ParCompMark Reference Manual

0.1

Generated by Doxygen 1.4.6

Fri Jul 21 15:45:08 2006

Contents

Chapter 1

ParCompMark(p. 13) API Reference

1.1 What is this?

This is the complete API reference for **ParCompMark**(p. 13); contained within are the specifications for each class, attributes, and methods.

1.2 Manual

You can find the official **ParCompMark**(p. 13) manual `here`.

1.3 Building and installation of ParCompMark

You can find building and intallation instructions `here`.

1.4 Articles

Various (more advanced) `tutorials and articles` on how to use **ParCompMark**(p. 13).

1.5 Frequently Asked Questions

And `answers`.

Chapter 2

ParCompMark Namespace Index

2.1 ParCompMark Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

ParCompMark Hierarchical Index

3.1 ParCompMark Class Hierarchy

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

Chapter 4

ParCompMark Class Index

4.1 ParCompMark Class List

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

Chapter 5

ParCompMark File Index

5.1 ParCompMark File List

Here is a list of all files with brief descriptions:

Chapter 6

ParCompMark Namespace Documentation

6.1 ParCompMark Namespace Reference

6.1.1 Detailed Description

This namespace contains the classes of project **ParCompMark**(p. 13). The source files starts with `PCM` prefix. There is a unit test for this project called **ParCompMarkTest**(p. 16).

Classes

- class **Application**
- class **Buffer**
- class **Cluster**
- class **Container**
- class **Context**
- class **DummyLock**
- class **DynLoad**
- class **Exception**
- class **GLXGLContext**
- class **GLXRenderWindow**
- class **HandleClient**
- class **Host**
- class **Lock**
- class **Logger**
- class **Mutex**
- class **Name**
- class **Network**
- class **Node**
- class **OldContainer**
- class **OutputNode**
- class **Pointer**
- class **Process**
- class **Singleton**

- class **SqVM**
- class **Thread**
- class **Timer**
- class **XDisplay**

Typedefs

- typedef __u8 **u8**
- typedef __s8 **s8**
- typedef __u16 **u16**
- typedef __s16 **s16**
- typedef __u32 **u32**
- typedef __s32 **s32**
- typedef __u64 **u64**
- typedef __s64 **s64**
- typedef double **Real**

6.1.2 Typedef Documentation

6.1.2.1 typedef double ParCompMark::Real

Definition at line 48 of file PCMPremeditations.h.

6.1.2.2 typedef __s16 ParCompMark::s16

Definition at line 43 of file PCMPremeditations.h.

6.1.2.3 typedef __s32 ParCompMark::s32

Definition at line 45 of file PCMPremeditations.h.

6.1.2.4 typedef __s64 ParCompMark::s64

Definition at line 47 of file PCMPremeditations.h.

6.1.2.5 typedef __s8 ParCompMark::s8

Definition at line 41 of file PCMPremeditations.h.

6.1.2.6 typedef __u16 ParCompMark::u16

Definition at line 42 of file PCMPremeditations.h.

6.1.2.7 typedef __u32 ParCompMark::u32

Definition at line 44 of file PCMPremeditations.h.

6.1.2.8 typedef __u64 ParCompMark::u64

Definition at line 46 of file PCMPremeditations.h.

6.1.2.9 typedef __u8 ParCompMark::u8

Definition at line 40 of file PCMPremeditations.h.

6.2 ParCompMarkTest Namespace Reference

6.2.1 Detailed Description

This namespace contains the classes of project **ParCompMarkTest**(p. 16). The source files starts with `Test` prefix. This is a unit test project for project **ParCompMark**(p. 13).

Classes

- class **TestApplication**
- class **TestBuffer**
- class **TestCluster**
- class **TestContainer**
- class **TestContext**
- class **TestDummyLock**
- class **TestDynLoad**
- class **TestException**
- class **TestGLXGLContext**
- class **TestGLXRenderWindow**
- class **TestHandleClient**
- class **TestHost**
- class **TestLock**
- class **TestLogger**
- class **TestMutex**
- class **TestName**
- class **TestNetwork**
- class **TestNode**
- class **TestOldContainer**
- class **TestOutputNode**
- class **TestPointer**
- class **TestProcess**
- class **TestSingleton**
- class **TestSqVM**
- class **TestThread**
- class **TestTimer**
- class **TestXDisplay**
- struct **MyClass**

Typedefs

- typedef **Container**< **MyClass**, **Mutex** > **MyContainer**

Functions

- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestApplication**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestBuffer**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestCluster**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestContainer**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestContext**)

- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestDummyLock**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestDynLoad**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestException**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestGLXGLContext**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestGLXRenderWindow**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestHandleClient**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestHost**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestLock**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestLogger**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestMutex**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestName**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestNetwork**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestNode**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestOldContainer**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestOutputNode**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestPointer**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestProcess**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestSingleton**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestSqVM**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestThread**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestTimer**)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (**TestXDisplay**)

6.2.2 Typedef Documentation

6.2.2.1 typedef Container< MyClass, Mutex > ParCompMarkTest::MyContainer

Test container

Definition at line 40 of file TestContainer.cpp.

6.2.3 Function Documentation

6.2.3.1 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestXDisplay)

6.2.3.2 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestTimer)

6.2.3.3 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestThread)

6.2.3.4 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestSqVM)

6.2.3.5 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestSingleton)

6.2.3.6 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestProcess)

6.2.3.7 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestPointer)

6.2.3.8 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestOutputNode)

6.2.3.9 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestOldContainer)

6.2.3.10 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestNode)

6.2.3.11 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestNetwork)

6.2.3.12 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestName)

6.2.3.13 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestMutex)

6.2.3.14 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestLogger)

6.2.3.15 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestLock)

6.2.3.16 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestHost)

6.2.3.17 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestHandleClient)

6.2.3.18 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestGLXRenderWindow)

6.2.3.19 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestGLXGLContext)

6.2.3.20 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestException)

6.2.3.21 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestDynLoad)

6.2.3.22 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestDummyLock)

6.2.3.23 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestContext)

6.2.3.24 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestContainer)

6.2.3.25 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestCluster)

Method tests

void TestCluster::test_getHosts() { Cluster ∗c;

CPPUNIT_ASSERT_NO_THROW(c = new Cluster()); CPPUNIT_ASSERT_NO_THROW(c->getHosts()); printf("\ntestNumberElement: %d\n", c->getHosts()->getElementNumber()); }

6.2.3.26 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestBuffer)

6.2.3.27 ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION (TestApplication)

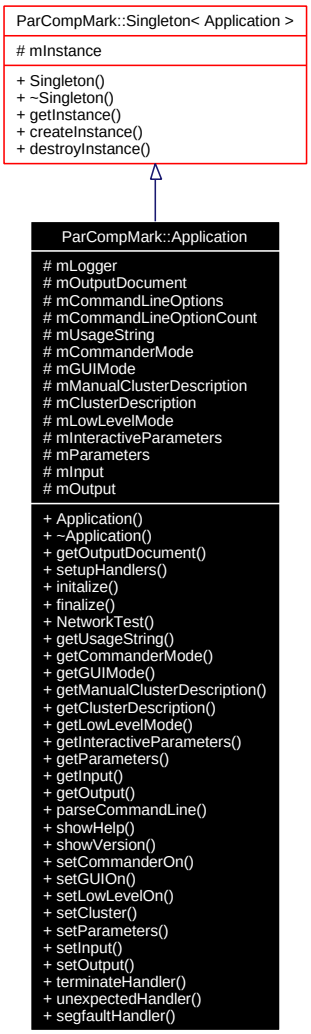
Chapter 7

ParCompMark Class Documentation

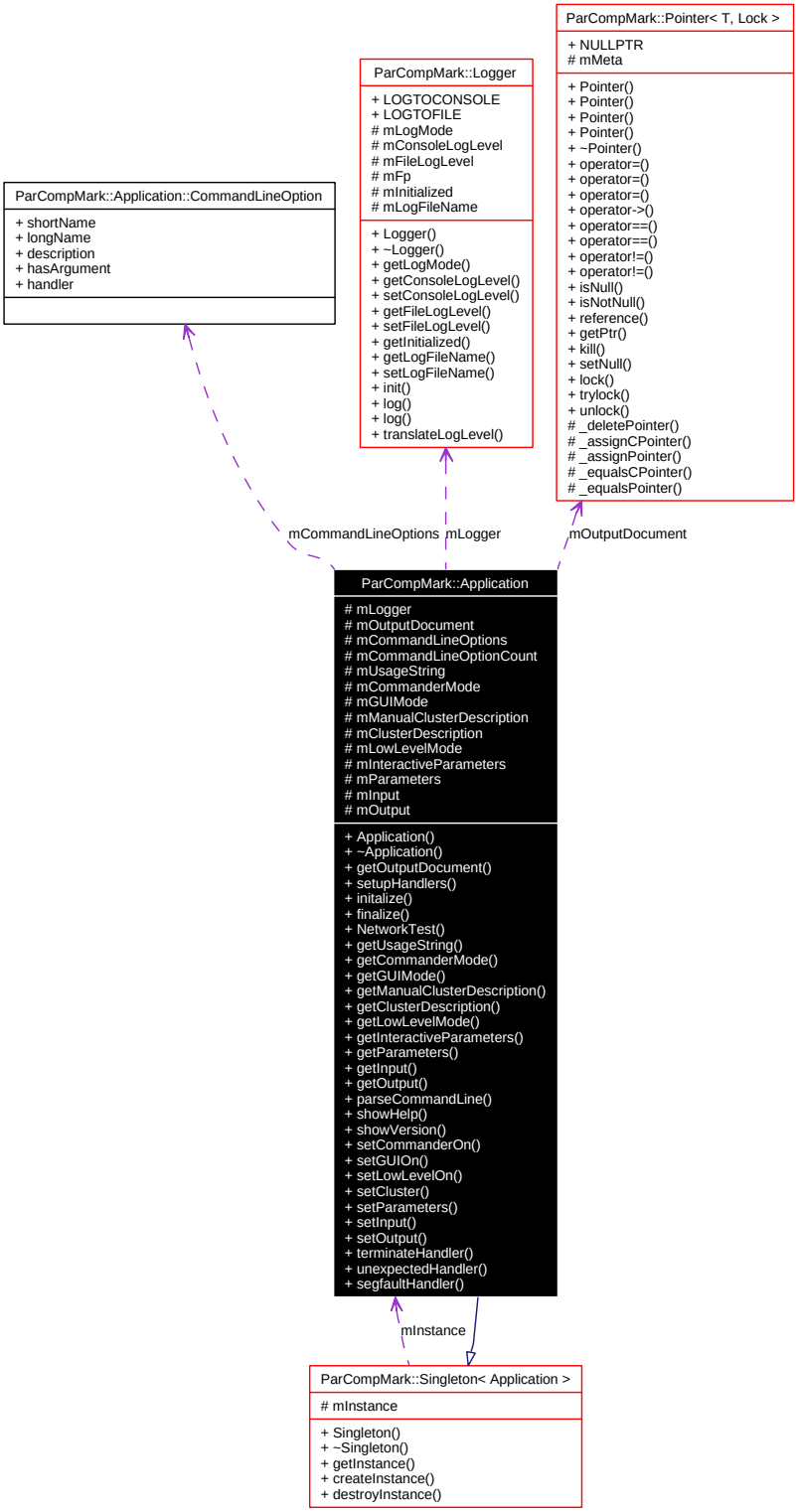
7.1 ParCompMark::Application Class Reference

`#include <PCMApplication.h>`

Inheritance diagram for ParCompMark::Application:



Collaboration diagram for ParCompMark::Application:



7.1.1 Detailed Description

This singleton class handles the application initializing, command line parsing, starting etc. tasks.

Definition at line 59 of file PCMApplication.h.

Public Member Functions

- **Application** ()
- virtual ∼**Application** ()
- **OutputNode::Pointer** & **getOutputDocument** ()
- virtual void **setupHandlers** () const
- virtual void **initalize** () const
- virtual void **finalize** () const
- virtual void **NetworkTest** ()

Static Public Member Functions

- static const std::string & **getUsageString** ()
- static const bool & **getCommanderMode** ()
- static const bool & **getGUIMode** ()
- static const bool & **getManualClusterDescription** ()
- static const std::string & **getClusterDescription** ()
- static const bool & **getLowLevelMode** ()
- static const bool & **getInteractiveParameters** ()
- static const std::string & **getParameters** ()
- static const std::string & **getInput** ()
- static const std::string & **getOutput** ()
- static void **parseCommandLine** (const **u32** &argc, const char ∗∗&argv)
- static void **showHelp** (const std::string &strarg)
- static void **showVersion** (const std::string &strarg)
- static void **setCommanderOn** (const std::string &strarg)
- static void **setGUIOn** (const std::string &strarg)
- static void **setLowLevelOn** (const std::string &strarg)
- static void **setCluster** (const std::string &strarg)
- static void **setParameters** (const std::string &strarg)
- static void **setInput** (const std::string &strarg)
- static void **setOutput** (const std::string &strarg)
- static void **terminateHandler** ()
- static void **unexpectedHandler** ()
- static void **segfaultHandler** (int value)

Protected Attributes

- **Logger** ∗ **mLogger**
- **OutputNode::Pointer mOutputDocument**

Static Protected Attributes

- static const **Application::CommandLineOption mCommandLineOptions** [ ]
- static const **u32 mCommandLineOptionCount** = sizeof(**mCommandLineOptions**) / sizeof(**mCommandLineOptions**[0])
- static const std::string **mUsageString** = "ParCompMark [options]"
- static bool **mCommanderMode**
- static bool **mGUIMode**
- static bool **mManualClusterDescription**
- static std::string **mClusterDescription**
- static bool **mLowLevelMode**
- static bool **mInteractiveParameters**
- static std::string **mParameters**
- static std::string **mInput**
- static std::string **mOutput**

Classes

- struct **CommandLineOption**

7.1.2 Constructor & Destructor Documentation

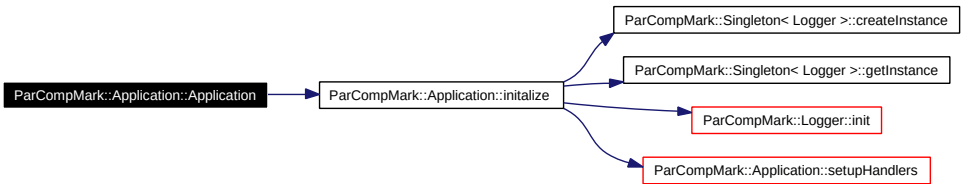
7.1.2.1 ParCompMark::Application::Application ()

Default constructor.

Definition at line 95 of file PCMApplication.cpp.

References initalize(), mCommanderMode, mGUIMode, mInput, mInteractiveParameters, mLowLevelMode, mManualClusterDescription, and mOutput.

Here is the call graph for this function:



7.1.2.2 ParCompMark::Application::~Application () [virtual]

The destructor. This class has virtual destructor.

Definition at line 129 of file PCMApplication.cpp.

References finalize().

Here is the call graph for this function:

7.1.3 Member Function Documentation

7.1.3.1 void ParCompMark::Application::finalize () const [virtual]

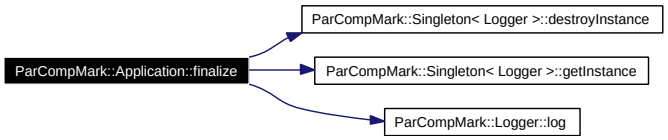
Finalize the PCM application.

Definition at line 332 of file PCMApplication.cpp.

References ParCompMark::Singleton< Logger >::destroyInstance(), ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Logger::log(), mLogger, and ParCompMark::Logger::NOTICE.

Referenced by terminateHandler(), and ~Application().

Here is the call graph for this function:



7.1.3.2 const std::string & ParCompMark::Application::getClusterDescription () [inline, static]

Remarks:
Getter of mClusterDescription.

Returns value of mClusterDescription.

Returns:
The value of mClusterDescription

Definition at line 527 of file PCMApplication.h.

7.1.3.3 const bool & ParCompMark::Application::getCommanderMode () [inline, static]

Remarks:
Getter of mCommanderMode.

Returns value of mCommanderMode.

Returns:
The value of mCommanderMode

Definition at line 506 of file PCMApplication.h.

7.1.3.4 const bool & ParCompMark::Application::getGUIMode () [inline, static]

Remarks:
Getter of mGUIMode.

Returns value of mGUIMode.

Returns:
The value of mGUIMode

Definition at line 513 of file PCMApplication.h.

7.1.3.5 const std::string & ParCompMark::Application::getInput () [inline, static]

Remarks:
Getter of mInput.

Returns value of mInput.

Returns:
The value of mInput

Definition at line 555 of file PCMApplication.h.

7.1.3.6 const bool & ParCompMark::Application::getInteractiveParameters () [inline, static]

Remarks:
Getter of mInteractiveParameters.

Returns value of mInteractiveParameters.

Returns:
The value of mInteractiveParameters

Definition at line 541 of file PCMApplication.h.

7.1.3.7 const bool & ParCompMark::Application::getLowLevelMode () [inline, static]

Remarks:
Getter of mLowLevelMode.

Returns value of mLowLevelMode.

Returns:
The value of mLowLevelMode

Definition at line 534 of file PCMApplication.h.

7.1.3.8 const bool & ParCompMark::Application::getManualClusterDescription () [inline, static]

Remarks:
Getter of mManualClusterDescription.

Returns value of mManualClusterDescription.

Returns:
The value of mManualClusterDescription

Definition at line 520 of file PCMApplication.h.

7.1.3.9 const std::string & ParCompMark::Application::getOutput () [inline, static]

Remarks:
Getter of mOutput.

Returns value of mOutput.

Returns:
The value of mOutput

Definition at line 562 of file PCMApplication.h.

7.1.3.10 OutputNode::Pointer & ParCompMark::Application::getOutputDocument () [inline]

Remarks:
Getter of mOutputDocument.

Returns value of mOutputDocument.

Returns:
The value of mOutputDocument

Definition at line 492 of file PCMApplication.h.

7.1.3.11 const std::string & ParCompMark::Application::getParameters () [inline, static]

Remarks:
Getter of mParameters.

Returns value of mParameters.

Returns:
The value of mParameters

Definition at line 548 of file PCMApplication.h.

7.1.3.12 const std::string & ParCompMark::Application::getUsageString () [inline, static]

Remarks:
Getter of mUsageString.

Returns value of mUsageString.

Returns:
The value of mUsageString

Definition at line 499 of file PCMApplication.h.

7.1.3.13 void ParCompMark::Application::initalize () const [virtual]

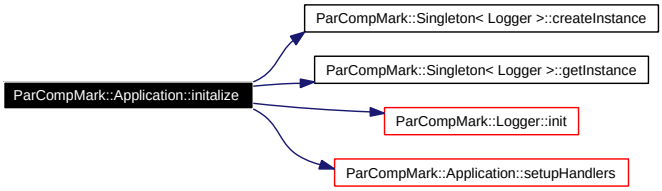
Initialize the PCM application.

Definition at line 320 of file PCMApplication.cpp.

References ParCompMark::Singleton< Logger >::createInstance(), ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Logger::init(), and setupHandlers().

Referenced by Application().

Here is the call graph for this function:



7.1.3.14 void ParCompMark::Application::NetworkTest () [virtual]

Network(p. 134) testing method.

Definition at line 343 of file PCMApplication.cpp.

References ParCompMark::Host::init().

Referenced by main().

Here is the call graph for this function:

7.1.3.15 void ParCompMark::Application::parseCommandLine (const u32 & *argc*, const char **& *argv*) [static]

Parse ANSI C command line.

Parameters:
← ***argc*** Number of command line arguments.

← ***argv*** Array of command line arguments.

Definition at line 141 of file PCMApplication.cpp.

Referenced by main().

7.1.3.16 void ParCompMark::Application::segfaultHandler (int *value*) [static]

Segmentation fault handler.

Parameters:
← ***value*** Signal parameter.

Definition at line 298 of file PCMApplication.cpp.

7.1.3.17 void ParCompMark::Application::setCluster (const std::string & *strarg*) [static]

Set cluster description file.

Parameters:
← ***strarg*** String argument.

Definition at line 250 of file PCMApplication.cpp.

References mClusterDescription, and mManualClusterDescription.

7.1.3.18 void ParCompMark::Application::setCommanderOn (const std::string & *strarg*) [static]

Set commander mode.

Parameters:
← ***strarg*** String argument (dummy).

Definition at line 229 of file PCMApplication.cpp.

References mCommanderMode.

7.1.3.19 void ParCompMark::Application::setGUIOn (const std::string & *strarg*) [static]

Set GUI mode.

Parameters:
← ***strarg*** String argument (dummy).

Definition at line 236 of file PCMApplication.cpp.

References mGUIMode.

7.1.3.20 void ParCompMark::Application::setInput (const std::string & *strarg*) [static]

Set input file.

Parameters:
← ***strarg*** String argument.

Definition at line 266 of file PCMApplication.cpp.

References mInput.

7.1.3.21 void ParCompMark::Application::setLowLevelOn (const std::string & *strarg*) [static]

Set low-level scripting mode.

Parameters:
← ***strarg*** String argument (dummy).

Definition at line 243 of file PCMApplication.cpp.

References mLowLevelMode.

7.1.3.22 void ParCompMark::Application::setOutput (const std::string & *strarg*) [static]

Set output file.

Parameters:
← ***strarg*** String argument.

Definition at line 273 of file PCMApplication.cpp.

References mOutput.

7.1.3.23 void ParCompMark::Application::setParameters (const std::string & *strarg*) [static]

Set parameters description file.

Parameters:
← ***strarg*** String argument.

