paramxx

1.0.6

Generated by Doxygen 1.5.6

Thu Oct 2 17:54:19 2008

Contents

1 The C++ parameter interface library

1.1 Copyright

2 Directory Hierarchy

2.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

3 Class Index

3.1 Class Hierarchy

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

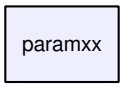
4 Class Index

4.1 Class List

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

5 Directory Documentation

5.1 /data/macabretmp/dj/paramxx/paramxx/ Directory Reference



Files

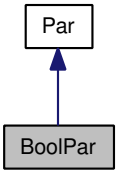
- file **BoolPar.cc**
- file **BoolPar.h**
- file **CommentPar.cc**
- file **CommentPar.h**
- file **LongPar.cc**
- file **LongPar.h**
- file **NameAttributeValue.cc**
- file **NameAttributeValue.h**
- file **Par.cc**
- file **Par.h**
- file **Parameters.cc**
- file **Parameters.h**
- file **ParFile.cc**
- file **ParFile.h**
- file **ParFileException.cc**
- file **ParFileException.h**
- file **ParFilename.cc**
- file **ParFilename.h**
- file **ParTxt.cc**
- file **ParTxt.h**
- file **RealPar.cc**
- file **RealPar.h**
- file **stlPtrWrapper.h**
- file **StringPar.cc**
- file **StringPar.h**

6 Class Documentation

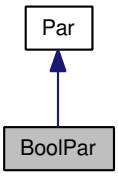
6.1 BoolPar Class Reference

`#include <BoolPar.h>`

Inheritance diagram for BoolPar:



Collaboration diagram for BoolPar:



Public Member Functions

- BoolPar ()
- BoolPar (const BoolPar &par)
- **BoolPar** (ParTxt &par) throw (ParFileException)
- bool pgetb (void) const throw (ParFileException)
- void **set_val** (const string &str) throw (ParFileException, Exception)

6.1.1 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.

Definition at line 36 of file BoolPar.h.

6.1.2 Constructor & Destructor Documentation

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

The default constructor.

Definition at line 45 of file BoolPar.h.

6.1.2.2 BoolPar::BoolPar (const BoolPar & *par*) [inline]

The copy constructor.

Definition at line 50 of file BoolPar.h.

6.1.3 Member Function Documentation

6.1.3.1 bool BoolPar::pgetb (void) const throw (ParFileException) [virtual]

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

Reimplemented from Par.

Definition at line 59 of file BoolPar.cc.

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

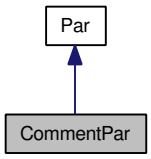
The documentation for this class was generated from the following files:

- BoolPar.h
- BoolPar.cc

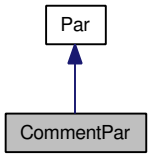
6.2 CommentPar Class Reference

`#include <CommentPar.h>`

Inheritance diagram for CommentPar:



Collaboration diagram for CommentPar:



Public Member Functions

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

6.2.1 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.

Definition at line 35 of file CommentPar.h.

6.2.2 Constructor & Destructor Documentation

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

The copy constructor.

Definition at line 46 of file CommentPar.h.

6.2.3 Member Function Documentation

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

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

Reimplemented from Par.

Definition at line 58 of file CommentPar.h.

References Par::print().

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

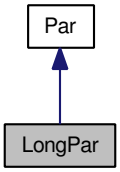
- CommentPar.h

- CommentPar.cc

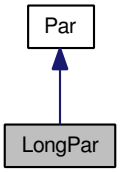
6.3 LongPar Class Reference

`#include <LongPar.h>`

Inheritance diagram for LongPar:



Collaboration diagram for LongPar:



Public Member Functions

- LongPar (const LongPar &par)
- **LongPar** (ParTxt &par) throw (ParFileException, Exception)
- int pgeti () const throw (ParFileException)

  only defined in RealPar

- long pgetl () const throw (ParFileException)

  only defined in LongPar

- void **set_val** (const string &str) throw (ParFileException, Exception)

6.3.1 Detailed Description

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

Definition at line 35 of file LongPar.h.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 LongPar::LongPar (const LongPar & *par*) [inline]

The copy constructor.

Definition at line 46 of file LongPar.h.

6.3.3 Member Function Documentation

6.3.3.1 int LongPar::pgeti (void) const throw (ParFileException) [virtual]

only defined in RealPar

Reimplemented from Par.

