<< (ostream &os, Par ∗par)

4.5.1 Detailed Description

The base class for the boolean, double, integer and string parameters.

4.5.2 Member Enumeration Documentation

4.5.2.1 ParFileDelimit

```
enum Par::ParFileDelimit
```

The delimit within the par file

4.5.2.2 ParType

```
enum Par::ParType
```

The position of the parameters.

Enumerator

PARNAME	The name of the parameter
PARTYPE	The type of the parameter
PARMODE	The mode of the parameter
PARVALUE	The value of the parameter
PARMINIMUM	The minimum value of the parameter
PARMAXIMUM	The maximum value of the parameter
PARPROMPT	The name of the parameter

4.5.3 Constructor & Destructor Documentation

4.5.3.1 ~Par()

```
virtual Par::~Par (
            void  )  [inline], [virtual]
```

The destructor.

4.5.3.2 Par()

```
Par::Par (
            void  )  [inline]
```

The default constructor.

4.5.4 Member Function Documentation

4.5.4.1 get_mode()

```
string Par::get_mode (
            void  ) const  [inline]
```

Get the mode of the parameter file

4.5.4.2 get_name()

```
string Par::get_name (
            void  ) const  [inline]
```

Get the name member of the class.

4.5.4.3 get_prompt()

```
string Par::get_prompt (
            void  ) const  [inline]
```

Get the prompt of the parameter

4.5.4.4 pgetb()

```
bool Par::pgetb (
            void  ) const  [virtual]
```

The user is trying to get the boolean value of a parameter which is anything but a boolean.

Reimplemented in BoolPar.

Referenced by ParFile::pgetb().

4.5.4.5 pgetd()

```
double Par::pgetd (
            void  ) const  [virtual]
```

only defined in BoolPar

The user is trying to get the double value of a parameter which is anything but a double.

Reimplemented in RealPar.

Referenced by ParFile::pgetd().

4.5.4.6 pgeti()

```
int Par::pgeti (
            void  ) const  [virtual]
```

only defined in RealPar

Reimplemented in LongPar.

Referenced by ParFile::pgeti().

4.5.4.7 pgetl()

```
long Par::pgetl (
            void  ) const  [virtual]
```

only defined in LongPar

Reimplemented in LongPar.

Referenced by ParFile::pgetl().

4.5.4.8 pgetstr()

```
void Par::pgetstr (
            char result[],
            size_t size ) const  [virtual]
```

only defined in LongPar

Reimplemented in StringPar.

4.5.4.9 pgetstring()

```
string Par::pgetstring (
            void  ) const  [virtual]
```

only defined in StringPar

Reimplemented in StringPar.

Referenced by ParFile::pgetstr(), and ParFile::pgetstring().

4.5.4.10 plist()

```
void Par::plist (
            ostream & os ) const  [virtual]
```

All the derived Par class, with the exception of the comment class, will call this function to print on the screen the parameters with the format name = val prompt

Reimplemented in CommentPar.

4.5.4.11 print()

```
void Par::print (
            ostream & os ) const  [virtual]
```

only defined in StringPar

To output the parameter of the form: name,type,mode,val,min,max,prompt

Reimplemented in StringPar.

Referenced by CommentPar::plist(), and ParFile::print().

4.5.5 Friends And Related Function Documentation

4.5.5.1 operator<< [1/2]

```
ostream& operator<< (
            ostream & os,
            Par & par )  [friend]
```

This operator outputs the contents of the object.

4.5.5.2 operator<< [2/2]

```
ostream& operator<< (
            ostream & os,
            Par * par )  [friend]
```

This operator outputs the contents of the object.

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

- Par.h
- Par.cc

4.6 Parameters Class Reference

Public Member Functions

- **Parameters** (int argc, char *argv[], const char *fname=NULL, const char *boolpars[]=0, const char *doublepars[]=0, const char *intpars[]=0, const char *stringpars[]=0)
- void **init_Parameters** (int argc, char *argv[], const char *fname=NULL, const char *boolpars[]=0, const char *doublepars[]=0, const char *intpars[]=0, const char *stringpars[]=0)
- virtual void **print** (ostream &os=cout, const char *prefix="#: ") const
- bool **pgetb** (const char *) const
- int **pgeti** (const char *) const
- double **pgetd** (const char *) const
- string **pgetstring** (const char *) const

Friends

- ostream & **operator**<< (ostream &os, Parameters &a)
- ostream & **operator**<< (ostream &os, Parameters *a)