Definition at line 258 of file PCMApplication.cpp.

References mInteractiveParameters, and mParameters.

7.1.3.24 void ParCompMark::Application::setupHandlers () const [virtual]

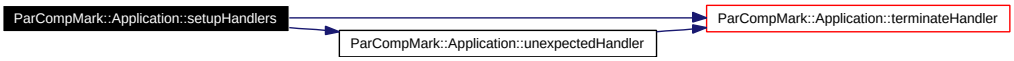
Setup special event handlers.

Definition at line 309 of file PCMApplication.cpp.

References terminateHandler(), and unexpectedHandler().

Referenced by initalize().

Here is the call graph for this function:



7.1.3.25 void ParCompMark::Application::showHelp (const std::string & *strarg*) [static]

Write help to std out.

Parameters:
← ***strarg*** String argument (dummy).

Definition at line 196 of file PCMApplication.cpp.

References mUsageString.

7.1.3.26 void ParCompMark::Application::showVersion (const std::string & *strarg*) [static]

Write version to std out.

Parameters:
← ***strarg*** String argument (dummy).

Definition at line 219 of file PCMApplication.cpp.

7.1.3.27 void ParCompMark::Application::terminateHandler () [static]

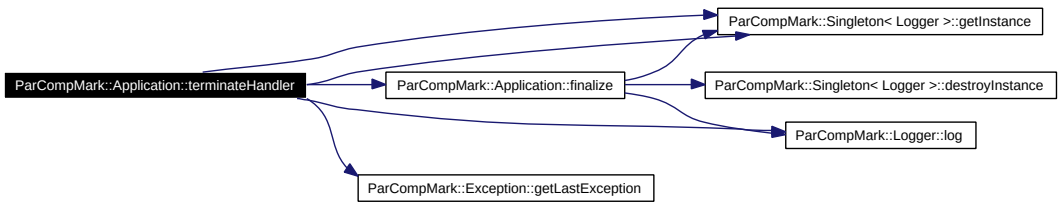
Abnormal termination handler.

Definition at line 280 of file PCMApplication.cpp.

References ParCompMark::Logger::FATAL, finalize(), ParCompMark::Singleton< Application >::getInstance(), ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Exception::getLastException(), and ParCompMark::Logger::log().

Referenced by setupHandlers(), and unexpectedHandler().

Here is the call graph for this function:



7.1.3.28 void ParCompMark::Application::unexpectedHandler () [static]

Unexpected exception handler.

Definition at line 291 of file PCMApplication.cpp.

References terminateHandler().

Referenced by setupHandlers().

Here is the call graph for this function:

7.1.4 Member Data Documentation

7.1.4.1 std::string ParCompMark::Application::mClusterDescription [static, protected]

Cluster(p. 46) description file name.

Remarks:
This is own attribute of this class.

Definition at line 149 of file PCMApplication.h.

Referenced by setCluster().

7.1.4.2 bool ParCompMark::Application::mCommanderMode [static, protected]

Indicates commander mode (default false).

Remarks:
This is own attribute of this class.

Definition at line 128 of file PCMApplication.h.

Referenced by Application(), and setCommanderOn().

7.1.4.3 const u32 ParCompMark::Application::mCommandLineOptionCount = sizeof(mCommandLineOptions) / sizeof(mCommandLineOptions[0]) [static, protected]

Number of command line options.

Remarks:
This is own attribute of this class.

Definition at line 108 of file PCMApplication.h.

7.1.4.4 const Application::CommandLineOption ParCompMark::Application::mCommandLineOptions [static, protected]

Initial value:

```
{
        {'h', "help ", "show help", false, Application::showHelp}
        ,
        {'v', "version ", "show version", false, Application::showVersion}
        ,
        {'c', "enable-commander ", "enable commander mode", false, Application::setCommanderOn}
        ,
        {'g', "enable-gui ", "enable GUI", false, Application::setGUIOn}
        ,
        {'C', "cluster ", "set cluster description", true, Application::setCluster}
        ,
        {'l', "enable-lowlevel ", "indicate the input script is a low-level script", false, Application::s
        ,
        {'p', "parameters ", "set input parameters from file", true, Application::setParameters}
        ,
        {'i', "input ", "set intput script file", true, Application::setInput}
        ,
        {'o', "output ", "set output filename", true, Application::setOutput}
        ,
 }
```

Command line options for **ParCompMark**(p. 13).

Remarks:
This is own attribute of this class.

Definition at line 101 of file PCMApplication.h.

7.1.4.5 bool ParCompMark::Application::mGUIMode [static, protected]

Indicates GUI mode (default false).

Remarks:
This is own attribute of this class.

Definition at line 135 of file PCMApplication.h.

Referenced by Application(), and setGUIOn().

7.1.4.6 std::string ParCompMark::Application::mInput [static, protected]

Input script file name.

Remarks:
This is own attribute of this class.

Definition at line 177 of file PCMApplication.h.

Referenced by Application(), and setInput().

7.1.4.7 bool ParCompMark::Application::mInteractiveParameters [static, protected]

Indicates interactive parameter settings (default true).

Remarks:
This is own attribute of this class.

Definition at line 163 of file PCMApplication.h.

Referenced by Application(), and setParameters().

7.1.4.8 Logger* ParCompMark::Application::mLogger [protected]

Logger(p. 118) object.

Remarks:
This is own attribute of this class.

Definition at line 197 of file PCMApplication.h.

Referenced by finalize().

7.1.4.9 bool ParCompMark::Application::mLowLevelMode [static, protected]

Indicates low-level scripting mode (default false).

Remarks:
This is own attribute of this class.

Definition at line 156 of file PCMApplication.h.

Referenced by Application(), and setLowLevelOn().

7.1.4.10 bool ParCompMark::Application::mManualClusterDescription [static, protected]

Indicates manual cluster description (default false).

Remarks:
This is own attribute of this class.

Definition at line 142 of file PCMApplication.h.

Referenced by Application(), and setCluster().

7.1.4.11 std::string ParCompMark::Application::mOutput [static, protected]

Output file name.

Remarks:
This is own attribute of this class.

Definition at line 184 of file PCMApplication.h.

Referenced by Application(), and setOutput().

7.1.4.12 OutputNode::Pointer ParCompMark::Application::mOutputDocument [protected]

Root of the output document.

Remarks:
This is own attribute of this class.

Definition at line 204 of file PCMApplication.h.

7.1.4.13 std::string ParCompMark::Application::mParameters [static, protected]

Parameters description file name.

Remarks:
This is own attribute of this class.

Definition at line 170 of file PCMApplication.h.

Referenced by setParameters().

7.1.4.14 const std::string ParCompMark::Application::mUsageString = "ParCompMark [options]" [static, protected]

Application(p. 21) usage string.

Remarks:
This is own attribute of this class.

Definition at line 115 of file PCMApplication.h.

Referenced by showHelp().

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

- **PCMApplication.h**
- **PCMApplication.cpp**

7.2 ParCompMark::Application::CommandLineOption Struct Reference

`#include <PCMApplication.h>`

7.2.1 Detailed Description

Structure describing a command line option for command line parsing.

Definition at line 76 of file PCMApplication.h.

Public Attributes

- **s8 shortName**
- std::string **longName**
- std::string **description**
- bool **hasArgument**
- void(∗ **handler**)(const std::string &)

7.2.2 Member Data Documentation

7.2.2.1 std::string ParCompMark::Application::CommandLineOption::description

Description

Definition at line 84 of file PCMApplication.h.

7.2.2.2 void(∗ ParCompMark::Application::CommandLineOption::handler)(const std::string &)

Argument handler method

7.2.2.3 bool ParCompMark::Application::CommandLineOption::hasArgument

The option has an argument

Definition at line 86 of file PCMApplication.h.

7.2.2.4 std::string ParCompMark::Application::CommandLineOption::longName

Long name

Definition at line 82 of file PCMApplication.h.

7.2.2.5 s8 ParCompMark::Application::CommandLineOption::shortName

Short name

Definition at line 80 of file PCMApplication.h.

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

- **PCMApplication.h**

7.3 ParCompMark::Buffer Class Reference

`#include <PCMBuffer.h>`

7.3.1 Detailed Description

Memory buffer class.

Definition at line 49 of file PCMBuffer.h.

Public Types

- typedef **Pointer**< **Buffer**, **Mutex** > **Pointer**

Public Member Functions

- **Buffer** ()
- virtual ~**Buffer** ()
- const PCuint & **getLeft** () const
- const PCuint & **getTop** () const
- const PCuint & **getWidth** () const
- const PCuint & **getHeight** () const
- const bool & **getOwnPointers** () const
- PCuint * **getColour** ()
- void **setColour** (PCuint *colour)
- void * **getDepth** ()
- void **setDepth** (void *depth)
- const PCint & **getDepthFormat** () const
- const PCint & **getOutputRowPixel** () const
- virtual void **init** (const int &left, const int &top, const int &width, const int &height, const int &depthFormat)

Protected Member Functions

- virtual void **freeBuffers** ()

Protected Attributes

- PCuint **mLeft**
- PCuint **mTop**
- PCuint **mWidth**
- PCuint **mHeight**
- bool **mOwnPointers**
- PCuint * **mColour**
- void * **mDepth**
- PCint **mDepthFormat**
- PCint **mOutputRowPixel**

7.3.2 Member Typedef Documentation

7.3.2.1 typedef Pointer< Buffer, Mutex > ParCompMark::Buffer::Pointer

Type for pointer on that class.

Definition at line 67 of file PCMBuffer.h.

7.3.3 Constructor & Destructor Documentation

7.3.3.1 ParCompMark::Buffer::Buffer ()

Default constructor.

Definition at line 38 of file PCMBuffer.cpp.

References mColour, mDepth, mHeight, mLeft, mOutputRowPixel, mOwnPointers, mTop, and mWidth.

7.3.3.2 ParCompMark::Buffer::~Buffer () [virtual]

The destructor. This class has virtual destructor.

Definition at line 64 of file PCMBuffer.cpp.

References freeBuffers().

Here is the call graph for this function:

7.3.4 Member Function Documentation

7.3.4.1 void ParCompMark::Buffer::freeBuffers () [protected, virtual]

Free mColour and mDepth buffers.

Definition at line 100 of file PCMBuffer.cpp.

References mColour, mDepth, mDepthFormat, and mOwnPointers.

Referenced by ~Buffer().

7.3.4.2 PCuint * ParCompMark::Buffer::getColour () [inline]

Remarks:

Getter of mColour.

Returns value of mColour.

Returns:

The value of mColour

Definition at line 374 of file PCMBuffer.h.

Referenced by ParCompMark::Process::runningProcess().

7.3.4.3 void * ParCompMark::Buffer::getDepth () [inline]

Remarks:
Getter of mDepth.

Returns value of mDepth.

Returns:
The value of mDepth

Definition at line 390 of file PCMBuffer.h.

7.3.4.4 const PCint & ParCompMark::Buffer::getDepthFormat () const [inline]

Remarks:
Getter of mDepthFormat.

Returns value of mDepthFormat.

Returns:
The value of mDepthFormat

Definition at line 406 of file PCMBuffer.h.

7.3.4.5 const PCuint & ParCompMark::Buffer::getHeight () const [inline]

Remarks:
Getter of mHeight.

Returns value of mHeight.

Returns:
The value of mHeight

Definition at line 360 of file PCMBuffer.h.

Referenced by ParCompMark::Process::runningProcess().

7.3.4.6 const PCuint & ParCompMark::Buffer::getLeft () const [inline]

Remarks:
Getter of mLeft.

Returns value of mLeft.

Returns:
The value of mLeft

Definition at line 339 of file PCMBuffer.h.

7.3.4.7 const PCint & ParCompMark::Buffer::getOutputRowPixel () const [inline]

Remarks:
Getter of mOutputRowPixel.

Returns value of mOutputRowPixel.

Returns:
The value of mOutputRowPixel

Definition at line 413 of file PCMBuffer.h.

7.3.4.8 const bool & ParCompMark::Buffer::getOwnPointers () const [inline]

Remarks:
Getter of mOwnPointers.

Returns value of mOwnPointers.

Returns:
The value of mOwnPointers

Definition at line 367 of file PCMBuffer.h.

7.3.4.9 const PCuint & ParCompMark::Buffer::getTop () const [inline]

Remarks:
Getter of mTop.

Returns value of mTop.

Returns:
The value of mTop

Definition at line 346 of file PCMBuffer.h.

7.3.4.10 const PCuint & ParCompMark::Buffer::getWidth () const [inline]

Remarks:
Getter of mWidth.

Returns value of mWidth.

Returns:
The value of mWidth

Definition at line 353 of file PCMBuffer.h.

Referenced by ParCompMark::Process::runningProcess().

7.3.4.11 void ParCompMark::Buffer::init (const int & *left*, const int & *top*, const int & *width*, const int & *height*, const int & *depthFormat*) [virtual]

Init the buffer.

Parameters:
← ***left***
← ***top***
← ***width***
← ***height***
← ***depthFormat***

Definition at line 75 of file PCMBuffer.cpp.

References mColour, mDepth, mDepthFormat, mHeight, mLeft, mTop, and mWidth.

7.3.4.12 void ParCompMark::Buffer::setColour (PCuint * *colour*) [inline]

Remarks:
Setter of mColour.

Sets value of mColour.

Parameters:
← ***colour*** The value of mColour

Definition at line 381 of file PCMBuffer.h.

7.3.4.13 void ParCompMark::Buffer::setDepth (void * *depth*) [inline]

Remarks:
Setter of mDepth.

Sets value of mDepth.

Parameters:
← ***depth*** The value of mDepth

Definition at line 397 of file PCMBuffer.h.

7.3.5 Member Data Documentation

7.3.5.1 PCuint* ParCompMark::Buffer::mColour [protected]

Colour information buffer.

Remarks:
This is own attribute of this class.

Definition at line 115 of file PCMBuffer.h.

Referenced by Buffer(), freeBuffers(), and init().

7.3.5.2 void* ParCompMark::Buffer::mDepth [protected]

Depth information buffer.

Remarks:
This is own attribute of this class.

Definition at line 122 of file PCMBuffer.h.

Referenced by Buffer(), freeBuffers(), and init().

7.3.5.3 PCint ParCompMark::Buffer::mDepthFormat [protected]

Depth format for deleting.

Remarks:
This is own attribute of this class.

Definition at line 129 of file PCMBuffer.h.

Referenced by freeBuffers(), and init().

7.3.5.4 PCuint ParCompMark::Buffer::mHeight [protected]

Remarks:
This is own attribute of this class.

Definition at line 101 of file PCMBuffer.h.

Referenced by Buffer(), and init().

7.3.5.5 PCuint ParCompMark::Buffer::mLeft [protected]

Remarks:
This is own attribute of this class.

Definition at line 80 of file PCMBuffer.h.

Referenced by Buffer(), and init().

7.3.5.6 PCint ParCompMark::Buffer::mOutputRowPixel [protected]

If the output is not the whole frame, this is the remainde pixels in a row.

Remarks:
This is own attribute of this class.

Definition at line 136 of file PCMBuffer.h.

Referenced by Buffer().

7.3.5.7 bool ParCompMark::Buffer::mOwnPointers [protected]

The **Buffer**(p. 39) is responsible for deallocation of mColour and mDepth.

Remarks:
This is own attribute of this class.

Definition at line 108 of file PCMBuffer.h.

Referenced by Buffer(), and freeBuffers().

7.3.5.8 PCuint ParCompMark::Buffer::mTop [protected]

Remarks:
This is own attribute of this class.

Definition at line 87 of file PCMBuffer.h.

Referenced by Buffer(), and init().

7.3.5.9 PCuint ParCompMark::Buffer::mWidth [protected]

Remarks:
This is own attribute of this class.

Definition at line 94 of file PCMBuffer.h.

Referenced by Buffer(), and init().

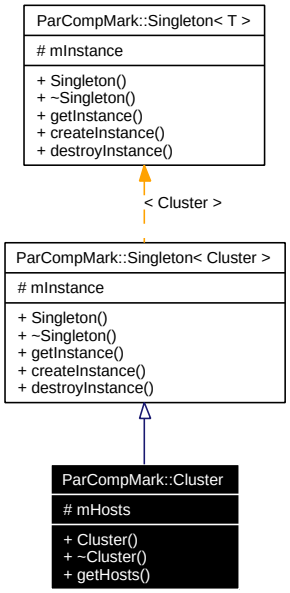
The documentation for this class was generated from the following files:

- **PCMBuffer.h**
- **PCMBuffer.cpp**

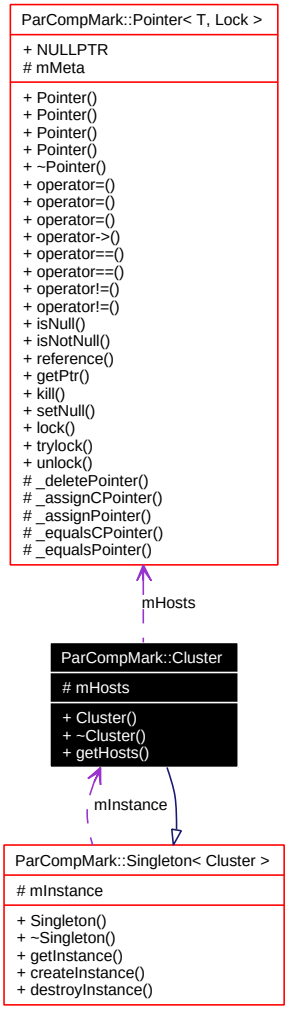
7.4 ParCompMark::Cluster Class Reference

`#include <PCMCluster.h>`

Inheritance diagram for ParCompMark::Cluster:



Collaboration diagram for ParCompMark::Cluster:



7.4.1 Detailed Description

Class contain hosts. Description of the physical grid.

Definition at line 61 of file PCMCluster.h.

Public Member Functions

- **Cluster** ()
- virtual ~**Cluster** ()
- **OldContainer::Pointer** & **getHosts** ()

Protected Attributes

- **OldContainer::Pointer mHosts**

7.4.2 Constructor & Destructor Documentation

7.4.2.1 ParCompMark::Cluster::Cluster ()

Default constructor.

Definition at line 39 of file PCMCluster.cpp.

References mHosts.

7.4.2.2 ParCompMark::Cluster::~Cluster () [virtual]

The destructor. This class has virtual destructor.

Definition at line 49 of file PCMCluster.cpp.

7.4.3 Member Function Documentation

7.4.3.1 OldContainer::Pointer & ParCompMark::Cluster::getHosts () [inline]

Remarks:
Getter of mHosts.

Returns value of mHosts.

Returns:
The value of mHosts

Definition at line 151 of file PCMCluster.h.

Referenced by ParCompMark::Network::buildCluster(), and ParCompMark::HandleClient::task().

7.4.4 Member Data Documentation

7.4.4.1 OldContainer::Pointer ParCompMark::Cluster::mHosts [protected]

Host(p. 112) map.

Remarks:
This is own attribute of this class.

Definition at line 82 of file PCMCluster.h.

Referenced by Cluster().

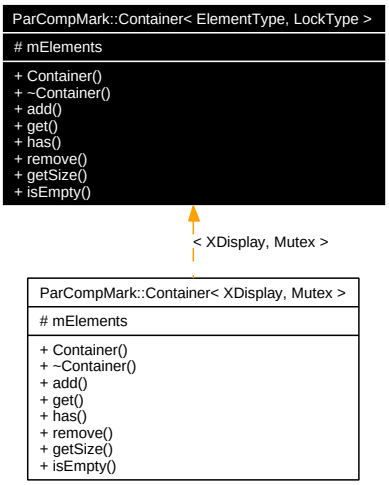
The documentation for this class was generated from the following files:

- **PCMCluster.h**
- **PCMCluster.cpp**

7.5 ParCompMark::Container< ElementType, LockType > Class Template Reference

`#include <PCMContainer.h>`

Inheritance diagram for ParCompMark::Container< ElementType, LockType >:



7.5.1 Detailed Description

template<class ElementType, class LockType> class ParCompMark::Container< ElementType, LockType >

String addressed map of typed smart pointers.

Definition at line 52 of file PCMContainer.h.

Public Types

- typedef **Pointer**< **Container**< ElementType, LockType >, LockType > **Pointer**
- typedef ElementType::Pointer **ElementPointer**
- typedef std::map< std::string, **ElementPointer** > **ElementsMap**
- typedef ElementsMap::iterator **Iterator**

Public Member Functions

- **Container** ()
- virtual ~**Container** ()
- virtual void **add** (std::string name, **ElementPointer** element)
- virtual **ElementPointer get** (const std::string &name)
- virtual bool **has** (const std::string &name)
- virtual void **remove** (const std::string &name)
- virtual **u32 getSize** ()
- virtual bool **isEmpty** ()

Protected Attributes

- **ElementsMap mElements**

7.5.2 Member Typedef Documentation

7.5.2.1 template<class ElementType, class LockType> typedef ElementType::Pointer ParCompMark::Container< ElementType, LockType >::ElementPointer

Type definition for pointer on elements.

Definition at line 74 of file PCMContainer.h.

7.5.2.2 template<class ElementType, class LockType> typedef std::map< std::string, ElementPointer > ParCompMark::Container< ElementType, LockType >::ElementsMap

Type definition for map of elements.

Definition at line 78 of file PCMContainer.h.

7.5.2.3 template<class ElementType, class LockType> typedef ElementsMap::iterator ParCompMark::Container< ElementType, LockType >::Iterator

Type definition for iterator on map of elements.

Definition at line 81 of file PCMContainer.h.