Definition at line 100 of file LongPar.cc.

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

6.3.3.2 long LongPar::pgetl (void) const throw (ParFileException) [inline, virtual]

only defined in LongPar

Reimplemented from Par.

Definition at line 53 of file LongPar.h.

References Par::PARVALUE.

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

- LongPar.h
- LongPar.cc

6.4 NameAttributeValue Class Reference

`#include <NameAttributeValue.h>`

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)

6.4.1 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'.

Definition at line 40 of file NameAttributeValue.h.

6.4.2 Constructor & Destructor Documentation

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

Destructor.

Definition at line 61 of file NameAttributeValue.h.

6.4.2.2 NameAttributeValue::NameAttributeValue () [inline]

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

Definition at line 64 of file NameAttributeValue.h.

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

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

Definition at line 70 of file NameAttributeValue.h.

References init_NameAttributeValue().

6.4.3 Member Function Documentation

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

return the attribute of the object

Definition at line 73 of file NameAttributeValue.h.

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

return the name of the object

Definition at line 76 of file NameAttributeValue.h.

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

return the pointer to the value of the object

Definition at line 79 of file NameAttributeValue.h.

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

Set the name, attribute and value.

Definition at line 70 of file NameAttributeValue.cc.

Referenced by NameAttributeValue().

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

Print the content of the object to a stream.

Definition at line 103 of file NameAttributeValue.cc.

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

set the attribute of the object.

Definition at line 88 of file NameAttributeValue.h.

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

set the name of the object.

Definition at line 91 of file NameAttributeValue.h.

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

set the value of the object.

Definition at line 94 of file NameAttributeValue.h.

6.4.4 Friends And Related Function Documentation

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

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

Definition at line 45 of file NameAttributeValue.h.

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

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

Definition at line 53 of file NameAttributeValue.h.

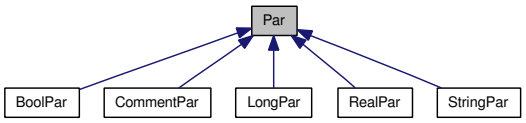
The documentation for this class was generated from the following files:

- NameAttributeValue.h
- NameAttributeValue.cc

6.5 Par Class Reference

`#include <Par.h>`

Inheritance 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 throw (ParFileException)
- virtual double pgetd (void) const throw (ParFileException)

  only defined in BoolPar

- virtual int pgeti (void) const throw (ParFileException)

  only defined in RealPar

- virtual long pgetl (void) const throw (ParFileException)

  only defined in LongPar

- virtual void pgetstr (char result[ ], size_t size) const throw (ParFileException)

  only defined in LongPar

- virtual string pgetstring (void) const throw (ParFileException)

  only defined in StringPar

- virtual void print (ostream &os) const

  only defined in StringPar

- virtual void **set_val** (const string &str) throw (ParFileException, Exception)

Static Public Member Functions

- static int **my_tokenize** (char *str, char *delimit, char ***tokens) throw (Exception)
- 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)

6.5.1 Detailed Description

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

Definition at line 41 of file Par.h.

6.5.2 Member Enumeration Documentation

6.5.2.1 enum Par::ParFileDelimit

The delimit within the par file

Definition at line 64 of file Par.h.

6.5.2.2 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

Definition at line 69 of file Par.h.

6.5.3 Constructor & Destructor Documentation

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

The destructor.

Definition at line 89 of file Par.h.

6.5.3.2 Par::Par (void) [inline]

The default constructor.

Definition at line 94 of file Par.h.

6.5.4 Member Function Documentation

6.5.4.1 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.

Definition at line 105 of file Par.cc.

References PARMODE, PARNAME, PARPROMPT, and PARVALUE.

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

Get the name member of the class.

Definition at line 110 of file Par.h.

References PARNAME.

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

Get the mode of the parameter file

Definition at line 115 of file Par.h.

References PARMODE.

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

Get the prompt of the parameter

Definition at line 120 of file Par.h.

References PARPROMPT.

6.5.4.5 bool Par::pgetb (void) const throw (ParFileException) [virtual]

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

Reimplemented in BoolPar.

Definition at line 146 of file Par.cc.

Referenced by ParFile::pgetb().

6.5.4.6 double Par::pgetd (void) const throw (ParFileException) [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.

Definition at line 153 of file Par.cc.

Referenced by ParFile::pgetd().