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

- Parameters.h
- Parameters.cc

4.7 ParFile Class Reference

```
#include <ParFile.h>
```

Public Member Functions

- ~ParFile ()
- ParFile ()
- ParFile (int argc, char **argv, const char *file=NULL)
- string get_filename ()
- void init_ParFile (int argc, char **argv, const char *file=NULL)
- Par * get_par (const char *name) const

  Get the pointer to the parameter.
- void print (ostream &os) const
- bool pgetb (const char *name) const
- double pgetd (const char *name) const
- int pgeti (const char *name) const
- long pgetl (const char *name) const
- void pgetstr (const char *name, char result[], size_t size) const
- string pgetstring (const char *name) const
- void pget (const char *name, bool &result) const

Same functionality as ParFile::pgetb.

- void pget (const char *name, double &result) const

  Same functionality as ParFile::pgetd.

- void pget (const char *name, int &result) const

  Same functionality as ParFile::pgeti.

- void pget (const char *name, long &result) const

  Same functionality as ParFile::pgetl.

- void pget (const char *name, char result[ ], size_t) const

  Same functionality as ParFile::pgetstr.

- void pget (const char *name, string &rult) const

  Same functionality as ParFile::pgetstring.

- int traverse (int(*fct)(Par *)) const

Static Public Member Functions

- static int **is_blank** (const char *str)
- static int is_comment (const char *str, char delimit='#')

Friends

- ostream & operator<< (ostream &os, ParFile &parFile)
- ostream & operator<< (ostream &os, ParFile *parFile)

4.7.1 Detailed Description

To get the parameters in the parameter file.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 ~ParFile()

```
ParFile::~ParFile ( )
```

The destructor.

4.7.2.2 ParFile() [1/2]

```
ParFile::ParFile ( )  [inline]
```

The default constructor.

4.7.2.3 ParFile() [2/2]

```
ParFile::ParFile (
            int argc,
            char ** argv,
            const char * file = NULL )
```

The constructor to load the parameter file and parse the command line args. Calling this constructor is equivalent to call the default constructor then call the init_ParFile method.

Parameters

argc	The number of parameters from the command line.
argv	The command line arguments.
file	The specific file to use.

Returns

void

Exceptions

ParFileException	

References init_ParFile().

4.7.3 Member Function Documentation

4.7.3.1 get_filename()

```
string ParFile::get_filename ( )  [inline]
```

Get the parameter filename.

Returns

The name of the parameter file.

4.7.3.2 get_par()

```
Par * ParFile::get_par (
            const char * name ) const
```

Get the pointer to the parameter.

Referenced by pgetb(), pgetd(), pgeti(), pgetl(), pgetstr(), and pgetstring().

4.7.3.3 init_ParFile()

```
void ParFile::init_ParFile (
            int argc,
            char ** argv,
            const char * file = NULL )
```

Load the appropiate parameter file and parse the command line args.

Parameters

argc	The number of parameters from the command line.
argv	The command line arguments.
file	The specific file to use.

Returns

void

Exceptions

ParFileException	

Referenced by ParFile().

4.7.3.4 is_comment()

```
int ParFile::is_comment (
            const char * str,
            char delimit = '#' )  [static]
```

if the first character is #, or the line is blank, then the line is a comment

4.7.3.5 pget() [1/6]

```
void ParFile::pget (
            const char * name,
            bool & result ) const
```

Same functionality as ParFile::pgetb.

References pgetb().

4.7.3.6 pget() [2/6]

```
void ParFile::pget (
            const char * name,
            double & result ) const
```

Same functionality as ParFile::pgetd.

References pgetd().

4.7.3.7 pget() [3/6]

```
void ParFile::pget (
            const char * name,
            int & result ) const
```

Same functionality as ParFile::pgeti.

References pgeti().

4.7.3.8 pget() [4/6]

```
void ParFile::pget (
            const char * name,
            long & result ) const
```

Same functionality as ParFile::pgetl.

References pgetl().

4.7.3.9 pget() [5/6]

```
void ParFile::pget (
            const char * name,
            char result[],
            size_t size ) const
```

Same functionality as ParFile::pgetstr.

References pgetstr().

4.7.3.10 pget() [6/6]

```
void ParFile::pget (
            const char * name,
            string & rult ) const
```

Same functionality as ParFile::pgetstring.

References pgetstring().