7.5.2.4 template<class ElementType, class LockType> typedef Pointer< Container < ElementType, LockType >, LockType > ParCompMark::Container< ElementType, LockType >::Pointer

Type for pointer on this class.

Definition at line 71 of file PCMContainer.h.

7.5.3 Constructor & Destructor Documentation

7.5.3.1 template<class ElementType, class LockType> ParCompMark::Container< ElementType, LockType >::Container () [inline]

Default constructor.

Definition at line 191 of file PCMContainer.h.

7.5.3.2 template<class ElementType, class LockType> ParCompMark::Container< ElementType, LockType >::~Container () [inline, virtual]

The destructor. This class has virtual destructor.

Definition at line 200 of file PCMContainer.h.

7.5.4 Member Function Documentation

7.5.4.1 template<class ElementType, class LockType> void ParCompMark::Container< ElementType, LockType >::add (std::string *name*, ElementPointer *element*) [virtual]

Add an element.

Parameters:

← ***name*** **Name**(p. 131) of the element

← ***element*** Element

Definition at line 212 of file PCMContainer.h.

7.5.4.2 template<class ElementType, class LockType> ElementType::Pointer ParCompMark::Container< ElementType, LockType >::get (const std::string & *name*) [virtual]

Get an element by name.

Parameters:

← ***name*** **Name**(p. 131) of the element

Returns:

Pointer(p. 170) to the element

Definition at line 221 of file PCMContainer.h.

Referenced by ParCompMark::Host::openXDisplay().

7.5.4.3 template<class ElementType, class LockType> u32 ParCompMark::Container< ElementType, LockType >::getSize () [virtual]

Return the number of elements.

Returns:

Pointer(p. 170) to the element

Definition at line 246 of file PCMContainer.h.

7.5.4.4 template<class ElementType, class LockType> bool ParCompMark::Container< ElementType, LockType >::has (const std::string & *name*) [virtual]

Search for an element by name.

Parameters:

← ***name*** **Name**(p. 131) of the element

Returns:

The container has the element.

Definition at line 230 of file PCMContainer.h.

Referenced by ParCompMark::Host::openXDisplay().

7.5.4.5 template<class ElementType, class LockType> bool ParCompMark::Container< ElementType, LockType >::isEmpty () [virtual]

Return true if the container is empty.

Returns:
True if the container is empty

Definition at line 253 of file PCMContainer.h.

7.5.4.6 template<class ElementType, class LockType> void ParCompMark::Container< ElementType, LockType >::remove (const std::string & *name*) [virtual]

Remove an element by name.

Parameters:
← ***name*** **Name**(p. 131) of the element

Definition at line 238 of file PCMContainer.h.

7.5.5 Member Data Documentation

7.5.5.1 template<class ElementType, class LockType> ElementsMap ParCompMark::Container< ElementType, LockType >::mElements [protected]

Map of elements.

Remarks:
This is own attribute of this class.

Definition at line 94 of file PCMContainer.h.

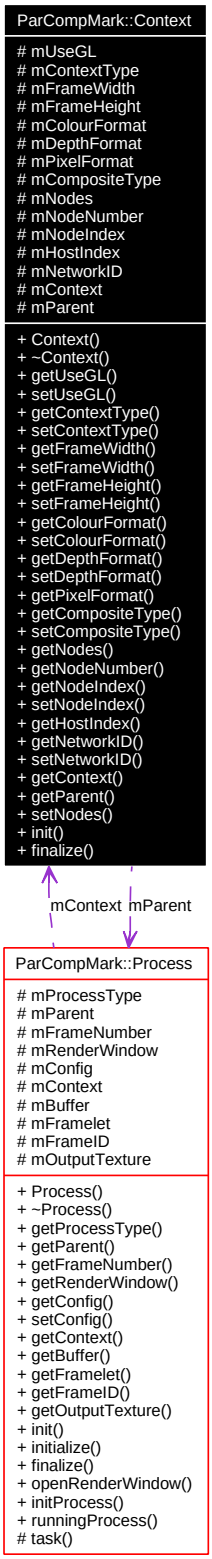
The documentation for this class was generated from the following file:

- **PCMContainer.h**

7.6 ParCompMark::Context Class Reference

`#include <PCMContext.h>`

Collaboration diagram for ParCompMark::Context:


ParCompMark::Context
mUseGL
mContextType
mFrameWidth
mFrameHeight
mColourFormat
mDepthFormat
mPixelFormat
mCompositeType
mNodes
mNodeNumber
mNodeIndex
mHostIndex
mNetworkID
mContext
mParent
+ Context()
+ ~Context()
+ getUseGL()
+ setUseGL()
+ getContextType()
+ setContextType()
+ getFrameWidth()
+ setFrameWidth()
+ getFrameHeight()
+ setFrameHeight()
+ getColourFormat()
+ setColourFormat()
+ getDepthFormat()
+ setDepthFormat()
+ getPixelFormat()
+ getCompositeType()
+ setCompositeType()
+ getNodes()
+ getNodeNumber()
+ getNodeIndex()
+ setNodeIndex()
+ getHostIndex()
+ getNetworkID()
+ setNetworkID()
+ getContext()
+ getParent()
+ setNodes()
+ init()
+ finalize()
mContext
mParent
ParCompMark::Process
mProcessType
mParent
mFrameNumber
mRenderWindow
mConfig
mContext
mBuffer
mFramelet
mFrameID
mOutputTexture
+ Process()
+ ~Process()
+ getProcessType()
+ getParent()
+ getFrameNumber()
+ getRenderWindow()
+ getConfig()
+ setConfig()
+ getContext()
+ getBuffer()
+ getFramelet()
+ getFrameID()
+ getOutputTexture()
+ init()
+ initialize()
+ finalize()
+ openRenderWindow()
+ initProcess()
+ runningProcess()
# task()

7.6.1 Detailed Description

Class contain PC context information.

Definition at line 55 of file PCMContext.h.

Public Types

- enum **ContextType** { **LOCAL**, **GLOBAL** }

Public Member Functions

- **Context** (**Process** ∗parent)
- virtual ∼**Context** ()
- const bool & **getUseGL** () const
- void **setUseGL** (const bool &usegl)
- const **Context::ContextType** & **getContextType** () const
- void **setContextType** (const **Context::ContextType** &contexttype)
- const PCint & **getFrameWidth** () const
- void **setFrameWidth** (const PCint &framewidth)
- const PCint & **getFrameHeight** () const
- void **setFrameHeight** (const PCint &frameheight)
- const PCint & **getColourFormat** () const
- void **setColourFormat** (const PCint &colourformat)
- const PCint & **getDepthFormat** () const
- void **setDepthFormat** (const PCint &depthformat)
- const PCint & **getPixelFormat** () const
- const PCint & **getCompositeType** () const
- void **setCompositeType** (const PCint &compositetype)
- char ∗∗ **getNodes** () const
- const PCint & **getNodeNumber** () const
- const int & **getNodeIndex** () const
- void **setNodeIndex** (const int &nodeindex)
- const PCint & **getHostIndex** () const
- const PCint & **getNetworkID** () const
- void **setNetworkID** (const PCint &networkid)
- const PCcontext & **getContext** () const
- **Process** ∗ **getParent** ()
- virtual void **setNodes** (const std::string &nodes)
- virtual void **init** ()
- virtual void **finalize** ()

Protected Attributes

- bool **mUseGL**
- **ContextType mContextType**
- PCint **mFrameWidth**
- PCint **mFrameHeight**
- PCint **mColourFormat**
- PCint **mDepthFormat**

- PCint **mPixelFormat**
- PCint **mCompositeType**
- char ∗∗ **mNodes**
- PCint **mNodeNumber**
- int **mNodeIndex**
- PCint **mHostIndex**
- PCint **mNetworkID**
- PCcontext **mContext**
- **Process** ∗ **mParent**

7.6.2 Member Enumeration Documentation

7.6.2.1 enum ParCompMark::Context::ContextType

The control type of PC context (Local, Global).

Enumerator:

LOCAL Local context.

GLOBAL Global context.

Definition at line 72 of file PCMContext.h.

7.6.3 Constructor & Destructor Documentation

7.6.3.1 ParCompMark::Context::Context (Process ∗ *parent*)

Create **Context**(p. 53). Normally **Process**(p. 182) calles this constructor.

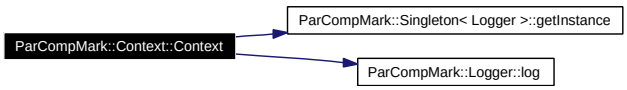
Parameters:

← ***parent*** Parent host

Definition at line 41 of file PCMContext.cpp.

References Assert, ParCompMark::Singleton< Logger >::getInstance(), LOCAL, ParCompMark::Logger::log(), mColourFormat, mCompositeType, mContextType, mDepthFormat, mFrameHeight, mFrameWidth, mHostIndex, mNetworkID, mNodeNumber, mNodes, mPixelFormat, mUseGL, and ParCompMark::Logger::NOTICE.

Here is the call graph for this function:



7.6.3.2 ParCompMark::Context::∼Context () [virtual]

The destructor. This class has virtual destructor.

Definition at line 83 of file PCMContext.cpp.

References mNodeNumber, and mNodes.

7.6.4 Member Function Documentation

7.6.4.1 void ParCompMark::Context::finalize () [virtual]

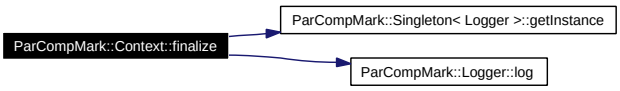
Finalize the PC context.

Definition at line 290 of file PCMContext.cpp.

References Except, ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Logger::log(), mContext, and ParCompMark::Logger::NOTICE.

Referenced by ParCompMark::Process::finalize().

Here is the call graph for this function:



7.6.4.2 const PCint & ParCompMark::Context::getColourFormat () const [inline]

Remarks:
Getter of mColourFormat.

Returns value of mColourFormat.

Returns:
The value of mColourFormat

Definition at line 592 of file PCMContext.h.

7.6.4.3 const PCint & ParCompMark::Context::getCompositeType () const [inline]

Remarks:
Getter of mCompositeType.

Returns value of mCompositeType.

Returns:
The value of mCompositeType

Definition at line 627 of file PCMContext.h.

7.6.4.4 const PCcontext & ParCompMark::Context::getContext () const [inline]

Remarks:
Getter of mContext.

Returns value of mContext.

Returns:
The value of mContext

Definition at line 690 of file PCMContext.h.

Referenced by ParCompMark::Process::task().

7.6.4.5 const Context::ContextType & ParCompMark::Context::getContextType () const [inline]

Remarks:
Getter of mContextType.

Returns value of mContextType.

Returns:
The value of mContextType

Definition at line 550 of file PCMContext.h.

7.6.4.6 const PCint & ParCompMark::Context::getDepthFormat () const [inline]

Remarks:
Getter of mDepthFormat.

Returns value of mDepthFormat.

Returns:
The value of mDepthFormat

Definition at line 606 of file PCMContext.h.

7.6.4.7 const PCint & ParCompMark::Context::getFrameHeight () const [inline]

Remarks:
Getter of mFrameHeight.

Returns value of mFrameHeight.

Returns:
The value of mFrameHeight

Definition at line 578 of file PCMContext.h.

7.6.4.8 const PCint & ParCompMark::Context::getFrameWidth () const [inline]

Remarks:
Getter of mFrameWidth.

Returns value of mFrameWidth.

Returns:
The value of mFrameWidth

Definition at line 564 of file PCMContext.h.

7.6.4.9 const PCint & ParCompMark::Context::getHostIndex () const [inline]

Remarks:
Getter of mHostIndex.

Returns value of mHostIndex.

Returns:
The value of mHostIndex

Definition at line 669 of file PCMContext.h.

7.6.4.10 const PCint & ParCompMark::Context::getNetworkID () const [inline]

Remarks:
Getter of mNetworkID.

Returns value of mNetworkID.

Returns:
The value of mNetworkID

Definition at line 676 of file PCMContext.h.

7.6.4.11 const int & ParCompMark::Context::getNodeIndex () const [inline]

Remarks:
Getter of mNodeIndex.

Returns value of mNodeIndex.

Returns:
The value of mNodeIndex

Definition at line 655 of file PCMContext.h.

7.6.4.12 const PCint & ParCompMark::Context::getNodeNumber () const [inline]

Remarks:
Getter of mNodeNumber.

Returns value of mNodeNumber.

Returns:
The value of mNodeNumber

Definition at line 648 of file PCMContext.h.

7.6.4.13 char ** ParCompMark::Context::getNodes () const [inline]

Remarks:
Getter of mNodes.

Returns value of mNodes.

Returns:
The value of mNodes

Definition at line 641 of file PCMContext.h.

7.6.4.14 Process * ParCompMark::Context::getParent () [inline]

Remarks:
Getter of mParent.

Returns value of mParent.

Returns:
The value of mParent

Definition at line 697 of file PCMContext.h.

7.6.4.15 const PCint & ParCompMark::Context::getPixelFormat () const [inline]

Remarks:
Getter of mPixelFormat.

Returns value of mPixelFormat.

Returns:
The value of mPixelFormat

Definition at line 620 of file PCMContext.h.

7.6.4.16 const bool & ParCompMark::Context::getUseGL () const [inline]

Remarks:
Getter of mUseGL.

Returns value of mUseGL.

Returns:
The value of mUseGL

Definition at line 536 of file PCMContext.h.

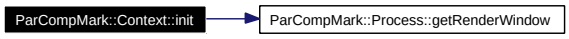
7.6.4.17 void ParCompMark::Context::init () [virtual]

Init the PC context.

Definition at line 136 of file PCMContext.cpp.

References Except, ParCompMark::Process::getRenderWindow(), GLOBAL, mCompositeType, mContext, mContextType, mFrameHeight, mFrameWidth, mNetworkID, mNodeIndex, mNodes, mParent, mPixelFormat, and mUseGL.

Here is the call graph for this function:



7.6.4.18 void ParCompMark::Context::setColourFormat (const PCint & *colourformat*) [inline]

Remarks:
Setter of mColourFormat.

Sets value of mColourFormat.

Parameters:
← ***colourformat*** The value of mColourFormat

Definition at line 599 of file PCMContext.h.

7.6.4.19 void ParCompMark::Context::setCompositeType (const PCint & *compositetype*) [inline]

Remarks:
Setter of mCompositeType.

Sets value of mCompositeType.

Parameters:
← ***compositetype*** The value of mCompositeType

Definition at line 634 of file PCMContext.h.

7.6.4.20 void ParCompMark::Context::setContextType (const Context::ContextType & *contexttype*) [inline]

Remarks:
Setter of mContextType.

Sets value of mContextType.

Parameters:
← ***contexttype*** The value of mContextType

Definition at line 557 of file PCMContext.h.

7.6.4.21 void ParCompMark::Context::setDepthFormat (const PCint & *depthformat*) [inline]

Remarks:
Setter of mDepthFormat.

Sets value of mDepthFormat.

Parameters:
← ***depthformat*** The value of mDepthFormat

Definition at line 613 of file PCMContext.h.

7.6.4.22 void ParCompMark::Context::setFrameHeight (const PCint & *frameheight*) [inline]

Remarks:
Setter of mFrameHeight.

Sets value of mFrameHeight.

Parameters:
← ***frameheight*** The value of mFrameHeight

Definition at line 585 of file PCMContext.h.

7.6.4.23 void ParCompMark::Context::setFrameWidth (const PCint & *framewidth*) [inline]

Remarks:
Setter of mFrameWidth.

Sets value of mFrameWidth.

Parameters:
← ***framewidth*** The value of mFrameWidth

Definition at line 571 of file PCMContext.h.

7.6.4.24 void ParCompMark::Context::setNetworkID (const PCint & *networkid*) [inline]

Remarks:
Setter of mNetworkID.

Sets value of mNetworkID.

Parameters:
← ***networkid*** The value of mNetworkID

Definition at line 683 of file PCMContext.h.

7.6.4.25 void ParCompMark::Context::setNodeIndex (const int & *nodeindex*) [inline]

Remarks:
Setter of mNodeIndex.

Sets value of mNodeIndex.

Parameters:
← ***nodeindex*** The value of mNodeIndex

Definition at line 662 of file PCMContext.h.

7.6.4.26 void ParCompMark::Context::setNodes (const std::string & *nodes*) [virtual]

Create nodes char struct from string.

Parameters:
← ***nodes*** Nodes description.

Definition at line 104 of file PCMContext.cpp.

References mNodeNumber, and mNodes.

7.6.4.27 void ParCompMark::Context::setUseGL (const bool & *usegl*) [inline]

Remarks:
Setter of mUseGL.

Sets value of mUseGL.

Parameters:
← ***usegl*** The value of mUseGL

Definition at line 543 of file PCMContext.h.

7.6.5 Member Data Documentation

7.6.5.1 PCint ParCompMark::Context::mColourFormat [protected]

Colour format of rendered frame by the context.

Remarks:
This is own attribute of this class.

Definition at line 121 of file PCMContext.h.

Referenced by Context().

7.6.5.2 PCint ParCompMark::Context::mCompositeType [protected]

Type of composition.

Remarks:
This is own attribute of this class.

Definition at line 142 of file PCMContext.h.

Referenced by Context(), and init().

7.6.5.3 PCcontext ParCompMark::Context::mContext [protected]

The PC context.

Remarks:
This is own attribute of this class.

Definition at line 184 of file PCMContext.h.

Referenced by finalize(), and init().

7.6.5.4 ContextType ParCompMark::Context::mContextType [protected]

Type of the context.

Remarks:
This is own attribute of this class.

Definition at line 100 of file PCMContext.h.

Referenced by Context(), and init().

7.6.5.5 PCint ParCompMark::Context::mDepthFormat [protected]

Depth format of rendered frame by the context.

Remarks:
This is own attribute of this class.

Definition at line 128 of file PCMContext.h.

Referenced by Context().

7.6.5.6 PCint ParCompMark::Context::mFrameHeight [protected]

Height of rendered frame by the context.

Remarks:
This is own attribute of this class.

Definition at line 114 of file PCMContext.h.

Referenced by Context(), and init().

7.6.5.7 PCint ParCompMark::Context::mFrameWidth [protected]

Width of rendered frame by the context.

Remarks:
This is own attribute of this class.

Definition at line 107 of file PCMContext.h.

Referenced by Context(), and init().

7.6.5.8 PCint ParCompMark::Context::mHostIndex [protected]

The own host index.

Remarks:
This is own attribute of this class.

Definition at line 170 of file PCMContext.h.

Referenced by Context().

7.6.5.9 PCint ParCompMark::Context::mNetworkID [protected]

The network ID.

Remarks:
This is own attribute of this class.

Definition at line 177 of file PCMContext.h.

Referenced by Context(), and init().

7.6.5.10 int ParCompMark::Context::mNodeIndex [protected]

Our place in the mNodes list.

Remarks:
This is own attribute of this class.

Definition at line 163 of file PCMContext.h.

Referenced by init().

7.6.5.11 PCint ParCompMark::Context::mNodeNumber [protected]

Our place in the mNodes list.

Remarks:
This is own attribute of this class.

Definition at line 156 of file PCMContext.h.

Referenced by Context(), setNodes(), and ~Context().

7.6.5.12 char** ParCompMark::Context::mNodes [protected]

Nodes in the context.

Remarks:
This is own attribute of this class.

Definition at line 149 of file PCMContext.h.

Referenced by Context(), init(), setNodes(), and ~Context().

7.6.5.13 Process* ParCompMark::Context::mParent [protected]

Parent **Process**(p. 182) of this **Context**(p. 53).

Remarks:
This attribute references an attribute.

Definition at line 191 of file PCMContext.h.

Referenced by init().

7.6.5.14 PCint ParCompMark::Context::mPixelFormat [protected]

The pixel format. Or link between depth and colour format.

Remarks:
This is own attribute of this class.

Definition at line 135 of file PCMContext.h.

Referenced by Context(), and init().

7.6.5.15 bool ParCompMark::Context::mUseGL [protected]

Use graphics card video memory (OpenGL).

Remarks:
This is own attribute of this class.

Definition at line 93 of file PCMContext.h.

Referenced by Context(), and init().

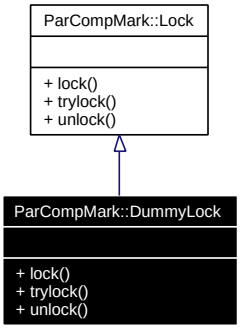
The documentation for this class was generated from the following files:

- **PCMContext.h**
- **PCMContext.cpp**

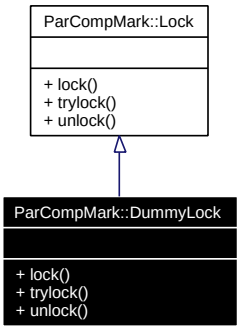
7.7 ParCompMark::DummyLock Class Reference

`#include <PCMDummyLock.h>`

Inheritance diagram for ParCompMark::DummyLock:



Collaboration diagram for ParCompMark::DummyLock:



7.7.1 Detailed Description

Dummy lock implementing the **Lock**(p. 116) interface. Does not do anything.

Definition at line 47 of file PCMDummyLock.h.

Public Member Functions

- virtual void **lock** ()
- virtual bool **trylock** ()
- virtual void **unlock** ()

7.7.2 Member Function Documentation

7.7.2.1 void ParCompMark::DummyLock::lock () [inline, virtual]

Lock(p. 116) the lock. Does not do anything.

Implements **ParCompMark::Lock** (p. 116).

Definition at line 113 of file PCMDummyLock.h.

7.7.2.2 bool ParCompMark::DummyLock::trylock () [inline, virtual]

Try locking the lock. Does not do anything. Always return true.