6.5.4.7 int Par::pgeti (void) const throw (ParFileException) [virtual]

only defined in RealPar

Reimplemented in LongPar.

Definition at line 160 of file Par.cc.

Referenced by ParFile::pgeti().

6.5.4.8 long Par::pgetl (void) const throw (ParFileException) [virtual]

only defined in LongPar

Reimplemented in LongPar.

Definition at line 167 of file Par.cc.

Referenced by ParFile::pgetl().

6.5.4.9 void Par::pgetstr (char *result*[ ], size_t *size*) const throw (ParFileException) [virtual]

only defined in LongPar

Reimplemented in StringPar.

Definition at line 174 of file Par.cc.

6.5.4.10 string Par::pgetstring (void) const throw (ParFileException) [virtual]

only defined in StringPar

Reimplemented in StringPar.

Definition at line 182 of file Par.cc.

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

6.5.4.11 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.

Definition at line 189 of file Par.cc.

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

6.5.5 Friends And Related Function Documentation

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

This operator outputs the contents of the object.

Definition at line 46 of file Par.h.

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

This operator outputs the contents of the object.

Definition at line 54 of file Par.h.

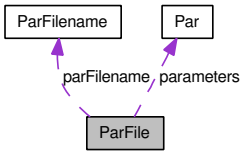
The documentation for this class was generated from the following files:

- Par.h
- Par.cc

6.6 ParFile Class Reference

`#include <ParFile.h>`

Collaboration diagram for ParFile:



Public Member Functions

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

  Get the pointer to the parameter.

- void print (ostream &os) const
- bool pgetb (const char *name) const throw (ParFileException)
- double pgetd (const char *name) const throw (ParFileException)
- int pgeti (const char *name) const throw (ParFileException)
- long pgetl (const char *name) const throw (ParFileException)
- void pgetstr (const char *name, char result[ ], size_t size) const throw (ParFileException)
- string pgetstring (const char *name) const throw (ParFileException)
- void pget (const char *name, bool &result) const throw (ParFileException)

  Same functionality as ParFile::pgetb.

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

  Same functionality as ParFile::pgetd.

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

  Same functionality as ParFile::pgeti.

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

  Same functionality as ParFile::pgetl.

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

  Same functionality as ParFile::pgetstr.

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

  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)

6.6.1 Detailed Description

To get the parameters in the parameter file.

Examples:

ptest.cc.

Definition at line 38 of file ParFile.h.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 ParFile::~ParFile ()

The destructor.

Definition at line 47 of file ParFile.cc.

6.6.2.2 ParFile::ParFile () [inline]

The default constructor.

Definition at line 66 of file ParFile.h.

6.6.2.3 ParFile::ParFile (int *argc*, char ** *argv*, const char * *file* = NULL) throw (ParFileException, Exception)

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

Definition at line 59 of file ParFile.cc.

References init_ParFile().

6.6.3 Member Function Documentation

6.6.3.1 string ParFile::get_filename () [inline]

Get the parameter filename.

Returns:

The name of the parameter file.

Definition at line 85 of file ParFile.h.

References ParFilename::get_filename().

6.6.3.2 void ParFile::init_ParFile (int *argc*, char ** *argv*, const char * *file* = NULL) throw (ParFileException, Exception)

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

Definition at line 294 of file ParFile.cc.

Referenced by ParFile().

6.6.3.3 Par * ParFile::get_par (const char * *name*) const throw (ParFileException)

Get the pointer to the parameter.

Definition at line 255 of file ParFile.cc.

References Par::get_value().

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

6.6.3.4 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.

Definition at line 368 of file ParFile.cc.

References Par::print().

6.6.3.5 bool ParFile::pgetb (const char * *name*) const throw (ParFileException)

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

Examples:

ptest.cc.

Definition at line 138 of file ParFile.cc.

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

Referenced by pget().

6.6.3.6 double ParFile::pgetd (const char * *name*) const throw (ParFileException)

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

Examples:

ptest.cc.

Definition at line 149 of file ParFile.cc.

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

Referenced by pget().

6.6.3.7 int ParFile::pgeti (const char * *name*) const throw (ParFileException)

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

Examples:

ptest.cc.

Definition at line 160 of file ParFile.cc.

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

Referenced by pget().

6.6.3.8 long ParFile::pgetl (const char * *name*) const throw (ParFileException)

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

Examples:

ptest.cc.

Definition at line 171 of file ParFile.cc.

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

Referenced by pget().