4.7.3.11 pgetb()

```
bool ParFile::pgetb (
            const char * name ) const
```

To get a parameter of type boolean.

Parameters

name	The name of the parameter to retrieve its value

Returns

The value of the parameter name.

Exceptions

ParFileException	

References get_par(), and Par::pgetb().

Referenced by pget().

4.7.3.12 pgetd()

```
double ParFile::pgetd (
            const char * name ) const
```

To get a parameter of type double.

Parameters

name	The name of the parameter to retrieve its value

Returns

The value of the parameter name.

Exceptions

ParFileException	

References get_par(), and Par::pgetd().

Referenced by pget().

4.7.3.13 pgeti()

```
int ParFile::pgeti (
            const char * name ) const
```

To get a parameter of type long.

Parameters

name	The name of the parameter to retrieve its value

Returns

The value of the parameter name.

Exceptions

ParFileException	

References get_par(), and Par::pgeti().

Referenced by pget().

4.7.3.14 pgetl()

```
long ParFile::pgetl (
            const char * name ) const
```

To get a parameter of type long.

Parameters

name	The name of the parameter to retrieve its value

Returns

The value of the parameter name.

Exceptions

ParFileException	

References get_par(), and Par::pgetl().

Referenced by pget().

4.7.3.15 pgetstr()

```
void ParFile::pgetstr (
            const char * name,
            char result[],
            size_t size ) const
```

To get a parameter of type char[]. The length of the result in the following function must be at least size+1

Parameters

name	The name of the parameter to retrieve its value

Returns

The value of the parameter name.

Exceptions

ParFileException	

References get_par(), and Par::pgetstring().

Referenced by pget().

4.7.3.16 pgetstring()

```
string ParFile::pgetstring (
            const char * name ) const
```

To get a parameter of type string.

Parameters

name	The name of the parameter to retrieve its value

Returns

The value of the parameter name.

Exceptions

ParFileException	

References get_par(), and Par::pgetstring().

Referenced by pget().

4.7.3.17 print()

```
void ParFile::print (
            ostream & os ) const
```

Loop over the parameters to call the print method for each parameter. The individual parameter will print its content in the form: name,type,mode,val,min,max,prompt

Parameters

os	The stream to print the content of the parameter file.

References Par::print().

4.7.3.18 traverse()

```
int ParFile::traverse (
            int(*)(Par *) fct ) const
```

Traverse the parameters list and invoke the function fct, if fct returns a non-zero the traversal is aborted and the fct return value is returned.

4.7.4 Friends And Related Function Documentation

4.7.4.1 operator<< [1/2]

```
ostream& operator<< (
            ostream & os,
            ParFile & parFile )  [friend]
```

This operator outputs the contents of the object.

4.7.4.2 operator<< [2/2]

```
ostream& operator<< (
            ostream & os,
            ParFile * parFile )  [friend]
```

This operator outputs the contents of the object.

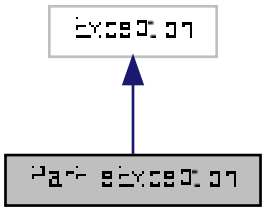
The documentation for this class was generated from the following files:

- ParFile.h
- ParFile.cc

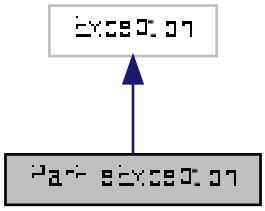
4.8 ParFileException Class Reference

```
#include <ParFileException.h>
```

Inheritance diagram for ParFileException:



Collaboration diagram for ParFileException:



Public Member Functions

- **ParFileException** (const std::string &msg)

4.8.1 Detailed Description

The exception thrown by the ParFile library.

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

- ParFileException.h
- ParFileException.cc

4.9 ParFilename Class Reference

```
#include <ParFilename.h>
```

Public Member Functions

- ~ParFilename (void)
- ParFilename ()
- ParFilename (int argc, char **argv, const char *file=NULL)
- string get_filename (void)
- void init_ParFilename (int argc, char **argv, const char *file=NULL)
- void **print** (ostream &os)

Static Public Member Functions

- static vector< string > parse_string (string &str, char delimit)

Friends

- ostream & operator<< (ostream &os, ParFilename &a)
- ostream & operator<< (ostream &os, ParFilename *a)