Returns:
True if the locking was successful. Always true.

Implements **ParCompMark::Lock** (p. 116).

Definition at line 120 of file PCMDummyLock.h.

7.7.2.3 void ParCompMark::DummyLock::unlock () [inline, virtual]

Unlock the lock. Does not do anything.

Implements **ParCompMark::Lock** (p. 117).

Definition at line 129 of file PCMDummyLock.h.

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

- **PCMDummyLock.h**

7.8 ParCompMark::DynLoad Class Reference

`#include <PCMDynLoad.h>`

7.8.1 Detailed Description

Dynamic load library.

Definition at line 53 of file PCMDynLoad.h.

Public Member Functions

- **DynLoad** (const std::string &libName)
- virtual ~**DynLoad** ()
- void * **getHandle** () const
- const std::string & **getLibraryName** () const
- virtual void * **getFunc** (const std::string &funcName) const

Protected Member Functions

- virtual void **load** ()
- virtual void **unload** ()

Protected Attributes

- void * **mHandle**
- std::string **mLibraryName**

7.8.2 Constructor & Destructor Documentation

7.8.2.1 ParCompMark::DynLoad::DynLoad (const std::string & *libName*)

Create a Dynamic load class, and load the libName named library.

Parameters:

← ***libName*** **Name**(p. 131) of the loaded library

Definition at line 38 of file PCMDynLoad.cpp.

References load(), and mLibraryName.

Here is the call graph for this function:

7.8.2.2 ParCompMark::DynLoad::~DynLoad () [virtual]

The destructor. This class has virtual destructor.

Definition at line 51 of file PCMDynLoad.cpp.

References mHandle, and unload().

Here is the call graph for this function:

7.8.3 Member Function Documentation

7.8.3.1 void * ParCompMark::DynLoad::getFunc (const std::string & *funcName*) const [virtual]

Get a function pointer from dynamic library by name(symbol).

Parameters:
← ***funcName*** **Name**(p. 131) of function

Returns:
Pointer(p. 170) to the function

Definition at line 66 of file PCMDynLoad.cpp.

References Assert, Except, and mHandle.

Referenced by ParCompMarkTest::TestDynLoad::test_getFunc_cstd__string().

7.8.3.2 void * ParCompMark::DynLoad::getHandle () const [inline]

Remarks:
Getter of mHandle.

Returns value of mHandle.

Returns:
The value of mHandle

Definition at line 187 of file PCMDynLoad.h.

7.8.3.3 const std::string & ParCompMark::DynLoad::getLibraryName () const [inline]

Remarks:
Getter of mLibraryName.

Returns value of mLibraryName.

Returns:
The value of mLibraryName

Definition at line 194 of file PCMDynLoad.h.

7.8.3.4 void ParCompMark::DynLoad::load () [protected, virtual]

Load a library. Called by constructor.

Definition at line 90 of file PCMDynLoad.cpp.

References Except, mHandle, and mLibraryName.

Referenced by DynLoad().

7.8.3.5 void ParCompMark::DynLoad::unload () [protected, virtual]

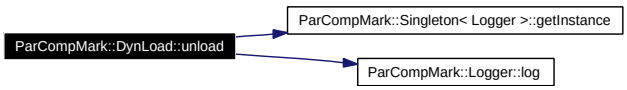
Load a library. Called by constructor.

Definition at line 102 of file PCMDynLoad.cpp.

References Except, ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Logger::log(), mHandle, mLibraryName, and ParCompMark::Logger::NOTICE.

Referenced by ~DynLoad().

Here is the call graph for this function:



7.8.4 Member Data Documentation

7.8.4.1 void* ParCompMark::DynLoad::mHandle [protected]

The dynamic library handler.

Remarks:
This is own attribute of this class.

Definition at line 74 of file PCMDynLoad.h.

Referenced by getFunc(), load(), unload(), and ~DynLoad().

7.8.4.2 std::string ParCompMark::DynLoad::mLibraryName [protected]

The dynamic library name.

Remarks:
This is own attribute of this class.

Definition at line 81 of file PCMDynLoad.h.

Referenced by DynLoad(), load(), and unload().

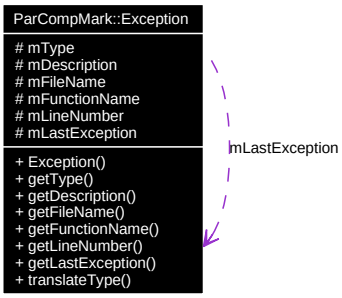
The documentation for this class was generated from the following files:

- **PCMDynLoad.h**
- **PCMDynLoad.cpp**

7.9 ParCompMark::Exception Class Reference

`#include <PCMException.h>`

Collaboration diagram for ParCompMark::Exception:



7.9.1 Detailed Description

Provides information about an internal error.

Remarks:

An application using PCM that the exceptions are caught, so all PCM functions should occur within a try{} catch(PCM::Exception& e) {} block.

Definition at line 62 of file PCMException.h.

Public Types

- enum **ExceptionType** {

  INTERNAL_ERROR, **NULL_POINTER_ERROR**, **INVALID_NAME_ERROR**, **INVALID_-ENUM_ERROR**,

  INVALID_VALUE_ERROR, **INVALID_OBJECT_ERROR**, **INVALID_CLASS_ERROR**, **INVALID_OPERATION_ERROR**,

  OPERATION_NOT_SUPPORTED_ERROR, **OUT_OF_MEMORY_ERROR**, **FILE_IO_-ERROR**, **FILE_FORMAT_ERROR**,

  USER_BREAK_ERROR, **MMAP_ERROR**, **IOCTL_ERROR**, **INVALID_DEVICE_ERROR** }

Public Member Functions

- **Exception** (const **ExceptionType** &type=INTERNAL_ERROR, const std::string &description="unknown", const std::string &fileName="unknown", const std::string &functionName="unknown", const **u32** &lineNumber=0)
- const **Exception::ExceptionType** & **getType** () const
- const std::string & **getDescription** () const
- const std::string & **getFileName** () const
- const std::string & **getFunctionName** () const
- const **u32** & **getLineNumber** () const

Static Public Member Functions

- static **Exception** ∗ **getLastException** ()
- static std::string **translateType** (const **Exception::ExceptionType** &type)

Protected Attributes

- **ExceptionType mType**
- std::string **mDescription**
- std::string **mFileName**
- std::string **mFunctionName**
- **u32 mLineNumber**

Static Protected Attributes

- static **Exception** ∗ **mLastException** = 0

7.9.2 Member Enumeration Documentation

7.9.2.1 enum ParCompMark::Exception::ExceptionType

Definitions of error codes.

Enumerator:

INTERNAL_ERROR Unknown internal error (mostly occured by another library).

NULL_POINTER_ERROR Nullpointer error.

INVALID_NAME_ERROR Invalid name error.

INVALID_ENUM_ERROR Invalid enumerated value error.

INVALID_VALUE_ERROR Invalid value error.

INVALID_OBJECT_ERROR Invalid object error.

INVALID_CLASS_ERROR Invalid class error (not proper derived class).

INVALID_OPERATION_ERROR Invalid operation error.

OPERATION_NOT_SUPPORTED_ERROR Operation is not supported on this platform.

OUT_OF_MEMORY_ERROR Out of memory error.

FILE_IO_ERROR File I/O error.

FILE_FORMAT_ERROR Invalid file format error.

USER_BREAK_ERROR The user stopped the application.

MMAP_ERROR Memory mapping error.

IOCTL_ERROR I/O control error.

INVALID_DEVICE_ERROR Not a valid device.

Definition at line 79 of file PCMException.h.

7.9.3 Constructor & Destructor Documentation

7.9.3.1 ParCompMark::Exception::Exception (const ExceptionType & *type* = INTERNAL_ERROR, const std::string & *description* = "unknown", const std::string & *fileName* = "unknown", const std::string & *functionName* = "unknown", const u32 & *lineNumber* = 0) [inline]

Default constructor.

Parameters:

← ***type*** Type of exception.

← ***description*** Textual description of the exception.

← ***fileName*** **Name**(p. 131) of the file where the exception was thrown.

← ***functionName*** **Name**(p. 131) of the function where the exception was thrown.

← ***lineNumber*** Number of the line where the exception was thrown.

Definition at line 322 of file PCMException.h.

7.9.4 Member Function Documentation

7.9.4.1 const std::string & ParCompMark::Exception::getDescription () const [inline]

Remarks:

Getter of mDescription.

Returns value of mDescription.

Returns:

The value of mDescription

Definition at line 356 of file PCMException.h.

Referenced by ParCompMarkTest::TestException::test_constructor_cExceptionType_cstd__string_cstd_-_string_cstd__string_cu32().

7.9.4.2 const std::string & ParCompMark::Exception::getFileName () const [inline]

Remarks:

Getter of mFileName.

Returns value of mFileName.

Returns:

The value of mFileName

Definition at line 363 of file PCMException.h.

Referenced by ParCompMarkTest::TestException::test_constructor_cExceptionType_cstd__string_cstd_-_string_cstd__string_cu32().

7.9.4.3 const std::string & ParCompMark::Exception::getFunctionName () const [inline]

Remarks:
Getter of mFunctionName.

Returns value of mFunctionName.

Returns:
The value of mFunctionName

Definition at line 370 of file PCMException.h.

Referenced by ParCompMarkTest::TestException::test_constructor_cExceptionType_cstd__string_cstd_-_string_cstd__string_cu32().

7.9.4.4 Exception * ParCompMark::Exception::getLastException () [inline, static]

Remarks:
Getter of mLastException.

Returns value of mLastException.

Returns:
The value of mLastException

Definition at line 384 of file PCMException.h.

Referenced by ParCompMark::Application::terminateHandler().

7.9.4.5 const u32 & ParCompMark::Exception::getLineNumber () const [inline]

Remarks:
Getter of mLineNumber.

Returns value of mLineNumber.

Returns:
The value of mLineNumber

Definition at line 377 of file PCMException.h.

Referenced by ParCompMarkTest::TestException::test_constructor_cExceptionType_cstd__string_cstd_-_string_cstd__string_cu32().

7.9.4.6 const Exception::ExceptionType & ParCompMark::Exception::getType () const [inline]

Remarks:
Getter of mType.

Returns value of mType.

Returns:
The value of mType

Definition at line 349 of file PCMException.h.

Referenced by ParCompMarkTest::TestException::test_constructor_cExceptionType_cstd__string_cstd_-_string_cstd__string_cu32().

7.9.4.7 std::string ParCompMark::Exception::translateType (const Exception::ExceptionType & *type*) [static]

Translate type to human readable format.

Parameters:
← ***type*** Enum exception value.

Returns:

Definition at line 44 of file PCMException.cpp.

References FILE_FORMAT_ERROR, FILE_IO_ERROR, INTERNAL_ERROR, INVALID_-CLASS_ERROR, INVALID_ENUM_ERROR, INVALID_NAME_ERROR, INVALID_OBJECT_-ERROR, INVALID_OPERATION_ERROR, INVALID_VALUE_ERROR, NULL_POINTER_ERROR, OPERATION_NOT_SUPPORTED_ERROR, OUT_OF_MEMORY_ERROR, and USER_BREAK_-ERROR.

7.9.5 Member Data Documentation

7.9.5.1 std::string ParCompMark::Exception::mDescription [protected]

Textual description of the exception.

Remarks:
This is own attribute of this class.

Definition at line 162 of file PCMException.h.

7.9.5.2 std::string ParCompMark::Exception::mFileName [protected]

Name(p. 131) of the file where the exception was thrown.

Remarks:
This is own attribute of this class.

Definition at line 169 of file PCMException.h.

7.9.5.3 std::string ParCompMark::Exception::mFunctionName [protected]

Name(p. 131) of the function where the exception was thrown.

Remarks:
This is own attribute of this class.

Definition at line 176 of file PCMException.h.

7.9.5.4 Exception * ParCompMark::Exception::mLastException = 0 [static, protected]

Pointer(p. 170) to the last raised exception.

Remarks:
This attribute references an attribute.

Definition at line 142 of file PCMException.h.

7.9.5.5 u32 ParCompMark::Exception::mLineNumber [protected]

Number of the line where the exception was thrown.

Remarks:
This is own attribute of this class.

Definition at line 183 of file PCMException.h.

7.9.5.6 ExceptionType ParCompMark::Exception::mType [protected]

Type of the exception.

Remarks:
This is own attribute of this class.

Definition at line 155 of file PCMException.h.

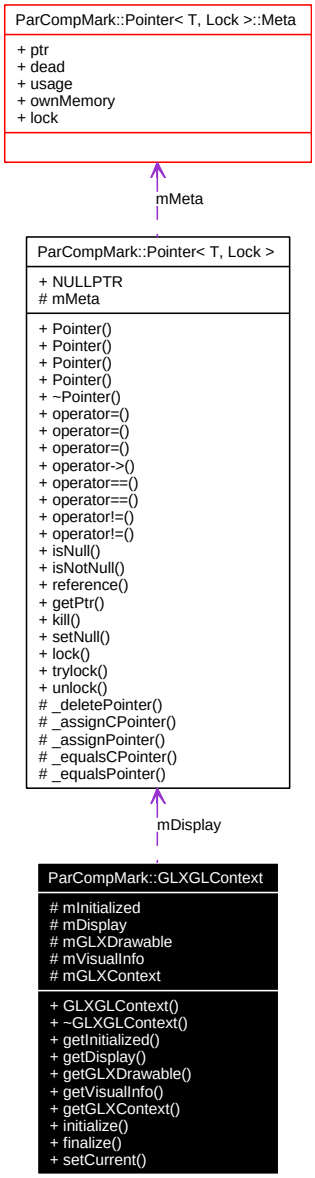
The documentation for this class was generated from the following files:

- **PCMException.h**
- **PCMException.cpp**

7.10 ParCompMark::GLXGLContext Class Reference

`#include <PCMGLXGLContext.h>`

Collaboration diagram for ParCompMark::GLXGLContext:



7.10.1 Detailed Description

Class that encapsulates a GLX context. Original source can be found in Ogre3D sources (`http://ogre3d.org`).

Definition at line 55 of file PCMGLXGLContext.h.

Public Types

- typedef **Pointer< GLXGLContext, DummyLock > Pointer**

Public Member Functions

- **GLXGLContext** (**XDisplay::Pointer** &display,::GLXDrawable glxDrawable,::XVisualInfo *visualInfo)
- virtual ~**GLXGLContext** ()
- const bool & **getInitialized** () const
- const **XDisplay::Pointer** & **getDisplay** () const
- const ::GLXDrawable & **getGLXDrawable** () const
- ::XVisualInfo * **getVisualInfo** () const
- const ::GLXContext & **getGLXContext** () const
- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual void **setCurrent** ()

Protected Attributes

- bool **mInitialized**
- **XDisplay::Pointer mDisplay**
- ::GLXDrawable **mGLXDrawable**
- ::XVisualInfo * **mVisualInfo**
- ::GLXContext **mGLXContext**

7.10.2 Member Typedef Documentation

7.10.2.1 typedef Pointer< GLXGLContext, DummyLock > ParCompMark::GLXGLContext::Pointer

Type for pointer on this class.

Definition at line 73 of file PCMGLXGLContext.h.

7.10.3 Constructor & Destructor Documentation

7.10.3.1 ParCompMark::GLXGLContext::GLXGLContext (XDisplay::Pointer & *display*, ::GLXDrawable *glxDrawable*, ::XVisualInfo * *visualInfo*)

Create GLX context.

Parameters:

→ ***display*** X display

← ***glxDrawable*** GLXDrawable

← ***visualInfo*** Visualinfo for context creation

Definition at line 38 of file PCMGLXGLContext.cpp.

References mGLXContext, mGLXDrawable, mInitialized, and mVisualInfo.

7.10.3.2 ParCompMark::GLXGLContext::~GLXGLContext () [virtual]

The destructor. This class has virtual destructor.

Definition at line 60 of file PCMGLXGLContext.cpp.

References finalize(), and mInitialized.

Here is the call graph for this function:

7.10.4 Member Function Documentation

7.10.4.1 void ParCompMark::GLXGLContext::finalize () [virtual]

Finalize the GL context.

Definition at line 85 of file PCMGLXGLContext.cpp.

References Assert, mDisplay, mGLXContext, mInitialized, and mVisualInfo.

Referenced by ~GLXGLContext().

7.10.4.2 const XDisplay::Pointer & ParCompMark::GLXGLContext::getDisplay () const [inline]

Remarks:
Getter of mDisplay.

Returns value of mDisplay.

Returns:
The value of mDisplay

Definition at line 258 of file PCMGLXGLContext.h.

7.10.4.3 const ::GLXContext & ParCompMark::GLXGLContext::getGLXContext () const [inline]

Remarks:
Getter of mGLXContext.

Returns value of mGLXContext.

Returns:
The value of mGLXContext

Definition at line 279 of file PCMGLXGLContext.h.

7.10.4.4 const ::GLXDrawable & ParCompMark::GLXGLContext::getGLXDrawable () const [inline]

Remarks:
Getter of mGLXDrawable.

Returns value of mGLXDrawable.

Returns:
The value of mGLXDrawable

Definition at line 265 of file PCMGLXGLContext.h.

7.10.4.5 const bool & ParCompMark::GLXGLContext::getInitialized () const [inline]

Remarks:
Getter of mInitialized.

Returns value of mInitialized.

Returns:
The value of mInitialized

Definition at line 251 of file PCMGLXGLContext.h.

7.10.4.6 inline::XVisualInfo * ParCompMark::GLXGLContext::getVisualInfo () const

Remarks:
Getter of mVisualInfo.

Returns value of mVisualInfo.

Returns:
The value of mVisualInfo

Definition at line 272 of file PCMGLXGLContext.h.

7.10.4.7 void ParCompMark::GLXGLContext::initialize () [virtual]

Initialize the GL context.

Definition at line 72 of file PCMGLXGLContext.cpp.

References Assert, mDisplay, mGLXContext, mInitialized, and mVisualInfo.

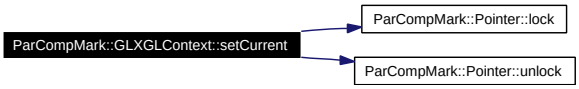
7.10.4.8 void ParCompMark::GLXGLContext::setCurrent () [virtual]

Enable the context. All subsequent rendering commands will go here.

Definition at line 102 of file PCMGLXGLContext.cpp.

References Assert, Except, ParCompMark::Pointer< T, Lock >::lock(), mDisplay, mGLXContext, mGLXDrawable, mInitialized, and ParCompMark::Pointer< T, Lock >::unlock().

Here is the call graph for this function:



7.10.5 Member Data Documentation

7.10.5.1 XDisplay::Pointer ParCompMark::GLXGLContext::mDisplay [protected]

Corresponding X Display.

Remarks:
This is own attribute of this class.

Definition at line 93 of file PCMGLXGLContext.h.

Referenced by finalize(), initialize(), and setCurrent().

7.10.5.2 ::GLXContext ParCompMark::GLXGLContext::mGLXContext [protected]

Wrapped GLX context.

Remarks:
This is own attribute of this class.

Definition at line 114 of file PCMGLXGLContext.h.

Referenced by finalize(), GLXGLContext(), initialize(), and setCurrent().

7.10.5.3 ::GLXDrawable ParCompMark::GLXGLContext::mGLXDrawable [protected]

Corresponding GLXDrawable.

Remarks:
This is own attribute of this class.

Definition at line 100 of file PCMGLXGLContext.h.

Referenced by GLXGLContext(), and setCurrent().

7.10.5.4 bool ParCompMark::GLXGLContext::mInitialized [protected]

The context is initialized.

Remarks:
This is own attribute of this class.

Definition at line 86 of file PCMGLXGLContext.h.

Referenced by finalize(), GLXGLContext(), initialize(), setCurrent(), and ~GLXGLContext().

7.10.5.5 ::XVisualInfo* ParCompMark::GLXGLContext::mVisualInfo [protected]

Visual info for the context.

Remarks:
This is own attribute of this class.

Definition at line 107 of file PCMGLXGLContext.h.

Referenced by finalize(), GLXGLContext(), and initialize().