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

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

Examples:

ptest.cc.

Definition at line 182 of file ParFile.cc.

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

Referenced by pget().

6.6.3.10 string ParFile::pgetstring (const char * *name*) const throw (ParFileException)

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

Examples:

ptest.cc.

Definition at line 197 of file ParFile.cc.

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

Referenced by pget().

6.6.3.11 void ParFile::pget (const char * *name*, bool & *result*) const throw (ParFileException)

Same functionality as ParFile::pgetb.

Examples:

ptest.cc.

Definition at line 72 of file ParFile.cc.

References pgetb().

6.6.3.12 void ParFile::pget (const char * *name*, double & *result*) const throw (ParFileException)

Same functionality as ParFile::pgetd.

Definition at line 83 of file ParFile.cc.

References pgetd().

6.6.3.13 void ParFile::pget (const char * *name*, int & *result*) const throw (ParFileException)

Same functionality as ParFile::pgeti.

Definition at line 94 of file ParFile.cc.

References pgeti().

6.6.3.14 void ParFile::pget (const char * *name*, long & *result*) const throw (ParFileException)

Same functionality as ParFile::pgetl.

Definition at line 105 of file ParFile.cc.

References pgetl().

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

Same functionality as ParFile::pgetstr.

Definition at line 116 of file ParFile.cc.

References pgetstr().

6.6.3.16 void ParFile::pget (const char * *name*, string & *rult*) const throw (ParFileException)

Same functionality as ParFile::pgetstring.

Definition at line 127 of file ParFile.cc.

References pgetstring().

6.6.3.17 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.

Definition at line 561 of file ParFile.cc.

6.6.3.18 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

Definition at line 585 of file ParFile.cc.

6.6.4 Friends And Related Function Documentation

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

This operator outputs the contents of the object.

Definition at line 43 of file ParFile.h.

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

This operator outputs the contents of the object.

Definition at line 51 of file ParFile.h.

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

- ParFile.h
- ParFile.cc

6.7 ParFileException Class Reference

`#include <ParFileException.h>`

Public Member Functions

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

6.7.1 Detailed Description

The exception thrown by the ParFile library.

Examples:

ptest.cc.

Definition at line 33 of file ParFileException.h.

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

- ParFileException.h
- ParFileException.cc

6.8 ParFilename Class Reference

`#include <ParFilename.h>`

Public Member Functions

- ~ParFilename (void)
- ParFilename ()
- ParFilename (int argc, char **argv, const char *file=NULL) throw (ParFileException)
- string get_filename (void)
- void init_ParFilename (int argc, char **argv, const char *file=NULL) throw (ParFileException)
- 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)

6.8.1 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.

Definition at line 44 of file ParFilename.h.

6.8.2 Constructor & Destructor Documentation

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

destructor

Definition at line 69 of file ParFilename.h.

6.8.2.2 ParFilename::ParFilename () [inline]

default constructor

Definition at line 74 of file ParFilename.h.

6.8.2.3 ParFilename::ParFilename (int *argc*, char ** *argv*, const char * *file* = NULL) throw (ParFileException)

constructor

Definition at line 48 of file ParFilename.cc.

6.8.3 Member Function Documentation

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

Get the parameter filename

Definition at line 85 of file ParFilename.h.

Referenced by ParFile::get_filename().

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

Initialize the parameter

Definition at line 245 of file ParFilename.cc.

6.8.3.3 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.

Definition at line 315 of file ParFilename.cc.

6.8.4 Friends And Related Function Documentation

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

This output operator outputs the contents of the object.

Definition at line 49 of file ParFilename.h.

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

This output operator outputs the contents of the object.

Definition at line 57 of file ParFilename.h.

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

- ParFilename.h
- ParFilename.cc

6.9 ParTxt Class Reference

A class to handle the Param library specific string manipulation.

`#include <ParTxt.h>`

Public Member Functions

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

Static Public Member Functions

- static void verify_mode (const string &str) throw (ParFileException)

Static Public Attributes

- static int **restore** = 0

Protected Attributes

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

Friends

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

  This operator outputs the contents of the object.

- ostream & operator<< (ostream &os, ParTxt *par)

  This operator outputs the contents of the object.

6.9.1 Detailed Description

A class to handle the Param library specific string manipulation.

Definition at line 31 of file ParTxt.h.

6.9.2 Member Function Documentation

6.9.2.1 void ParTxt::verify_mode (const string & *str*) throw (ParFileException) [static]

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

Definition at line 127 of file ParTxt.cc.