4.9.1 Detailed Description

Find the parameter file. If PFILE=foo is set at the command line then the file 'foo' shall be the file to be opened. PFILE will overide even if a specific file to the function (3rd argument) Parfile is set. The class loops over the PFILES, UPARM environment variables to determine then the .par and extension to locate the parameter file.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 ~ParFilename()

```
ParFilename::~ParFilename (
            void  )  [inline]
```

destructor

4.9.2.2 ParFilename() [1/2]

```
ParFilename::ParFilename ( )  [inline]
```

default constructor

4.9.2.3 ParFilename() [2/2]

```
ParFilename::ParFilename (
            int argc,
            char ** argv,
            const char * file = NULL )
```

constructor

4.9.3 Member Function Documentation

4.9.3.1 get_filename()

```
string ParFilename::get_filename (
            void  )  [inline]
```

Get the parameter filename

4.9.3.2 init_ParFilename()

```
void ParFilename::init_ParFilename (
            int argc,
            char ** argv,
            const char * file = NULL )
```

Initialize the parameter

4.9.3.3 parse_string()

```
vector< string > ParFilename::parse_string (
            string & str,
            char delimit )  [static]
```

Parameters

str	the string to be split.
delimiter	the delimit to split the string.

Splits the input line on the parameter delimiter.

Returns

Vector of tokens, one for each delimiter delimited field.

4.9.4 Friends And Related Function Documentation

4.9.4.1 operator<< [1/2]

```
ostream& operator<< (
            ostream & os,
            ParFilename & a )  [friend]
```

This output operator outputs the contents of the object.

4.9.4.2 operator<< [2/2]

```
ostream& operator<< (
            ostream & os,
            ParFilename * a )  [friend]
```

This output operator outputs the contents of the object.

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

- ParFilename.h
- ParFilename.cc

4.10 ParTxt Class Reference

A class to handle the Param library specific string manipulation.

```
#include <ParTxt.h>
```

Public Member Functions

- **ParTxt** (char *str=NULL)
- void **init_ParTxt** (char *str=NULL)
- **operator char** * () const
- **operator char** ** () const
- char * **get_token** (size_t n)
- void **print** (std::ostream &os)

Static Public Member Functions

- static void verify_mode (const char *)

Static Public Attributes

- static int **restore** =0

Protected Attributes

- char * **buffer**
- char ** **buffer_argv**
- size_t **num_tokens**

Friends

- std::ostream & operator<< (std::ostream &os, ParTxt &par)

  This operator outputs the contents of the object.
- std::ostream & operator<< (std::ostream &os, ParTxt *par)

  This operator outputs the contents of the object.

4.10.1 Detailed Description

A class to handle the Param library specific string manipulation.

4.10.2 Member Function Documentation

4.10.2.1 verify_mode()

```
void ParTxt::verify_mode (
            const char * str ) [static]
```

Check to see if 'mode' is one of the followig options: a, b, h, H, l

4.10.3 Friends And Related Function Documentation

4.10.3.1 operator<< [1/2]

```
std::ostream& operator<< (
            std::ostream & os,
            ParTxt & par )  [friend]
```

This operator outputs the contents of the object.

4.10.3.2 operator<< [2/2]

```
std::ostream& operator<< (
            std::ostream & os,
            ParTxt * par )  [friend]
```

This operator outputs the contents of the object.

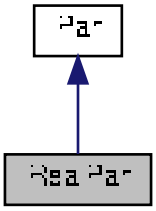
The documentation for this class was generated from the following files:

- ParTxt.h
- ParTxt.cc

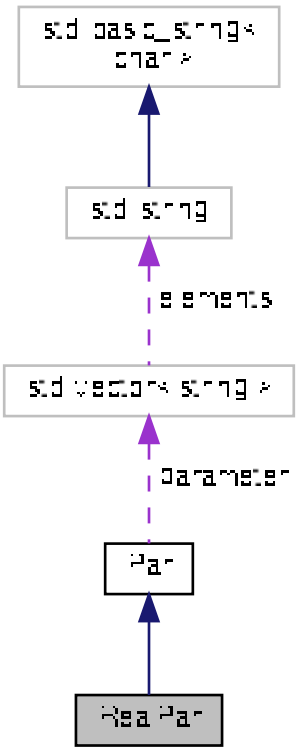
4.11 RealPar Class Reference

```
#include <RealPar.h>
```

Inheritance diagram for RealPar:



Collaboration diagram for RealPar:



Public Member Functions

- RealPar (const RealPar &par)

  The copy constructor.
- RealPar (ParTxt &par)

  constructor.
- double pgetd (void) const

  only defined in BoolPar
- void **set_val** (const string &str)

Additional Inherited Members

4.11.1 Detailed Description

An double parameter. The class is reponsible to make sure that the parameter can only take a double value.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 RealPar() [1/2]

```
RealPar::RealPar (
            const RealPar & par )  [inline]
```

The copy constructor.