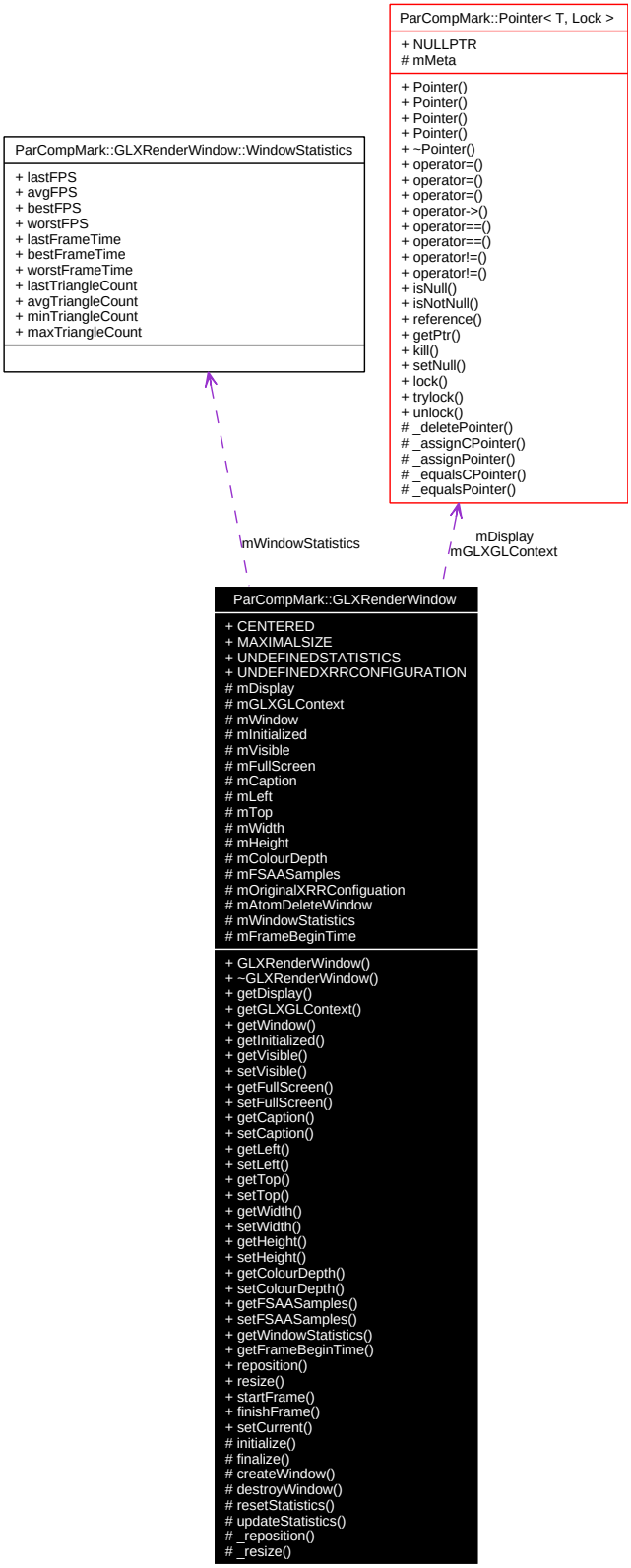
The documentation for this class was generated from the following files:

- **PCMGLXGLContext.h**
- **PCMGLXGLContext.cpp**

7.11 ParCompMark::GLXRenderWindow Class Reference

`#include <PCMGLXRenderWindow.h>`

Collaboration diagram for ParCompMark::GLXRenderWindow:


ParCompMark::Pointer< T, Lock >
+ NULLPTR
# mMeta
+ Pointer()
+ Pointer()
+ Pointer()
+ Pointer()
+ ~Pointer()
+ operator=()
+ operator=()
+ operator=()
+ operator->()
+ operator==()
+ operator==()
+ operator!=()
+ operator!=()
+ isNull()
+ isNotNull()
+ reference()
+ getPtr()
+ kill()
+ setNull()
+ lock()
+ trylock()
+ unlock()
# _deletePointer()
# _assignCPointer()
# _assignPointer()
# _equalsCPointer()
# _equalsPointer()
ParCompMark::GLXRenderWindow::WindowStatistics
+ lastFPS
+ avgFPS
+ bestFPS
+ worstFPS
+ lastFrameTime
+ bestFrameTime
+ worstFrameTime
+ lastTriangleCount
+ avgTriangleCount
+ minTriangleCount
+ maxTriangleCount
mWindowStatistics
mDisplay
mGLXGLContext
ParCompMark::GLXRenderWindow
+ CENTERED
+ MAXIMALSIZE
+ UNDEFINEDSTATISTICS
+ UNDEFINEDXRRCONFIGURATION
# mDisplay
# mGLXGLContext
# mWindow
# mInitialized
# mVisible
# mFullScreen
# mCaption
# mLeft
# mTop
# mWidth
# mHeight
# mColourDepth
# mFSAASamples
# mOriginalXRRConfiguation
# mAtomDeleteWindow
# mWindowStatistics
# mFrameBeginTime
+ GLXRenderWindow()
+ ~GLXRenderWindow()
+ getDisplay()
+ getGLXGLContext()
+ getWindow()
+ getInitialized()
+ getVisible()
+ setVisible()
+ getFullScreen()
+ setFullScreen()
+ getCaption()
+ setCaption()
+ getLeft()
+ setLeft()
+ getTop()
+ setTop()
+ getWidth()
+ setWidth()
+ getHeight()
+ setHeight()
+ getColourDepth()
+ setColourDepth()
+ getFSAASamples()
+ setFSAASamples()
+ getWindowStatistics()
+ getFrameBeginTime()
+ reposition()
+ resize()
+ startFrame()
+ finishFrame()
+ setCurrent()
# initialize()
# finalize()
# createWindow()
# destroyWindow()
# resetStatistics()
# updateStatistics()
# _reposition()
# _resize()

7.11.1 Detailed Description

Manages the target rendering window in GLX environment. Original source can be found in Ogre3D sources (`http://ogre3d.org`).

Definition at line 70 of file PCMGLXRenderWindow.h.

Public Types

- typedef **Pointer**< **GLXRenderWindow**, **DummyLock** > **Pointer**

Public Member Functions

- **GLXRenderWindow** (**XDisplay::Pointer** &display, const std::string caption="PCM Framework", const bool &fullScreen=true, const **u32** &colourDepth=0, const **s32** &left=**GLXRenderWindow::CENTERED**, const **s32** &top=**GLXRenderWindow::CENTERED**, const **u32** &width=**GLXRenderWindow::MAXIMALSIZE**, const **u32** &height=**GLXRenderWindow::MAXIMALSIZE**, const **u32** &fsaaSamples=0)
- virtual ~**GLXRenderWindow** ()
- **XDisplay::Pointer** & **getDisplay** ()
- **GLXGLContext::Pointer** & **getGLXGLContext** ()
- const ::Window & **getWindow** () const
- const bool & **getInitialized** () const
- const bool & **getVisible** () const
- void **setVisible** (const bool &visible)
- const bool & **getFullScreen** () const
- void **setFullScreen** (const bool &fullscreen)
- const std::string & **getCaption** () const
- void **setCaption** (const std::string &caption)
- const **s32** & **getLeft** () const
- void **setLeft** (const **s32** &left)
- const **s32** & **getTop** () const
- void **setTop** (const **s32** &top)
- const **u32** & **getWidth** () const
- void **setWidth** (const **u32** &width)
- const **u32** & **getHeight** () const
- void **setHeight** (const **u32** &height)
- const **u32** & **getColourDepth** () const
- void **setColourDepth** (const **u32** &colourdepth)
- const **u32** & **getFSAASamples** () const
- void **setFSAASamples** (const **u32** &fsaasamples)
- const **GLXRenderWindow::WindowStatistics** & **getWindowStatistics** () const
- const **Real** & **getFrameBeginTime** () const
- virtual void **reposition** (const **s32** &left, const **s32** &top)
- virtual void **resize** (const **u32** &width, const **u32** &height)
- virtual void **startFrame** ()
- virtual void **finishFrame** ()
- virtual void **setCurrent** ()

Static Public Attributes

- static const **s32 CENTERED** = -1
- static const **u32 MAXIMALSIZE** = 0
- static const **Real UNDEFINEDSTATISTICS** = -1.0
- static const **s32 UNDEFINEDXRRCONFIGURATION** = -1

Protected Member Functions

- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual void **createWindow** ()
- virtual void **destroyWindow** ()
- virtual void **resetStatistics** ()
- virtual void **updateStatistics** ()
- virtual void **_reposition** ()
- virtual void **_resize** ()

Protected Attributes

- **XDisplay::Pointer mDisplay**
- **GLXGLContext::Pointer mGLXGLContext**
- ::Window **mWindow**
- bool **mInitialized**
- bool **mVisible**
- bool **mFullScreen**
- std::string **mCaption**
- **s32 mLeft**
- **s32 mTop**
- **u32 mWidth**
- **u32 mHeight**
- **u32 mColourDepth**
- **u32 mFSAASamples**
- **s32 mOriginalXRRConfiguation**
- ::Atom **mAtomDeleteWindow**
- **WindowStatistics mWindowStatistics**
- **Real mFrameBeginTime**

Classes

- struct **WindowStatistics**

7.11.2 Member Typedef Documentation

7.11.2.1 typedef Pointer< GLXRenderWindow, DummyLock > ParCompMark::GLXRenderWindow::Pointer

Type for pointer on this class.

Definition at line 88 of file PCMGLXRenderWindow.h.

7.11.3 Constructor & Destructor Documentation

7.11.3.1 ParCompMark::GLXRenderWindow::GLXRenderWindow (XDisplay::Pointer & *display*, const std::string *caption* = "PCM Framework", const bool & *fullScreen* = true, const u32 & *colourDepth* = 0, const s32 & *left* = GLXRenderWindow::CENTERED, const s32 & *top* = GLXRenderWindow::CENTERED, const u32 & *width* = GLXRenderWindow::MAXIMALSIZE, const u32 & *height* = GLXRenderWindow::MAXIMALSIZE, const u32 & *fsaaSamples* = 0)

Create GLX render window. Normally called by XDisplay::createRenderWindow.

Parameters:

→ ***display*** X display

← ***caption*** Window caption

← ***fullScreen*** The window appears in full screen mode

← ***colourDepth*** Colour depth of the window

← ***left*** Horizontal position

← ***top*** Vertical position

← ***width*** Horizontal size

← ***height*** Vertical size

← ***fsaaSamples*** Number of fullscreen antialasing samples (Do not use FSAA samples other than 0 now with nVidia cards!)

Definition at line 55 of file PCMGLXRenderWindow.cpp.

References mCaption, mColourDepth, mFSAASamples, mFullScreen, mHeight, mInitialized, mLeft, mOriginalXRRConfiguation, mTop, mVisible, mWidth, mWindow, resetStatistics(), and UNDEFINEDXRRCONFIGURATION.

Here is the call graph for this function:

7.11.3.2 ParCompMark::GLXRenderWindow::~GLXRenderWindow () [virtual]

The destructor. This class has virtual destructor.

Definition at line 100 of file PCMGLXRenderWindow.cpp.

References finalize(), and mInitialized.

Here is the call graph for this function:

7.11.4 Member Function Documentation

7.11.4.1 void ParCompMark::GLXRenderWindow::_reposition () [inline, protected, virtual]

Efficiently set window position (for internal use).

Definition at line 922 of file PCMGLXRenderWindow.h.

7.11.4.2 void ParCompMark::GLXRenderWindow::_resize () [inline, protected, virtual]

Efficiently set window sizes (for internal use).

Definition at line 931 of file PCMGLXRenderWindow.h.

7.11.4.3 void ParCompMark::GLXRenderWindow::createWindow () [protected, virtual]

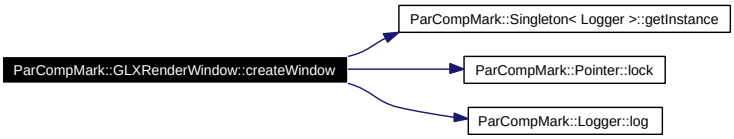
Create GLX Window. Protected method for internal use.

Definition at line 143 of file PCMGLXRenderWindow.cpp.

References CENTERED, ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Pointer< T, Lock >::lock(), ParCompMark::Logger::log(), MAXIMALSIZE, mCaption, mDisplay, mFSAASamples, mFullScreen, mHeight, mLeft, mOriginalXRRConfiguation, mTop, mWidth, ParCompMark::Logger::NOTICE, and ParCompMark::Logger::WARNING.

Referenced by initialize().

Here is the call graph for this function:



7.11.4.4 void ParCompMark::GLXRenderWindow::destroyWindow () [protected, virtual]

Destroy GLX Window. Protected method for internal use.

Definition at line 348 of file PCMGLXRenderWindow.cpp.

References Assert, mDisplay, and mWindow.

Referenced by finalize().

7.11.4.5 void ParCompMark::GLXRenderWindow::finalize () [protected, virtual]

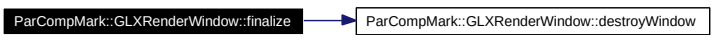
Finalize the GLX RenderWindow. Protected method for internal use.

Definition at line 131 of file PCMGLXRenderWindow.cpp.

References Assert, destroyWindow(), and mInitialized.

Referenced by ~GLXRenderWindow().

Here is the call graph for this function:



7.11.4.6 void ParCompMark::GLXRenderWindow::finishFrame () [inline, virtual]

Finish current frame.

Definition at line 909 of file PCMGLXRenderWindow.h.

7.11.4.7 const std::string & ParCompMark::GLXRenderWindow::getCaption () const [inline]

Remarks:
Getter of mCaption.

Returns value of mCaption.

Returns:
The value of mCaption

Definition at line 756 of file PCMGLXRenderWindow.h.

7.11.4.8 const u32 & ParCompMark::GLXRenderWindow::getColourDepth () const [inline]

Remarks:
Getter of mColourDepth.

Returns value of mColourDepth.

Returns:
The value of mColourDepth

Definition at line 833 of file PCMGLXRenderWindow.h.

7.11.4.9 XDisplay::Pointer & ParCompMark::GLXRenderWindow::getDisplay () [inline]

Remarks:
Getter of mDisplay.

Returns value of mDisplay.

Returns:
The value of mDisplay

Definition at line 690 of file PCMGLXRenderWindow.h.

7.11.4.10 const Real & ParCompMark::GLXRenderWindow::getFrameBeginTime () const [inline]

Remarks:
Getter of mFrameBeginTime.

Returns value of mFrameBeginTime.

Returns:
The value of mFrameBeginTime

Definition at line 868 of file PCMGLXRenderWindow.h.

7.11.4.11 const u32 & ParCompMark::GLXRenderWindow::getFSAASamples () const [inline]

Remarks:
Getter of mFSAASamples.

Returns value of mFSAASamples.

Returns:
The value of mFSAASamples

Definition at line 847 of file PCMGLXRenderWindow.h.

7.11.4.12 const bool & ParCompMark::GLXRenderWindow::getFullScreen () const [inline]

Remarks:
Getter of mFullScreen.

Returns value of mFullScreen.

Returns:
The value of mFullScreen

Definition at line 742 of file PCMGLXRenderWindow.h.

7.11.4.13 GLXGLContext::Pointer & ParCompMark::GLXRenderWindow::getGLXGLContext () [inline]

Remarks:
Getter of mGLXGLContext.

Returns value of mGLXGLContext.

Returns:
The value of mGLXGLContext

Definition at line 697 of file PCMGLXRenderWindow.h.

7.11.4.14 const u32 & ParCompMark::GLXRenderWindow::getHeight () const [inline]

Remarks:
Getter of mHeight.

Returns value of mHeight.

Returns:
The value of mHeight

Definition at line 819 of file PCMGLXRenderWindow.h.

7.11.4.15 const bool & ParCompMark::GLXRenderWindow::getInitialized () const [inline]

Remarks:
Getter of mInitialized.

Returns value of mInitialized.

Returns:
The value of mInitialized

Definition at line 711 of file PCMGLXRenderWindow.h.

7.11.4.16 const s32 & ParCompMark::GLXRenderWindow::getLeft () const [inline]

Remarks:
Getter of mLeft.

Returns value of mLeft.

Returns:
The value of mLeft

Definition at line 777 of file PCMGLXRenderWindow.h.

7.11.4.17 const s32 & ParCompMark::GLXRenderWindow::getTop () const [inline]

Remarks:
Getter of mTop.

Returns value of mTop.

Returns:
The value of mTop

Definition at line 791 of file PCMGLXRenderWindow.h.

7.11.4.18 const bool & ParCompMark::GLXRenderWindow::getVisible () const [inline]

Remarks:
Getter of mVisible.

Returns value of mVisible.

Returns:
The value of mVisible

Definition at line 718 of file PCMGLXRenderWindow.h.

7.11.4.19 const u32 & ParCompMark::GLXRenderWindow::getWidth () const [inline]

Remarks:
Getter of mWidth.

Returns value of mWidth.

Returns:
The value of mWidth

Definition at line 805 of file PCMGLXRenderWindow.h.

7.11.4.20 const ::Window & ParCompMark::GLXRenderWindow::getWindow () const [inline]

Remarks:
Getter of mWindow.

Returns value of mWindow.

Returns:
The value of mWindow

Definition at line 704 of file PCMGLXRenderWindow.h.

7.11.4.21 const GLXRenderWindow::WindowStatistics & ParCompMark::GLXRenderWindow::getWindowStatistics () const [inline]

Remarks:
Getter of mWindowStatistics.

Returns value of mWindowStatistics.

Returns:
The value of mWindowStatistics

Definition at line 861 of file PCMGLXRenderWindow.h.

7.11.4.22 void ParCompMark::GLXRenderWindow::initialize () [protected, virtual]

Initialize the GLX RenderWindow. Protected method for internal use.

Definition at line 119 of file PCMGLXRenderWindow.cpp.

References Assert, createWindow(), and mInitialized.

Here is the call graph for this function:

7.11.4.23 void ParCompMark::GLXRenderWindow::reposition (const s32 & *left*, const s32 & *top*) [inline, virtual]

Set new window position.

Parameters:

← ***left*** Horizontal position

← ***top*** Vertical position

Definition at line 879 of file PCMGLXRenderWindow.h.

7.11.4.24 void ParCompMark::GLXRenderWindow::resetStatistics () [protected, virtual]

Reset statistics. Protected method for internal use.

Definition at line 357 of file PCMGLXRenderWindow.cpp.

References ParCompMark::GLXRenderWindow::WindowStatistics::avgFPS, ParCompMark::GLXRenderWindow::WindowStatistics::avgTriangleCount, ParCompMark::GLXRenderWindow::WindowStatistics::bestFPS, ParCompMark::GLXRenderWindow::WindowStatistics::bestFrameTime, ParCompMark::Timer::getSystemTime(), ParCompMark::GLXRenderWindow::WindowStatistics::lastFPS, ParCompMark::GLXRenderWindow::WindowStatistics::lastFrameTime, ParCompMark::GLXRenderWindow::WindowStatistics::lastTriangleCount, ParCompMark::GLXRenderWindow::WindowStatistics::maxTriangleCount, mFrameBeginTime, ParCompMark::GLXRenderWindow::WindowStatistics::minTriangleCount, mWindowStatistics, UNDEFINEDSTATISTICS, ParCompMark::GLXRenderWindow::WindowStatistics::worstFPS, and ParCompMark::GLXRenderWindow::WindowStatistics::worstFrameTime.

Referenced by GLXRenderWindow().

Here is the call graph for this function:

7.11.4.25 void ParCompMark::GLXRenderWindow::resize (const u32 & *width*, const u32 & *height*) [inline, virtual]

Set new window sizes.

Parameters:

← ***width*** Horizontal size

← ***height*** Vertical size

Definition at line 890 of file PCMGLXRenderWindow.h.

7.11.4.26 void ParCompMark::GLXRenderWindow::setCaption (const std::string & *caption*) [inline]

Remarks:

Setter of mCaption.

Sets value of mCaption.

Parameters:
← ***caption*** The value of mCaption

Definition at line 763 of file PCMGLXRenderWindow.h.

Referenced by updateStatistics().

7.11.4.27 void ParCompMark::GLXRenderWindow::setColourDepth (const u32 & *colourdepth*) [inline]

Remarks:
Setter of mColourDepth.

Sets value of mColourDepth.

Parameters:
← ***colourdepth*** The value of mColourDepth

Definition at line 840 of file PCMGLXRenderWindow.h.

7.11.4.28 void ParCompMark::GLXRenderWindow::setCurrent () [virtual]

Enable the context of the window. All subsequent rendering commands will go on this window. The sideeffect of calling this method is reseting the window statistics.

Definition at line 112 of file PCMGLXRenderWindow.cpp.

References mGLXGLContext.

7.11.4.29 void ParCompMark::GLXRenderWindow::setFSAASamples (const u32 & *fsaasamples*) [inline]

Remarks:
Setter of mFSAASamples.

Sets value of mFSAASamples.

Parameters:
← ***fsaasamples*** The value of mFSAASamples

Definition at line 854 of file PCMGLXRenderWindow.h.

7.11.4.30 void ParCompMark::GLXRenderWindow::setFullScreen (const bool & *fullscreen*) [inline]

Remarks:
Setter of mFullScreen.

Sets value of mFullScreen.

Parameters:
← ***fullscreen*** The value of mFullScreen

Definition at line 749 of file PCMGLXRenderWindow.h.

7.11.4.31 void ParCompMark::GLXRenderWindow::setHeight (const u32 & *height*) [inline]

Remarks:
Setter of mHeight.

Sets value of mHeight.

Parameters:
← ***height*** The value of mHeight

Definition at line 826 of file PCMGLXRenderWindow.h.

7.11.4.32 void ParCompMark::GLXRenderWindow::setLeft (const s32 & *left*) [inline]

Remarks:
Setter of mLeft.

Sets value of mLeft.

Parameters:
← ***left*** The value of mLeft

Definition at line 784 of file PCMGLXRenderWindow.h.

7.11.4.33 void ParCompMark::GLXRenderWindow::setTop (const s32 & *top*) [inline]

Remarks:
Setter of mTop.

Sets value of mTop.

Parameters:
← ***top*** The value of mTop

Definition at line 798 of file PCMGLXRenderWindow.h.

7.11.4.34 void ParCompMark::GLXRenderWindow::setVisible (const bool & *visible*) [inline]

Remarks:
Setter of mVisible.

Sets value of mVisible.

Parameters:
← ***visible*** The value of mVisible

Definition at line 725 of file PCMGLXRenderWindow.h.

7.11.4.35 void ParCompMark::GLXRenderWindow::setWidth (const u32 & *width*) [inline]

Remarks:
Setter of mWidth.

Sets value of mWidth.

Parameters:
← ***width*** The value of mWidth

Definition at line 812 of file PCMGLXRenderWindow.h.

7.11.4.36 void ParCompMark::GLXRenderWindow::startFrame () [inline, virtual]

Start a frame.

Definition at line 901 of file PCMGLXRenderWindow.h.