6.9.3 Friends And Related Function Documentation

6.9.3.1 ostream& operator<< (ostream & *os*, ParTxt & *par*) [friend]

This operator outputs the contents of the object.

Definition at line 34 of file ParTxt.h.

6.9.3.2 ostream& operator<< (ostream & *os*, ParTxt * *par*) [friend]

This operator outputs the contents of the object.

Definition at line 40 of file ParTxt.h.

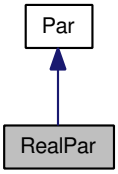
The documentation for this class was generated from the following files:

- ParTxt.h
- ParTxt.cc

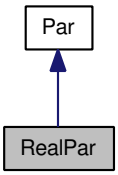
6.10 RealPar Class Reference

`#include <RealPar.h>`

Inheritance diagram for RealPar:



Collaboration diagram for RealPar:



Public Member Functions

- RealPar (const RealPar &par)

  The copy constructor.

- RealPar (ParTxt &par) throw (ParFileException, Exception)

  constructor.

- double pgetd (void) const throw (ParFileException)

  only defined in BoolPar

- void **set_val** (const string &str) throw (ParFileException, Exception)

6.10.1 Detailed Description

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

Definition at line 35 of file RealPar.h.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 RealPar::RealPar (const RealPar & *par*) [inline]

The copy constructor.

Definition at line 44 of file RealPar.h.

6.10.2.2 RealPar::RealPar (ParTxt & *par*) throw (ParFileException, Exception)

constructor.

Definition at line 32 of file RealPar.cc.

6.10.3 Member Function Documentation

6.10.3.1 double RealPar::pgetd (void) const throw (ParFileException) [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.

Definition at line 49 of file RealPar.h.

References Par::PARVALUE.

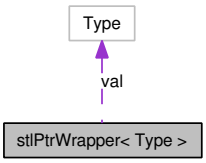
The documentation for this class was generated from the following files:

- RealPar.h
- RealPar.cc

6.11 stlPtrWrapper< Type > Class Template Reference

`#include <stlPtrWrapper.h>`

Collaboration diagram for stlPtrWrapper< Type >:



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)

6.11.1 Detailed Description

template<class Type> class stlPtrWrapper< Type >

A simple wrapper class to put pointers into STL containers.

Definition at line 36 of file stlPtrWrapper.h.

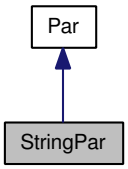
The documentation for this class was generated from the following file:

- stlPtrWrapper.h

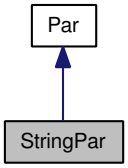
6.12 StringPar Class Reference

`#include <StringPar.h>`

Inheritance diagram for StringPar:



Collaboration diagram for StringPar:



Public Member Functions

- StringPar (const StringPar &par)
- **StringPar** (ParTxt &par) throw (ParFileException)
- void pgetstr (char result[ ], size_t size) const throw (ParFileException)

  only defined in LongPar

- string pgetstring () const throw (ParFileException)

  only defined in StringPar

- void print (ostream &os) const
- void **set_val** (const string &str) throw (ParFileException, Exception)

6.12.1 Detailed Description

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

Definition at line 35 of file StringPar.h.

6.12.2 Constructor & Destructor Documentation

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

The copy constructor.

Definition at line 46 of file StringPar.h.

6.12.3 Member Function Documentation

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

only defined in LongPar

Reimplemented from Par.

Definition at line 51 of file StringPar.h.

References Par::PARVALUE.

6.12.3.2 string StringPar::pgetstring (void) const throw (ParFileException) [inline, virtual]

only defined in StringPar

Reimplemented from Par.

Definition at line 56 of file StringPar.h.

References Par::PARVALUE.

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

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

Reimplemented from Par.

Definition at line 94 of file StringPar.cc.

References Par::PARMODE.