4.11.2.2 RealPar() [2/2]

```
RealPar::RealPar (
            ParTxt & par )
```

constructor.

References Par::PARMAXIMUM, Par::PARMINIMUM, and Par::PARVALUE.

4.11.3 Member Function Documentation

4.11.3.1 pgetd()

```
double RealPar::pgetd (
            void  ) const  [inline], [virtual]
```

only defined in BoolPar

The user is trying to get the double value of a parameter which is anything but a double.

Reimplemented from Par.

References Par::PARVALUE.

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

- RealPar.h
- RealPar.cc

4.12 stlPtrWrapper< Type > Class Template Reference

`#include <stlPtrWrapper.h>`

Public Member Functions

- **stlPtrWrapper** (Type x=0)
- **stlPtrWrapper** (const stlPtrWrapper< Type > &xxx)
- stlPtrWrapper & **operator=** (const stlPtrWrapper< Type > &rhs)
- const Type **operator()** () const
- Type **operator()** ()

Friends

- ostream & **operator**<< (ostream &os, stlPtrWrapper< Type > &a)
- ostream & **operator**<< (ostream &os, stlPtrWrapper< Type > *a)
- bool **operator==** (const stlPtrWrapper< Type > &xw1, const stlPtrWrapper< Type > &xw2)
- bool **operator**< (const stlPtrWrapper< Type > &xw1, const stlPtrWrapper< Type > &xw2)

4.12.1 Detailed Description

template<class Type>
class stlPtrWrapper< Type >

A simple wrapper class to put pointers into STL containers.

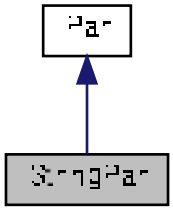
The documentation for this class was generated from the following file:

- stlPtrWrapper.h

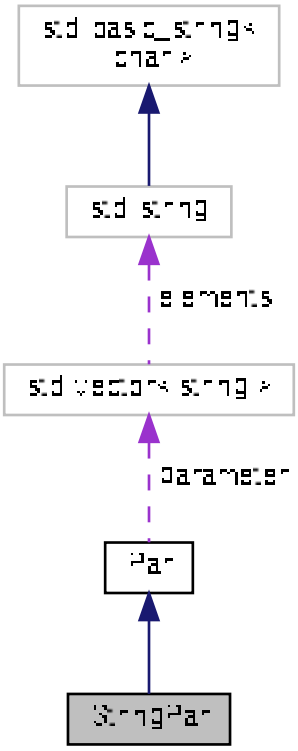
4.13 StringPar Class Reference

`#include <StringPar.h>`

Inheritance diagram for StringPar:



Collaboration diagram for StringPar:



Public Member Functions

- StringPar (const StringPar &par)
- **StringPar** (ParTxt &par)
- void pgetstr (char result[ ], size_t size) const
  only defined in LongPar
- string pgetstring () const
  only defined in StringPar
- void print (ostream &os) const
- void **set_val** (const string &str)

Additional Inherited Members

4.13.1 Detailed Description

A string parameter. The class is reponsible to make sure that the parameter can only take a string value.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 StringPar()

```
StringPar::StringPar (
            const StringPar & par )  [inline]
```

The copy constructor.

4.13.3 Member Function Documentation

4.13.3.1 pgetstr()

```
void StringPar::pgetstr (
            char result[],
            size_t size ) const  [inline], [virtual]
```

only defined in LongPar

Reimplemented from Par.

References Par::PARVALUE.

4.13.3.2 pgetstring()

```
string StringPar::pgetstring (
            void  ) const  [inline], [virtual]
```

only defined in StringPar

Reimplemented from Par.

References Par::PARVALUE.

4.13.3.3 print()

```
void StringPar::print (
            ostream & os ) const  [virtual]
```

To output the parameter of the form: name,type,mode,"val",min,max,prompt

Reimplemented from Par.

References Par::PARMODE.

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

- StringPar.h
- StringPar.cc

Chapter 5

Example Documentation

5.1 ptest.cc

The classic way to read the parameter file.

5.2 ptest.par

Given a parameter file of the form:

Index