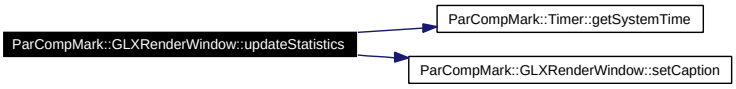
7.11.4.37 void ParCompMark::GLXRenderWindow::updateStatistics () [protected, virtual]

Update statistics of the window. Protected method for internal use.

Definition at line 381 of file PCMGLXRenderWindow.cpp.

References ParCompMark::GLXRenderWindow::WindowStatistics::bestFrameTime, ParCompMark::Timer::EPSILONDELAY, ParCompMark::Timer::getSystemTime(), ParCompMark::GLXRenderWindow::WindowStatistics::lastFPS, ParCompMark::GLXRenderWindow::WindowStatistics::lastFrameTime, mCaption, mFrameBeginTime, mWindowStatistics, setCaption(), UNDEFINEDSTATISTICS, and ParCompMark::GLXRenderWindow::WindowStatistics::worstFrameTime.

Here is the call graph for this function:



7.11.5 Member Data Documentation

7.11.5.1 const s32 ParCompMark::GLXRenderWindow::CENTERED = -1 [static]

Constant for centered position.

Remarks:
This is own attribute of this class.

Definition at line 128 of file PCMGLXRenderWindow.h.

Referenced by createWindow().

7.11.5.2 ::Atom ParCompMark::GLXRenderWindow::mAtomDeleteWindow [protected]

Atom to recognize window close events.

Remarks:
This is own attribute of this class.

Definition at line 260 of file PCMGLXRenderWindow.h.

7.11.5.3 const u32 ParCompMark::GLXRenderWindow::MAXIMALSIZE = 0 [static]

Constant for maximal size.

Remarks:
This is own attribute of this class.

Definition at line 135 of file PCMGLXRenderWindow.h.

Referenced by createWindow().

7.11.5.4 std::string ParCompMark::GLXRenderWindow::mCaption [protected]

Window caption.

Remarks:
This is own attribute of this class.

Definition at line 204 of file PCMGLXRenderWindow.h.

Referenced by createWindow(), GLXRenderWindow(), and updateStatistics().

7.11.5.5 u32 ParCompMark::GLXRenderWindow::mColourDepth [protected]

Colour depth of the window.

Remarks:
This is own attribute of this class.

Definition at line 239 of file PCMGLXRenderWindow.h.

Referenced by GLXRenderWindow().

7.11.5.6 XDisplay::Pointer ParCompMark::GLXRenderWindow::mDisplay [protected]

Corresponding X Display.

Remarks:
This is own attribute of this class.

Definition at line 162 of file PCMGLXRenderWindow.h.

Referenced by createWindow(), and destroyWindow().

7.11.5.7 Real ParCompMark::GLXRenderWindow::mFrameBeginTime [protected]

Time of stating a new frame.

Remarks:
This is own attribute of this class.

Definition at line 274 of file PCMGLXRenderWindow.h.

Referenced by resetStatistics(), and updateStatistics().

7.11.5.8 u32 ParCompMark::GLXRenderWindow::mFSAASamples [protected]

Number of fullscreen antialasing samples.

Remarks:
This is own attribute of this class.

Definition at line 246 of file PCMGLXRenderWindow.h.

Referenced by createWindow(), and GLXRenderWindow().

7.11.5.9 bool ParCompMark::GLXRenderWindow::mFullScreen [protected]

The window appears in full screen mode.

Remarks:
This is own attribute of this class.

Definition at line 197 of file PCMGLXRenderWindow.h.

Referenced by createWindow(), and GLXRenderWindow().

7.11.5.10 GLXGLContext::Pointer ParCompMark::GLXRenderWindow::mGLXGLContext [protected]

GLX context of the render window.

Remarks:
This is own attribute of this class.

Definition at line 169 of file PCMGLXRenderWindow.h.

Referenced by setCurrent().

7.11.5.11 u32 ParCompMark::GLXRenderWindow::mHeight [protected]

Vertical size of the window.

Remarks:
This is own attribute of this class.

Definition at line 232 of file PCMGLXRenderWindow.h.

Referenced by createWindow(), and GLXRenderWindow().

7.11.5.12 bool ParCompMark::GLXRenderWindow::mInitialized [protected]

The window is initialized.

Remarks:
This is own attribute of this class.

Definition at line 183 of file PCMGLXRenderWindow.h.

Referenced by finalize(), GLXRenderWindow(), initialize(), and ~GLXRenderWindow().

7.11.5.13 s32 ParCompMark::GLXRenderWindow::mLeft [protected]

Horizontal position of the window.

Remarks:
This is own attribute of this class.

Definition at line 211 of file PCMGLXRenderWindow.h.

Referenced by createWindow(), and GLXRenderWindow().

7.11.5.14 s32 ParCompMark::GLXRenderWindow::mOriginalXRRConfiguation [protected]

For storing original XRR configuration mode.

Remarks:
This is own attribute of this class.

Definition at line 253 of file PCMGLXRenderWindow.h.

Referenced by createWindow(), and GLXRenderWindow().

7.11.5.15 s32 ParCompMark::GLXRenderWindow::mTop [protected]

Vertical position of the window.

Remarks:
This is own attribute of this class.

Definition at line 218 of file PCMGLXRenderWindow.h.

Referenced by createWindow(), and GLXRenderWindow().

7.11.5.16 bool ParCompMark::GLXRenderWindow::mVisible [protected]

The window is visible.

Remarks:
This is own attribute of this class.

Definition at line 190 of file PCMGLXRenderWindow.h.

Referenced by GLXRenderWindow().

7.11.5.17 u32 ParCompMark::GLXRenderWindow::mWidth [protected]

Horizontal size of the window.

Remarks:
This is own attribute of this class.

Definition at line 225 of file PCMGLXRenderWindow.h.

Referenced by createWindow(), and GLXRenderWindow().

7.11.5.18 ::Window ParCompMark::GLXRenderWindow::mWindow [protected]

Wrapped GLX Window.

Remarks:
This is own attribute of this class.

Definition at line 176 of file PCMGLXRenderWindow.h.

Referenced by destroyWindow(), and GLXRenderWindow().

7.11.5.19 WindowStatistics ParCompMark::GLXRenderWindow::mWindowStatistics [protected]

Current window statistics.

Remarks:
This is own attribute of this class.

Definition at line 267 of file PCMGLXRenderWindow.h.

Referenced by resetStatistics(), and updateStatistics().

7.11.5.20 const Real ParCompMark::GLXRenderWindow::UNDEFINEDSTATISTICS = -1.0 [static]

Undefined statistics value.

Remarks:
This is own attribute of this class.

Definition at line 142 of file PCMGLXRenderWindow.h.

Referenced by resetStatistics(), and updateStatistics().

7.11.5.21 const s32 ParCompMark::GLXRenderWindow::UNDEFINEDXRRCONFIGURATION = -1 [static]

Constant for undefined XRR configuration.

Remarks:
This is own attribute of this class.

Definition at line 149 of file PCMGLXRenderWindow.h.

Referenced by GLXRenderWindow().

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

- **PCMGLXRenderWindow.h**
- **PCMGLXRenderWindow.cpp**

7.12 ParCompMark::GLXRenderWindow::WindowStatistics Struct Reference

`#include <PCMGLXRenderWindow.h>`

7.12.1 Detailed Description

Struct for window statistics (FPS values, frame times).

Definition at line 91 of file PCMGLXRenderWindow.h.

Public Attributes

- **Real lastFPS**
- **Real avgFPS**
- **Real bestFPS**
- **Real worstFPS**
- **Real lastFrameTime**
- **Real bestFrameTime**
- **Real worstFrameTime**
- **Real lastTriangleCount**
- **Real avgTriangleCount**
- **Real minTriangleCount**
- **Real maxTriangleCount**

7.12.2 Member Data Documentation

7.12.2.1 Real ParCompMark::GLXRenderWindow::WindowStatistics::avgFPS

Average frame rate

Definition at line 97 of file PCMGLXRenderWindow.h.

Referenced by ParCompMark::GLXRenderWindow::resetStatistics().

7.12.2.2 Real ParCompMark::GLXRenderWindow::WindowStatistics::avgTriangleCount

Average triangle count

Definition at line 111 of file PCMGLXRenderWindow.h.

Referenced by ParCompMark::GLXRenderWindow::resetStatistics().

7.12.2.3 Real ParCompMark::GLXRenderWindow::WindowStatistics::bestFPS

Frame rate for the frame with the best frame time

Definition at line 99 of file PCMGLXRenderWindow.h.

Referenced by ParCompMark::GLXRenderWindow::resetStatistics().

7.12.2.4 Real ParCompMark::GLXRenderWindow::WindowStatistics::bestFrameTime

Best frame time

Definition at line 105 of file PCMGLXRenderWindow.h.

Referenced by ParCompMark::GLXRenderWindow::resetStatistics(), and ParCompMark::GLXRenderWindow::updateStatistics().

7.12.2.5 Real ParCompMark::GLXRenderWindow::WindowStatistics::lastFPS

Frame rate for the last frame

Definition at line 95 of file PCMGLXRenderWindow.h.

Referenced by ParCompMark::GLXRenderWindow::resetStatistics(), and ParCompMark::GLXRenderWindow::updateStatistics().

7.12.2.6 Real ParCompMark::GLXRenderWindow::WindowStatistics::lastFrameTime

Last frame time

Definition at line 103 of file PCMGLXRenderWindow.h.

Referenced by ParCompMark::GLXRenderWindow::resetStatistics(), and ParCompMark::GLXRenderWindow::updateStatistics().

7.12.2.7 Real ParCompMark::GLXRenderWindow::WindowStatistics::lastTriangleCount

Triangle count for the last frame

Definition at line 109 of file PCMGLXRenderWindow.h.

Referenced by ParCompMark::GLXRenderWindow::resetStatistics().

7.12.2.8 Real ParCompMark::GLXRenderWindow::WindowStatistics::maxTriangleCount

Maximal triangle count

Definition at line 114 of file PCMGLXRenderWindow.h.

Referenced by ParCompMark::GLXRenderWindow::resetStatistics().

7.12.2.9 Real ParCompMark::GLXRenderWindow::WindowStatistics::minTriangleCount

Minimal triangle count

Definition at line 113 of file PCMGLXRenderWindow.h.

Referenced by ParCompMark::GLXRenderWindow::resetStatistics().

7.12.2.10 Real ParCompMark::GLXRenderWindow::WindowStatistics::worstFPS

Frame rate for the frame with the worst frame time

Definition at line 101 of file PCMGLXRenderWindow.h.

Referenced by ParCompMark::GLXRenderWindow::resetStatistics().

7.12.2.11 Real ParCompMark::GLXRenderWindow::WindowStatistics::worstFrameTime

Worst frame time

Definition at line 107 of file PCMGLXRenderWindow.h.

Referenced by ParCompMark::GLXRenderWindow::resetStatistics(), and ParCompMark::GLXRenderWindow::updateStatistics().

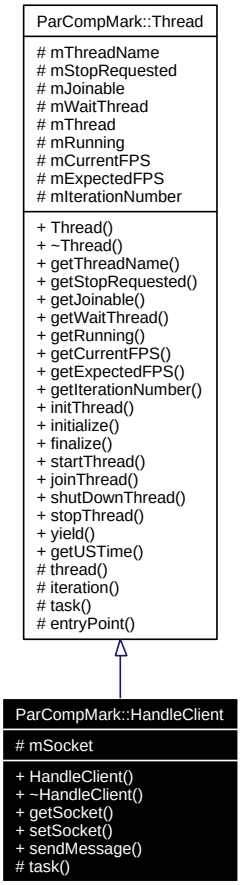
The documentation for this struct was generated from the following file:

- **PCMGLXRenderWindow.h**

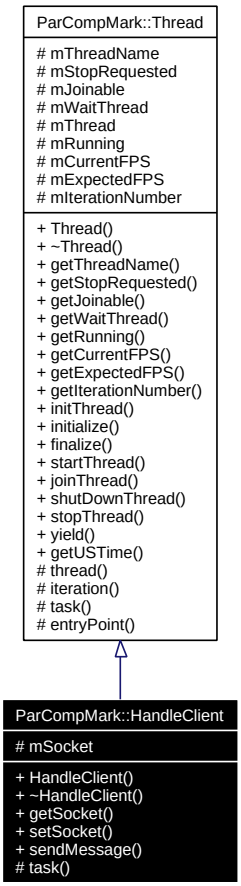
7.13 ParCompMark::HandleClient Class Reference

`#include <PCMHandleClient.h>`

Inheritance diagram for ParCompMark::HandleClient:



Collaboration diagram for ParCompMark::HandleClient:



7.13.1 Detailed Description

Class for handle client messages.

Definition at line 57 of file PCMHandleClient.h.

Public Types

- typedef **Pointer**< **HandleClient**, **DummyLock** > **Pointer**

Public Member Functions

- **HandleClient** ()
- virtual ~**HandleClient** ()
- const int & **getSocket** () const
- void **setSocket** (const int &socket)
- virtual void **sendMessage** (const std::string &mType, const std::string &message)

Protected Member Functions

- virtual void **task** ()

Protected Attributes

- int **mSocket**

7.13.2 Member Typedef Documentation

7.13.2.1 typedef Pointer< HandleClient, DummyLock > ParCompMark::HandleClient::Pointer

Type for pointer or this class.

Definition at line 75 of file PCMHandleClient.h.

7.13.3 Constructor & Destructor Documentation

7.13.3.1 ParCompMark::HandleClient::HandleClient ()

Create **HandleClient**(p. 107) class.

Definition at line 38 of file PCMHandleClient.cpp.

References mSocket.

7.13.3.2 ParCompMark::HandleClient::~HandleClient () [virtual]

The destructor. This class has virtual destructor.

Definition at line 52 of file PCMHandleClient.cpp.

References mSocket.

7.13.4 Member Function Documentation

7.13.4.1 const int & ParCompMark::HandleClient::getSocket () const [inline]

Remarks:
Getter of mSocket.

Returns value of mSocket.

Returns:
The value of mSocket

Definition at line 188 of file PCMHandleClient.h.

7.13.4.2 void ParCompMark::HandleClient::sendMessage (const std::string & *mType*, const std::string & *message*) [virtual]

Send a TCP/IP message.

Parameters:
← ***mType*** Type of the message.
← ***message*** The message.

Definition at line 69 of file PCMHandleClient.cpp.

References Assert, Except, and mSocket.

7.13.4.3 void ParCompMark::HandleClient::setSocket (const int & *socket*) [inline]

Remarks:
Setter of mSocket.

Sets value of mSocket.

Parameters:
← ***socket*** The value of mSocket

Definition at line 195 of file PCMHandleClient.h.

Referenced by ParCompMark::Network::acceptClientConnection().

7.13.4.4 void ParCompMark::HandleClient::task () [protected, virtual]

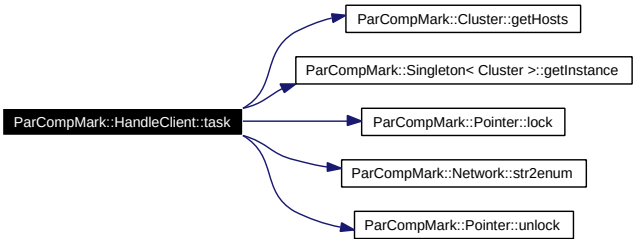
Handle a client.

Reimplemented from **ParCompMark::Thread** (p. 271).

Definition at line 91 of file PCMHandleClient.cpp.

References Assert, ParCompMark::Cluster::getHosts(), ParCompMark::Singleton< Cluster >::getInstance(), ParCompMark::Network::INITOK, ParCompMark::Pointer< T, Lock >::lock(), mSocket, ParCompMark::Network::str2enum(), ParCompMark::Network::TIME, and ParCompMark::Pointer< T, Lock >::unlock().

Here is the call graph for this function:



7.13.5 Member Data Documentation

7.13.5.1 int ParCompMark::HandleClient::mSocket [protected]

The communication socket.

Remarks:
This is own attribute of this class.

Definition at line 88 of file PCMHandleClient.h.

Referenced by HandleClient(), sendMessage(), task(), and ~HandleClient().

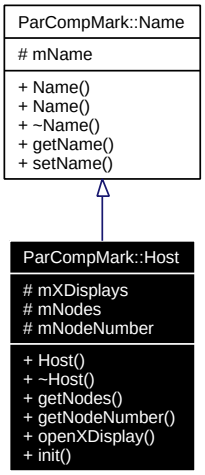
The documentation for this class was generated from the following files:

- **PCMHandleClient.h**
- **PCMHandleClient.cpp**

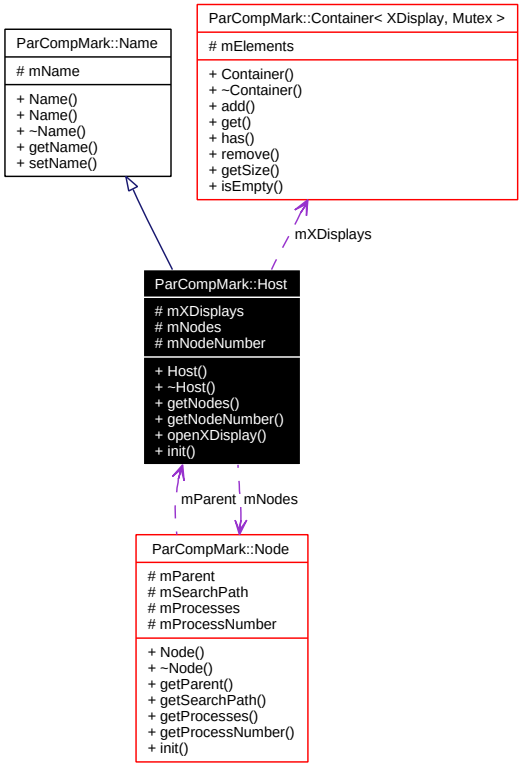
7.14 ParCompMark::Host Class Reference

`#include <PCMHost.h>`

Inheritance diagram for ParCompMark::Host:



Collaboration diagram for ParCompMark::Host:



7.14.1 Detailed Description

Class for describe a hosts.

Definition at line 59 of file PCMHost.h.

Public Member Functions

- **Host** (const std::string &name)
- virtual ~**Host** ()
- **Node** ** **getNodes** () const
- const int & **getNodeNumber** () const
- virtual **XDisplay::Pointer openXDisplay** (const std::string &displayName="")
- virtual void **init** (const std::string &conf)

Protected Attributes

- **Container**< **XDisplay**, **Mutex** >::**Pointer mXDisplays**
- **Node** ** **mNodes**
- int **mNodeNumber**

7.14.2 Constructor & Destructor Documentation

7.14.2.1 ParCompMark::Host::Host (const std::string & *name*)

Creates a host with a specified name.

Parameters:
← ***name*** **Name**(p. 131) of the host.

Definition at line 38 of file PCMHost.cpp.

References Except, mNodeNumber, mNodes, and mXDisplays.

7.14.2.2 ParCompMark::Host::~Host () `[virtual]`

The destructor. This class has virtual destructor.

Definition at line 62 of file PCMHost.cpp.

References mNodeNumber, and mNodes.

7.14.3 Member Function Documentation

7.14.3.1 const int & ParCompMark::Host::getNodeNumber () const `[inline]`

Remarks:
Getter of mNodeNumber.

Returns value of mNodeNumber.

Returns:
The value of mNodeNumber

Definition at line 205 of file PCMHost.h.

7.14.3.2 Node ** ParCompMark::Host::getNodes () const [inline]

Remarks:
Getter of mNodes.

Returns value of mNodes.

Returns:
The value of mNodes

Definition at line 198 of file PCMHost.h.

7.14.3.3 void ParCompMark::Host::init (const std::string & *conf*) [virtual]

Create and call init functions of the nodes on the hosts.

Parameters:
← ***conf*** Config string.

Definition at line 92 of file PCMHost.cpp.

References ParCompMark::Name::mName, mNodeNumber, and mNodes.

Referenced by ParCompMark::Application::NetworkTest().

7.14.3.4 XDisplay::Pointer ParCompMark::Host::openXDisplay (const std::string & *displayName* = "") [virtual]

Open an X display. An X display can be opened several times, and does not have to be closed. The host will close it when it is destructed. When no parameter is set, the default display will be opened.

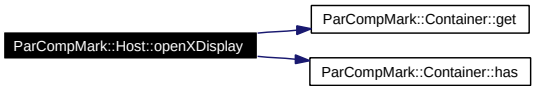
Parameters:
← ***displayName*** X display name

Returns:
Opened X Display

Definition at line 134 of file PCMHost.cpp.

References ParCompMark::Container< ElementType, LockType >::get(), ParCompMark::Container< ElementType, LockType >::has(), and mXDisplays.

Here is the call graph for this function:



7.14.4 Member Data Documentation

7.14.4.1 int ParCompMark::Host::mNodeNumber [protected]

Number of processes.

Remarks:
This is own attribute of this class.

Definition at line 94 of file PCMHost.h.

Referenced by Host(), init(), and ~Host().

7.14.4.2 Node** ParCompMark::Host::mNodes [protected]

Processes of the node.

Remarks:
This is own attribute of this class.

Definition at line 87 of file PCMHost.h.

Referenced by Host(), init(), and ~Host().

7.14.4.3 Container< XDisplay, Mutex >::Pointer ParCompMark::Host::mXDisplays [protected]

X Displays on this host.

Remarks:
This is own attribute of this class.

Definition at line 80 of file PCMHost.h.

Referenced by Host(), and openXDisplay().