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

- StringPar.h
- StringPar.cc

7 Example Documentation

7.1 ptest.cc

The classic way to read the parameter file.

```
#include <stdlib.h>
#include <iostream>

#include <testlib/testlib.h>

#include <paramxx/ParFile.h>

static int test_boolean( ParFile& pf ) {

  try {

    bool abool = pf.pgetb( "aboolean" );
```

```
    bool anotherbool;
    pf.pget( "aboolean", anotherbool );

    TEST( "boolean", abool, anotherbool );

    return EXIT_SUCCESS;

  } catch( ParFileException& pfe ) {

    cerr << pfe;
    return EXIT_FAILURE;

  } catch ( Exception& pfe ) {

    cerr << pfe;
    return EXIT_FAILURE;

  }

}

static int test_double( ParFile& pf ) {

  try {

    double adouble = pf.pgetd( "areal" );

    double anotherdouble;
    pf.pget( "areal", anotherdouble );

    TEST( "double", adouble, anotherdouble );

    return EXIT_SUCCESS;

  } catch( ParFileException& pfe ) {
    cerr << pfe;
    return EXIT_FAILURE;
  } catch ( Exception& pfe ) {

    cerr << pfe;
    return EXIT_FAILURE;

  }

}

static int test_int( ParFile& pf ) {

  try {

    int anint = pf.pgeti( "anint" );

    int anotherint;

    pf.pget( "anint", anotherint );

    TEST( "integer", anint, anotherint );
```

```
    return EXIT_SUCCESS;

  } catch ( ParFileException& pfe ) {

    cerr << pfe;
    return EXIT_FAILURE;

  } catch ( Exception& pfe ) {

    cerr << pfe;
    return EXIT_FAILURE;

  }

}

static int test_long( ParFile& pf ) {

  try {
    long along = pf.pgetl( "along" );

    long anotherlong;
    pf.pget( "along", anotherlong );

    TEST( "long", along, anotherlong );

    return EXIT_SUCCESS;

  } catch ( ParFileException& pfe ) {

    cerr << pfe;
    return EXIT_FAILURE;

  } catch ( Exception& pfe ) {

    cerr << pfe;
    return EXIT_FAILURE;

  }

}

static int test_string( ParFile& pf ) {


  try {

    string astring = pf.pgetstring( "astr" );

    string anotherstring;
    pf.pget( "astr", anotherstring );

    TESTSTRING( "string", astring.c_str( ), anotherstring.c_str( ) );

    return EXIT_SUCCESS;
```

```
  } catch ( ParFileException& pfe ) {

    cerr << pfe;
    return EXIT_FAILURE;

  } catch ( Exception& pfe ) {

    cerr << pfe;
    return EXIT_FAILURE;

  }

}

static int test( int argc, char** argv ) {

  try {
    ParFile pf( argc, argv );

    char str[ 256 ];
    pf.pgetstr( "foo", str, 256 );

    if ( EXIT_FAILURE == test_boolean( pf ) )
      return EXIT_FAILURE;

    if ( EXIT_FAILURE == test_int( pf ) )
      return EXIT_FAILURE;

    if ( EXIT_FAILURE == test_long( pf ) )
      return EXIT_FAILURE;

    if ( EXIT_FAILURE == test_string( pf ) )
      return EXIT_FAILURE;

    if ( EXIT_FAILURE == test_double( pf ) )
      return EXIT_FAILURE;

  } catch ( ParFileException& pfe ) {
    cerr << pfe;
    return EXIT_FAILURE;
  } catch ( Exception& e ) {
    cerr << e;
    return EXIT_FAILURE;
  }

  return EXIT_SUCCESS;

}

int main( int argc, char** argv ) {

  START( "Testing ParFile with indirrection" );

  if ( EXIT_FAILURE == test( argc, argv ) )
    return EXIT_FAILURE;

  SUMMARY();
```

```
  return EXIT_SUCCESS;
}
```

7.2 ptest.par

Given a parameter file of the form:

```
aboolean,b,a,)abooleanindirrection,,,"To test boolean indirrection"
abooleanindirrection,b,a,no,,,"blah blah..."
anint,i,a,)anintindirection,-5,5,"To test int indirrection"
anintindirection,i,a,3,-5,5,"blah blah..."
along,i,a,)alongindirrection,-5,5,"To test long indirrection"
alongindirrection,i,a,3,-5,5,"blah blah..."
areal,r,a,)arealindirrection,-25.0,25.0,"To test real indirrection"
arealindirrection,r,a,1,-10.0,10.0,"blah blah..."
astr,s,"a",)astringindirrection,now|is,,"To test string indirrection"
astringindirrection,s,"a",is,now|is|the|time,,"blah blah..."
foo,s,"a",a,,,"blah blah..."
mode,s,"ahq",a,,,"mode"
# To test is the library aborts if mode is set to batch:
#mode,s,"b",a,,,"mode"
```

Index