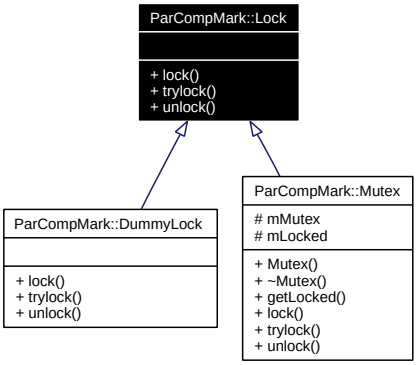
The documentation for this class was generated from the following files:

- **PCMHost.h**
- **PCMHost.cpp**

7.15 ParCompMark::Lock Class Reference

`#include <PCMLock.h>`

Inheritance diagram for ParCompMark::Lock:



7.15.1 Detailed Description

Lock(p. 116) interface. Provides lock, trylock and unlock methods.

Definition at line 45 of file PCMLock.h.

Public Member Functions

- virtual void **lock** ()=0
- virtual bool **trylock** ()=0
- virtual void **unlock** ()=0

7.15.2 Member Function Documentation

7.15.2.1 virtual void ParCompMark::Lock::lock () [pure virtual]

Lock(p. 116) the lock.

Implemented in **ParCompMark::DummyLock** (p. 67), and **ParCompMark::Mutex** (p. 128).

7.15.2.2 virtual bool ParCompMark::Lock::trylock () [pure virtual]

Try locking the lock.

Returns:
True if the locking was successful.

Implemented in **ParCompMark::DummyLock** (p. 68), and **ParCompMark::Mutex** (p. 128).

7.15.2.3 virtual void ParCompMark::Lock::unlock () [pure virtual]

Unlock the lock.

Implemented in **ParCompMark::DummyLock** (p. 68), and **ParCompMark::Mutex** (p. 129).

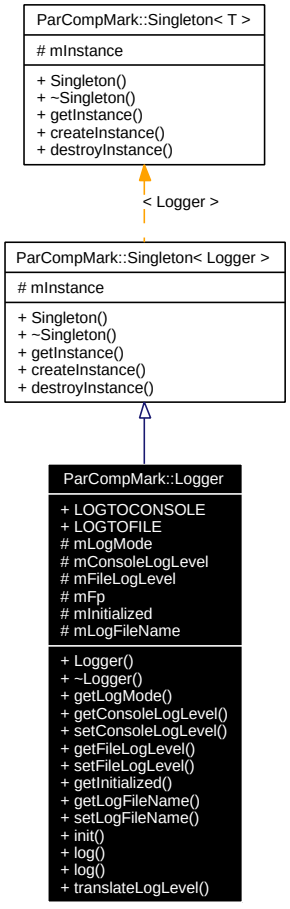
The documentation for this class was generated from the following file:

- **PCMLock.h**

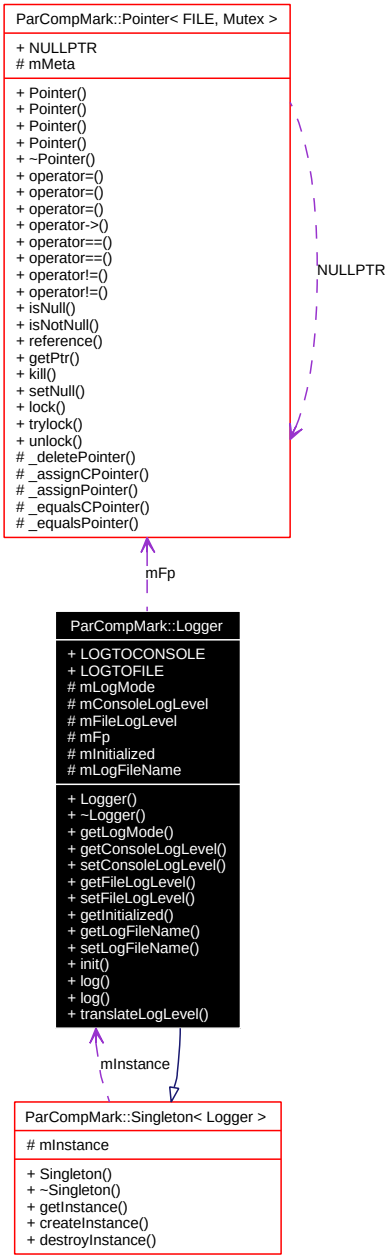
7.16 ParCompMark::Logger Class Reference

`#include <PCMLogger.h>`

Inheritance diagram for ParCompMark::Logger:



Collaboration diagram for ParCompMark::Logger:



7.16.1 Detailed Description

Class for very simple logging mechanism.

Definition at line 51 of file PCMLogger.h.

Public Types

- enum **LogLevel** {
  FATAL, **ERROR**, **WARNING**, **NOTICE**,
  DEBUG }

Public Member Functions

- **Logger** (const std::string &logFileName="pcm.log")
- virtual ~**Logger** ()
- const **u8** & **getLogMode** () const
- const **Logger::LogLevel** & **getConsoleLogLevel** () const
- void **setConsoleLogLevel** (const **Logger::LogLevel** &consoleloglevel)
- const **Logger::LogLevel** & **getFileLogLevel** () const
- void **setFileLogLevel** (const **Logger::LogLevel** &fileloglevel)
- const bool & **getInitialized** () const
- const std::string & **getLogFileName** () const
- void **setLogFileName** (const std::string &logfilename)
- virtual void **init** ()
- virtual void **log** (const **LogLevel** &loglevel, const std::string &message)
- virtual void **log** (const **Exception** &exception)

Static Public Member Functions

- static std::string **translateLogLevel** (const **LogLevel** &loglevel)

Static Public Attributes

- static const **u8 LOGTOCONSOLE** = 1
- static const **u8 LOGTOFILE** = 2

Protected Attributes

- **u8 mLogMode**
- **LogLevel mConsoleLogLevel**
- **LogLevel mFileLogLevel**
- **Pointer**< FILE, **Mutex** > **mFp**
- bool **mInitialized**
- std::string **mLogFileName**

7.16.2 Member Enumeration Documentation

7.16.2.1 enum ParCompMark::Logger::LogLevel

Definitions logging levels.

Enumerator:

FATAL Fatal error.

ERROR Error.

WARNING Warning.

NOTICE Notice.

DEBUG Debug.

Definition at line 68 of file PCMLogger.h.

7.16.3 Constructor & Destructor Documentation

7.16.3.1 ParCompMark::Logger::Logger (const std::string & *logFileName* = "pcm.log")

Create logger

Parameters:
← ***logFileName*** **Name**(p. 131) of the log file.

Definition at line 48 of file PCMLogger.cpp.

References DEBUG, ERROR, LOGTOCONSOLE, LOGTOFILE, mConsoleLogLevel, mFileLogLevel, mFp, mInitialized, mLogFileName, and mLogMode.

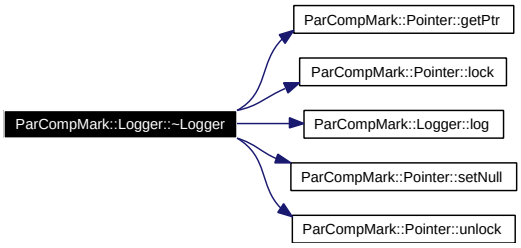
7.16.3.2 ParCompMark::Logger::~Logger () [virtual]

The destructor. This class has virtual destructor.

Definition at line 67 of file PCMLogger.cpp.

References ParCompMark::Pointer< T, Lock >::getPtr(), ParCompMark::Pointer< T, Lock >::lock(), log(), mFp, mInitialized, NOTICE, ParCompMark::Pointer< T, Lock >::setNull(), and ParCompMark::Pointer< T, Lock >::unlock().

Here is the call graph for this function:



7.16.4 Member Function Documentation

7.16.4.1 const Logger::LogLevel & ParCompMark::Logger::getConsoleLogLevel () const [inline]

Remarks:
Getter of mConsoleLogLevel.

Returns value of mConsoleLogLevel.

Returns:
The value of mConsoleLogLevel

Definition at line 344 of file PCMLogger.h.

7.16.4.2 const Logger::LogLevel & ParCompMark::Logger::getFileLogLevel () const [inline]

Remarks:
Getter of mFileLogLevel.

Returns value of mFileLogLevel.

Returns:
The value of mFileLogLevel

Definition at line 358 of file PCMLogger.h.

7.16.4.3 const bool & ParCompMark::Logger::getInitialized () const [inline]

Remarks:
Getter of mInitialized.

Returns value of mInitialized.

Returns:
The value of mInitialized

Definition at line 372 of file PCMLogger.h.

7.16.4.4 const std::string & ParCompMark::Logger::getLogFileName () const [inline]

Remarks:
Getter of mLogFileName.

Returns value of mLogFileName.

Returns:
The value of mLogFileName

Definition at line 379 of file PCMLogger.h.

7.16.4.5 const u8 & ParCompMark::Logger::getLogMode () const [inline]

Remarks:
Getter of mLogMode.

Returns value of mLogMode.

Returns:
The value of mLogMode

Definition at line 337 of file PCMLogger.h.

7.16.4.6 void ParCompMark::Logger::init () [virtual]

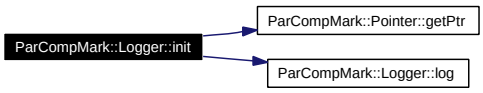
Initializes the **Logger**(p. 118).

Definition at line 88 of file PCMLogger.cpp.

References Assert, ParCompMark::Pointer< T, Lock >::getPtr(), log(), LOGTOFILE, mFp, mInitialized, mLogFileName, mLogMode, and NOTICE.

Referenced by ParCompMark::Application::initalize().

Here is the call graph for this function:



7.16.4.7 void ParCompMark::Logger::log (const Exception & *exception*) [inline, virtual]

Logs an exception.

Parameters:

← ***exception*** **Exception**(p. 73).

Definition at line 461 of file PCMLogger.h.

7.16.4.8 void ParCompMark::Logger::log (const LogLevel & *loglevel*, const std::string & *message*) [inline, virtual]

Logs a message.

Parameters:

← ***loglevel*** Message level.

← ***message*** Message string.

Definition at line 429 of file PCMLogger.h.

Referenced by ParCompMark::Network::acceptClientConnection(), ParCompMark::OldContainer::add(), ParCompMark::Context::Context(), ParCompMark::GLXRenderWindow::createWindow(), ParCompMark::Thread::entryPoint(), ParCompMark::Context::finalize(), ParCompMark::Application::finalize(), init(), ParCompMark::Network::initBroadcastRecieve(), ParCompMark::Network::initBroadcastSend(), ParCompMark::Network::initClient(), ParCompMark::Network::initNetwork(), ParCompMark::Network::initServer(), ParCompMark::Network::Network(), ParCompMark::Network::recieveBroadcastMessage(), ParCompMark::OldContainer::remove(), ParCompMark::Thread::shutDownThread(), ParCompMark::Thread::startThread(), ParCompMark::Thread::stopThread(), ParCompMark::Application::terminateHandler(), ParCompMark::DynLoad::unload(), ~Logger(), ParCompMark::OldContainer::~OldContainer(), and ParCompMark::Thread::~Thread().

7.16.4.9 void ParCompMark::Logger::setConsoleLogLevel (const Logger::LogLevel & *consoleloglevel*) [inline]

Remarks:

Setter of mConsoleLogLevel.

Sets value of mConsoleLogLevel.

Parameters:
← ***consoleloglevel*** The value of mConsoleLogLevel

Definition at line 351 of file PCMLogger.h.

7.16.4.10 void ParCompMark::Logger::setFileLogLevel (const Logger::LogLevel & *fileloglevel*) [inline]

Remarks:
Setter of mFileLogLevel.

Sets value of mFileLogLevel.

Parameters:
← ***fileloglevel*** The value of mFileLogLevel

Definition at line 365 of file PCMLogger.h.

7.16.4.11 void ParCompMark::Logger::setLogFileName (const std::string & *logfilename*) [inline]

Remarks:
Setter of mLogFileName.

Sets value of mLogFileName.

Parameters:
← ***logfilename*** The value of mLogFileName

Definition at line 386 of file PCMLogger.h.

7.16.4.12 std::string ParCompMark::Logger::translateLogLevel (const LogLevel & *loglevel*) [inline, static]

Translate log level to human readable.

Parameters:
← ***loglevel*** Logging level.

Returns:

Definition at line 397 of file PCMLogger.h.

7.16.5 Member Data Documentation

7.16.5.1 const u8 ParCompMark::Logger::LOGTOCONSOLE = 1 [static]

Indicates that the logger logs to console.

Remarks:
This is own attribute of this class.

Definition at line 98 of file PCMLogger.h.

Referenced by Logger().

7.16.5.2 const u8 ParCompMark::Logger::LOGTOFILE = 2 [static]

Indicates that the logger logs to a textfile.

Remarks:
This is own attribute of this class.

Definition at line 105 of file PCMLogger.h.

Referenced by init(), and Logger().

7.16.5.3 LogLevel ParCompMark::Logger::mConsoleLogLevel [protected]

Logging level for console (from FATAL to NOTICE).

Remarks:
This is own attribute of this class.

Definition at line 125 of file PCMLogger.h.

Referenced by Logger().

7.16.5.4 LogLevel ParCompMark::Logger::mFileLogLevel [protected]

Logging level for file (from FATAL to NOTICE).

Remarks:
This is own attribute of this class.

Definition at line 132 of file PCMLogger.h.

Referenced by Logger().

7.16.5.5 Pointer< FILE, Mutex > ParCompMark::Logger::mFp [protected]

File pointer.

Remarks:
This is own attribute of this class.

Definition at line 139 of file PCMLogger.h.

Referenced by init(), Logger(), and ~Logger().

7.16.5.6 bool ParCompMark::Logger::mInitialized [protected]

Indicates that the logger is initialized.

Remarks:
This is own attribute of this class.

Definition at line 146 of file PCMLogger.h.

Referenced by init(), Logger(), and ~Logger().

7.16.5.7 std::string ParCompMark::Logger::mLogFileName [protected]

Name(p. 131) of the log file.

Remarks:
This is own attribute of this class.

Definition at line 153 of file PCMLogger.h.

Referenced by init(), and Logger().

7.16.5.8 u8 ParCompMark::Logger::mLogMode [protected]

Logging mode (console and/or textfile).

Remarks:
This is own attribute of this class.

Definition at line 118 of file PCMLogger.h.

Referenced by init(), and Logger().

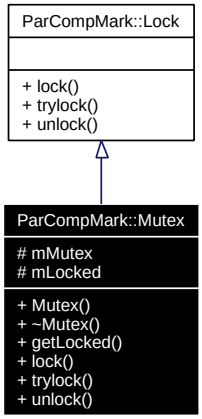
The documentation for this class was generated from the following files:

- **PCMLogger.h**
- **PCMLogger.cpp**

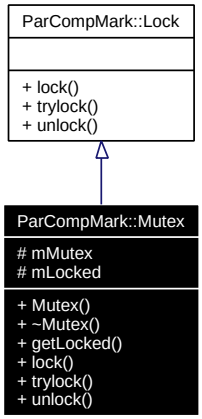
7.17 ParCompMark::Mutex Class Reference

`#include <PCMMutex.h>`

Inheritance diagram for ParCompMark::Mutex:



Collaboration diagram for ParCompMark::Mutex:



7.17.1 Detailed Description

Mutual exception lock.

Definition at line 51 of file PCMMutex.h.

Public Member Functions

- **Mutex** ()
- virtual ~**Mutex** ()
- const bool & **getLocked** () const
- virtual void **lock** ()
- virtual bool **trylock** ()
- virtual void **unlock** ()

Protected Attributes

- pthread_mutex_t **mMutex**
- bool **mLocked**

7.17.2 Constructor & Destructor Documentation

7.17.2.1 ParCompMark::Mutex::Mutex () [inline]

Default constructor.

Definition at line 170 of file PCMMutex.h.

7.17.2.2 ParCompMark::Mutex::~Mutex () [inline, virtual]

The destructor. This class has virtual destructor.

Definition at line 187 of file PCMMutex.h.

7.17.3 Member Function Documentation

7.17.3.1 const bool & ParCompMark::Mutex::getLocked () const [inline]

Remarks:
Getter of mLocked.

Returns value of mLocked.

Returns:
The value of mLocked

Definition at line 203 of file PCMMutex.h.

Referenced by ParCompMarkTest::TestMutex::test_constructor(), ParCompMarkTest::TestMutex::test_lock(), ParCompMarkTest::TestMutex::test_trylock(), and ParCompMarkTest::TestMutex::test_unlock().

7.17.3.2 void ParCompMark::Mutex::lock () [inline, virtual]

Lock(p. 116) the mutex.

Implements **ParCompMark::Lock** (p. 116).

Definition at line 214 of file PCMMutex.h.

Referenced by ParCompMarkTest::TestMutex::test_lock(), ParCompMarkTest::TestMutex::test_trylock(), and ParCompMarkTest::TestMutex::test_unlock().

7.17.3.3 bool ParCompMark::Mutex::trylock () [inline, virtual]

Try locking the mutex.

Returns:
True if the locking was successful.

Implements **ParCompMark::Lock** (p. 116).

Definition at line 222 of file PCMMutex.h.

7.17.3.4 void ParCompMark::Mutex::unlock () [inline, virtual]

Unlock the mutex.

Implements **ParCompMark::Lock** (p. 117).

Definition at line 230 of file PCMMutex.h.

Referenced by ParCompMarkTest::TestMutex::test_unlock().

7.17.4 Member Data Documentation

7.17.4.1 bool ParCompMark::Mutex::mLocked [protected]

Indicates that the mutex is locked.

Remarks:
This is own attribute of this class.

Definition at line 79 of file PCMMutex.h.

7.17.4.2 pthread_mutex_t ParCompMark::Mutex::mMutex [protected]

Guard mutex.

Remarks:
This is own attribute of this class.

Definition at line 72 of file PCMMutex.h.

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

- **PCMMutex.h**

7.18 ParCompMarkTest::MyClass Struct Reference

7.18.1 Detailed Description

Test class (for container test)

Definition at line 31 of file TestContainer.cpp.

Public Types

- typedef **ParCompMark::Pointer**< **MyClass**, **Mutex** > **Pointer**

Public Attributes

- int **i**

7.18.2 Member Typedef Documentation

7.18.2.1 typedef ParCompMark::Pointer< MyClass, Mutex > ParCompMarkTest::MyClass::Pointer

Definition at line 35 of file TestContainer.cpp.

7.18.3 Member Data Documentation

7.18.3.1 int ParCompMarkTest::MyClass::i

Definition at line 33 of file TestContainer.cpp.

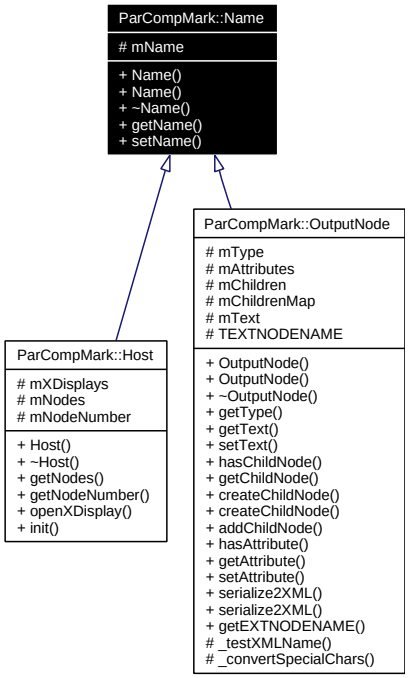
The documentation for this struct was generated from the following file:

- **TestContainer.cpp**

7.19 ParCompMark::Name Class Reference

`#include <PCMName.h>`

Inheritance diagram for ParCompMark::Name:



7.19.1 Detailed Description

This is a dummy class for inheritance. Contain grid data.

Definition at line 51 of file PCMName.h.

Public Member Functions

- **Name** ()
- **Name** (const std::string &name)
- virtual ~**Name** ()
- const std::string & **getName** () const
- void **setName** (const std::string &name)

Protected Attributes

- std::string **mName**

7.19.2 Constructor & Destructor Documentation

7.19.2.1 ParCompMark::Name::Name ()

Default constructor.

Definition at line 38 of file PCMName.cpp.

References mName.

7.19.2.2 ParCompMark::Name::Name (const std::string & *name*)

Default constructor.

Parameters:
← ***name*** **Name**(p. 131) of **Name**(p. 131):)

Definition at line 49 of file PCMName.cpp.

References mName.

7.19.2.3 ParCompMark::Name::~Name () [virtual]

The destructor. This class has virtual destructor.

Definition at line 60 of file PCMName.cpp.

7.19.3 Member Function Documentation

7.19.3.1 const std::string & ParCompMark::Name::getName () const [inline]

Remarks:
Getter of mName.

Returns value of mName.

Returns:
The value of mName

Definition at line 158 of file PCMName.h.

Referenced by ParCompMark::OldContainer::add(), and ParCompMark::OldContainer::remove().

7.19.3.2 void ParCompMark::Name::setName (const std::string & *name*) [inline]

Remarks:
Setter of mName.

Sets value of mName.

Parameters:
← ***name*** The value of mName

Definition at line 165 of file PCMName.h.

7.19.4 Member Data Documentation

7.19.4.1 std::string ParCompMark::Name::mName [protected]

Name(p. 131).

Remarks:
This is own attribute of this class.

Definition at line 72 of file PCMName.h.

Referenced by ParCompMark::Host::init(), Name(), ParCompMark::OutputNode::serialize2XML(), ParCompMarkTest::TestName::test_constructor(), ParCompMarkTest::TestOutputNode::test_constructor_cstd__string(), ParCompMarkTest::TestName::test_constructor_cstd__string(), and ParCompMarkTest::TestOutputNode::test_constructor_cstd__string_cNodeType().

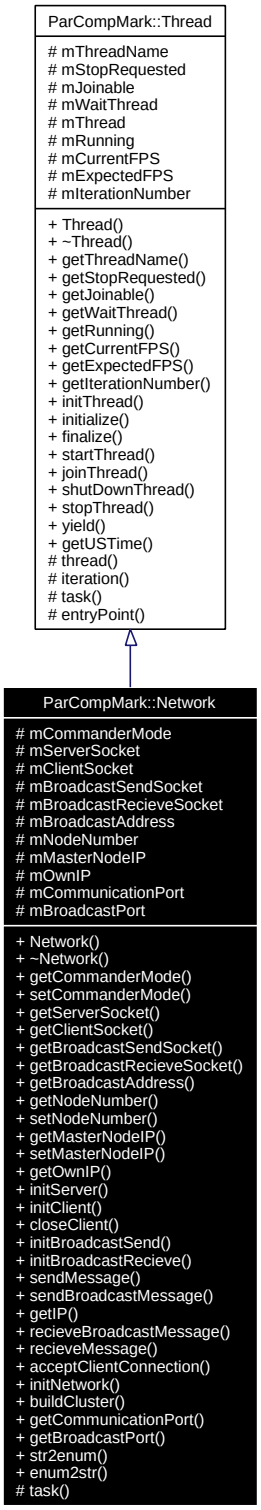
The documentation for this class was generated from the following files:

- **PCMName.h**
- **PCMName.cpp**

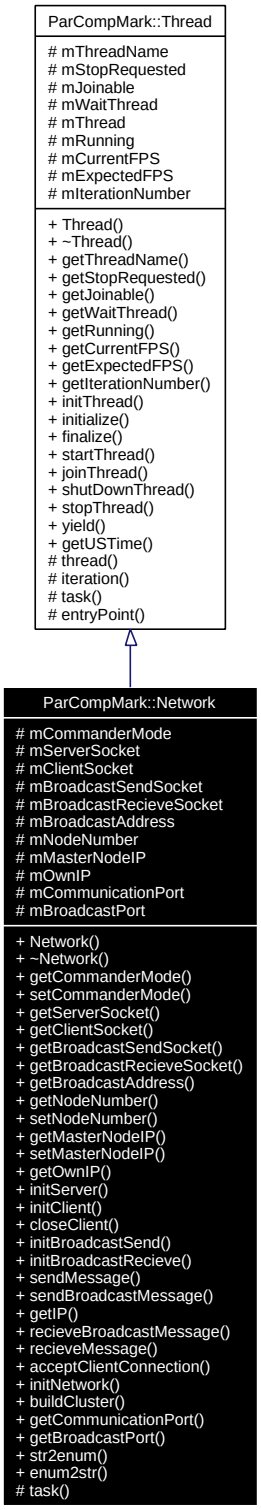
7.20 ParCompMark::Network Class Reference

`#include <PCMNetwork.h>`

Inheritance diagram for ParCompMark::Network:



Collaboration diagram for ParCompMark::Network:



7.20.1 Detailed Description

Class for network routines.

Definition at line 67 of file PCMNetwork.h.

Public Types

- typedef **Pointer**< **Network**, **DummyLock** > **Pointer**
- enum **MessageType** {
  INIT, **LOGIN**, **TIME**, **RESULT**,
  INITOK }

Public Member Functions

- **Network** (const std::string &name)
- virtual ~**Network** ()
- const bool & **getCommanderMode** () const
- void **setCommanderMode** (const bool &commandermode)
- const int & **getServerSocket** () const
- const int & **getClientSocket** () const
- const int & **getBroadcastSendSocket** () const
- const int & **getBroadcastRecieveSocket** () const
- const struct sockaddr_in & **getBroadcastAddress** () const
- const int & **getNodeNumber** () const
- void **setNodeNumber** (const int &nodenumber)
- const std::string & **getMasterNodeIP** () const
- void **setMasterNodeIP** (const std::string &masternodeip)
- const std::string & **getOwnIP** () const
- virtual void **initServer** ()
- virtual void **initClient** ()
- virtual void **closeClient** ()
- virtual void **initBroadcastSend** ()
- virtual void **initBroadcastRecieve** ()
- virtual void **sendMessage** (const **MessageType** &mType, const std::string &message)
- virtual void **sendBroadcastMessage** (const **MessageType** &mType, const std::string &message)
- virtual void **getIP** ()
- virtual void **recieveBroadcastMessage** ()
- virtual void **recieveMessage** ()
- virtual void **acceptClientConnection** ()
- virtual void **initNetwork** ()
- virtual void **buildCluster** ()

Static Public Member Functions

- static const unsigned short & **getCommunicationPort** ()
- static const unsigned short & **getBroadcastPort** ()
- static **Network::MessageType str2enum** (const std::string &strType)
- static std::string **enum2str** (const **Network::MessageType** &enumType)

Protected Member Functions

- virtual void **task** ()

Protected Attributes

- bool **mCommanderMode**
- int **mServerSocket**
- int **mClientSocket**
- int **mBroadcastSendSocket**
- int **mBroadcastRecieveSocket**
- sockaddr_in **mBroadcastAddress**
- int **mNodeNumber**
- std::string **mMasterNodeIP**
- std::string **mOwnIP**

Static Protected Attributes

- static unsigned short **mCommunicationPort** = 1234
- static unsigned short **mBroadcastPort** = 1235

7.20.2 Member Typedef Documentation

7.20.2.1 typedef Pointer< Network, DummyLock > ParCompMark::Network::Pointer

Type for pointer or this class.

Definition at line 111 of file PCMNetwork.h.

7.20.3 Member Enumeration Documentation

7.20.3.1 enum ParCompMark::Network::MessageType

Type of sended message.

Enumerator:

INIT Initiate string message.

LOGIN Client must log in.

TIME Client send their local time.

RESULT Client send their local time.

INITOK Wait for init message.

Definition at line 84 of file PCMNetwork.h.

7.20.4 Constructor & Destructor Documentation

7.20.4.1 ParCompMark::Network::Network (const std::string & *name*)

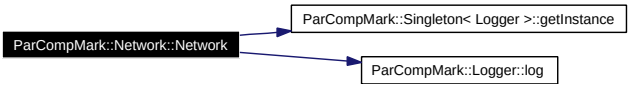
Create network class.

Parameters:
← ***name*** **Name**(p. 131) of the network.

Definition at line 45 of file PCMNetwork.cpp.

References ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Logger::log(), mBroadcastPort, mBroadcastRecieveSocket, mBroadcastSendSocket, mClientSocket, mCommanderMode, mCommunicationPort, mMasterNodeIP, mNodeNumber, mServerSocket, and ParCompMark::Logger::NOTICE.

Here is the call graph for this function:



7.20.4.2 ParCompMark::Network::~Network () [virtual]

The destructor. This class has virtual destructor.

Definition at line 80 of file PCMNetwork.cpp.

References mBroadcastRecieveSocket, mBroadcastSendSocket, mClientSocket, and mServerSocket.

7.20.5 Member Function Documentation

7.20.5.1 void ParCompMark::Network::acceptClientConnection () [virtual]

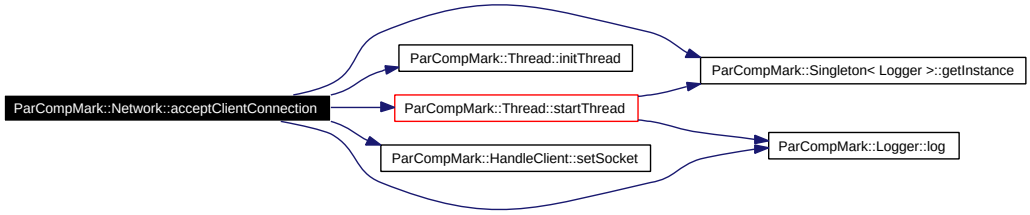
Server accept the client connections.

Definition at line 528 of file PCMNetwork.cpp.

References Except, ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Thread::initThread(), ParCompMark::Logger::log(), mServerSocket, ParCompMark::Logger::NOTICE, ParCompMark::HandleClient::setSocket(), and ParCompMark::Thread::startThread().

Referenced by task().

Here is the call graph for this function:



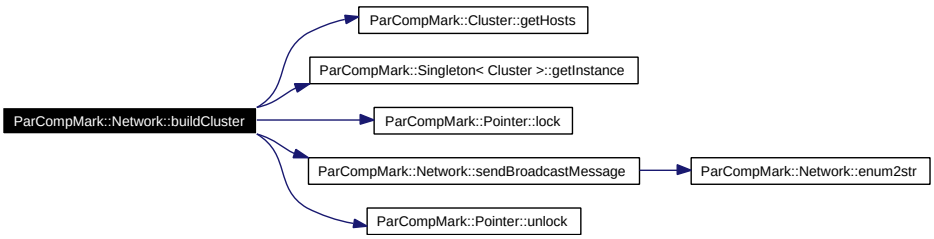
7.20.5.2 void ParCompMark::Network::buildCluster () [virtual]

Build cluster information.

Definition at line 576 of file PCMNetwork.cpp.

References ParCompMark::Cluster::getHosts(), ParCompMark::Singleton< Cluster >::getInstance(), ParCompMark::Pointer< T, Lock >::lock(), LOGIN, sendBroadcastMessage(), and ParCompMark::Pointer< T, Lock >::unlock().

Here is the call graph for this function:



7.20.5.3 void ParCompMark::Network::closeClient () [virtual]

Init a client process.

Definition at line 270 of file PCMNetwork.cpp.

References Assert, and mClientSocket.

Referenced by recieveBroadcastMessage().

7.20.5.4 std::string ParCompMark::Network::enum2str (const Network::MessageType & *enumType*) [static]

String convert to enum.

Parameters:

← ***enumType*** Type as enum.

Returns:

Type as string.

Definition at line 138 of file PCMNetwork.cpp.

References INIT, INITOK, LOGIN, RESULT, and TIME.

Referenced by sendBroadcastMessage(), and sendMessage().

7.20.5.5 const struct sockaddr_in & ParCompMark::Network::getBroadcastAddress () const [inline]

Remarks:

Getter of mBroadcastAddress.

Returns value of mBroadcastAddress.

Returns:
The value of mBroadcastAddress

Definition at line 571 of file PCMNetwork.h.

7.20.5.6 const unsigned short & ParCompMark::Network::getBroadcastPort () [inline, static]

Remarks:
Getter of mBroadcastPort.

Returns value of mBroadcastPort.

Returns:
The value of mBroadcastPort

Definition at line 536 of file PCMNetwork.h.

7.20.5.7 const int & ParCompMark::Network::getBroadcastRecieveSocket () const [inline]

Remarks:
Getter of mBroadcastRecieveSocket.

Returns value of mBroadcastRecieveSocket.

Returns:
The value of mBroadcastRecieveSocket

Definition at line 564 of file PCMNetwork.h.

7.20.5.8 const int & ParCompMark::Network::getBroadcastSendSocket () const [inline]

Remarks:
Getter of mBroadcastSendSocket.

Returns value of mBroadcastSendSocket.

Returns:
The value of mBroadcastSendSocket

Definition at line 557 of file PCMNetwork.h.

7.20.5.9 const int & ParCompMark::Network::getClientSocket () const [inline]

Remarks:
Getter of mClientSocket.

Returns value of mClientSocket.

Returns:
The value of mClientSocket

Definition at line 550 of file PCMNetwork.h.

7.20.5.10 const bool & ParCompMark::Network::getCommanderMode () const [inline]

Remarks:
Getter of mCommanderMode.

Returns value of mCommanderMode.

Returns:
The value of mCommanderMode

Definition at line 515 of file PCMNetwork.h.

7.20.5.11 const unsigned short & ParCompMark::Network::getCommunicationPort () [inline, static]

Remarks:
Getter of mCommunicationPort.

Returns value of mCommunicationPort.

Returns:
The value of mCommunicationPort

Definition at line 529 of file PCMNetwork.h.

7.20.5.12 void ParCompMark::Network::getIP () [virtual]

Get local machine IP address-es.

Definition at line 383 of file PCMNetwork.cpp.

7.20.5.13 const std::string & ParCompMark::Network::getMasterNodeIP () const [inline]

Remarks:
Getter of mMasterNodeIP.

Returns value of mMasterNodeIP.

Returns:
The value of mMasterNodeIP

Definition at line 592 of file PCMNetwork.h.

7.20.5.14 const int & ParCompMark::Network::getNodeNumber () const [inline]

Remarks:
Getter of mNodeNumber.

Returns value of mNodeNumber.

Returns:
The value of mNodeNumber

Definition at line 578 of file PCMNetwork.h.

7.20.5.15 const std::string & ParCompMark::Network::getOwnIP () const [inline]

Remarks:
Getter of mOwnIP.

Returns value of mOwnIP.

Returns:
The value of mOwnIP

Definition at line 606 of file PCMNetwork.h.

7.20.5.16 const int & ParCompMark::Network::getServerSocket () const [inline]

Remarks:
Getter of mServerSocket.

Returns value of mServerSocket.

Returns:
The value of mServerSocket

Definition at line 543 of file PCMNetwork.h.

7.20.5.17 void ParCompMark::Network::initBroadcastRecieve () [virtual]

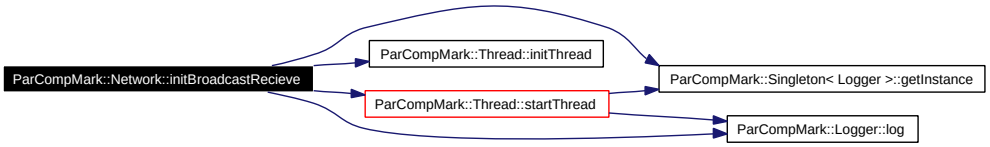
Init a broadcast reciever process.

Definition at line 311 of file PCMNetwork.cpp.

References Except, ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Thread::initThread(), ParCompMark::Logger::log(), mBroadcastPort, mBroadcastRecieveSocket, ParCompMark::Logger::NOTICE, and ParCompMark::Thread::startThread().

Referenced by initNetwork(), and ParCompMarkTest::TestNetwork::test_initBroadcastSend().

Here is the call graph for this function:



7.20.5.18 void ParCompMark::Network::initBroadcastSend () [virtual]

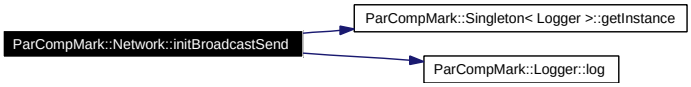
Init a broadcast sender process.

Definition at line 283 of file PCMNetwork.cpp.

References Except, ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Logger::log(), mBroadcastAddress, mBroadcastPort, mBroadcastSendSocket, and ParCompMark::Logger::NOTICE.

Referenced by initNetwork(), and ParCompMarkTest::TestNetwork::test_initBroadcastSend().

Here is the call graph for this function:



7.20.5.19 void ParCompMark::Network::initClient () [virtual]

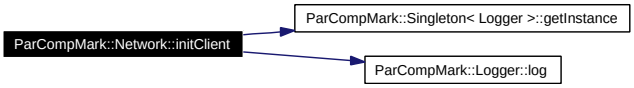
Init a client process.

Definition at line 237 of file PCMNetwork.cpp.

References Assert, Except, ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Logger::log(), mClientSocket, mCommunicationPort, mMasterNodeIP, and ParCompMark::Logger::NOTICE.

Referenced by recieveBroadcastMessage(), and ParCompMarkTest::TestNetwork::test_initClient().

Here is the call graph for this function:



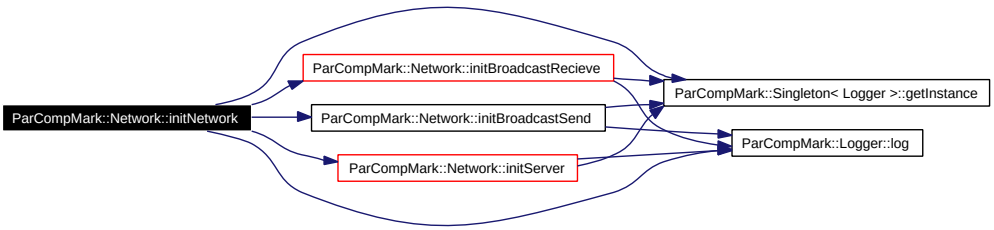
7.20.5.20 void ParCompMark::Network::initNetwork () [virtual]

Initial the network interface.

Definition at line 556 of file PCMNetwork.cpp.

References ParCompMark::Singleton< Logger >::getInstance(), initBroadcastRecieve(), initBroadcastSend(), initServer(), ParCompMark::Logger::log(), mCommanderMode, and ParCompMark::Logger::NOTICE.

Here is the call graph for this function:



7.20.5.21 void ParCompMark::Network::initServer () [virtual]

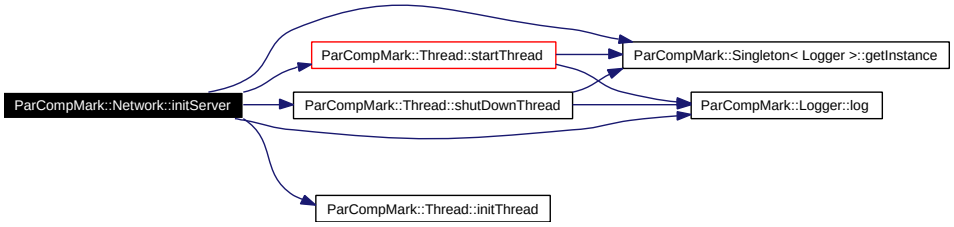
Init a server process.

Definition at line 183 of file PCMNetwork.cpp.

References Except, ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Thread::initThread(), ParCompMark::Logger::log(), mCommunicationPort, mNodeNumber, ParCompMark::Thread::mRunning, mServerSocket, ParCompMark::Logger::NOTICE, ParCompMark::Thread::shutDownThread(), and ParCompMark::Thread::startThread().

Referenced by initNetwork(), and ParCompMarkTest::TestNetwork::test_initClient().

Here is the call graph for this function:



7.20.5.22 void ParCompMark::Network::recieveBroadcastMessage () [virtual]

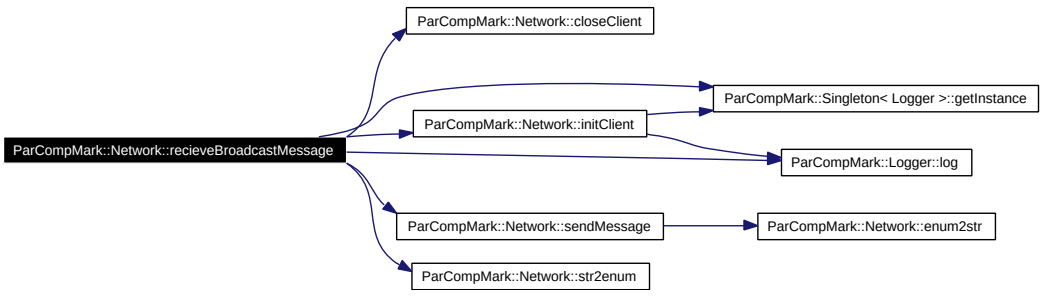
Wait for a broadcast message.

Definition at line 417 of file PCMNetwork.cpp.

References closeClient(), Except, ParCompMark::Singleton< Logger >::getInstance(), INIT, initClient(), INITOK, ParCompMark::Logger::log(), LOGIN, mBroadcastRecieveSocket, mMasterNodeIP, ParCompMark::Logger::NOTICE, RESULT, sendMessage(), str2enum(), and TIME.

Referenced by task().

Here is the call graph for this function:



7.20.5.23 void ParCompMark::Network::recieveMessage () [virtual]

Wait for a TCP/IP message.

Definition at line 489 of file PCMNetwork.cpp.

References Assert, and mClientSocket.

7.20.5.24 void ParCompMark::Network::sendBroadcastMessage (const MessageType & *mType*, const std::string & *message*) [virtual]

Send a broadcast message.

Parameters:

← ***mType*** Type of the message.

← ***message*** The message.

Definition at line 360 of file PCMNetwork.cpp.

References enum2str(), Except, mBroadcastAddress, and mBroadcastSendSocket.

Referenced by buildCluster(), and ParCompMarkTest::TestNetwork::test_initBroadcastSend().

Here is the call graph for this function:

7.20.5.25 void ParCompMark::Network::sendMessage (const MessageType & *mType*, const std::string & *message*) [virtual]

Send message with TCP protocol.

Parameters:

← ***mType*** Type of the message.

← ***message*** The message.

Definition at line 340 of file PCMNetwork.cpp.

References enum2str(), Except, and mClientSocket.

Referenced by recieveBroadcastMessage().

Here is the call graph for this function:

7.20.5.26 void ParCompMark::Network::setCommanderMode (const bool & *commandermode*) [inline]

Remarks:

Setter of mCommanderMode.

Sets value of mCommanderMode.

Parameters:

← ***commandermode*** The value of mCommanderMode

Definition at line 522 of file PCMNetwork.h.

Referenced by ParCompMarkTest::TestNetwork::test_initBroadcastSend(), and ParCompMarkTest::TestNetwork::test_initClient().

7.20.5.27 void ParCompMark::Network::setMasterNodeIP (const std::string & *masternodeip*) [inline]

Remarks:
Setter of mMasterNodeIP.

Sets value of mMasterNodeIP.

Parameters:
← ***masternodeip*** The value of mMasterNodeIP

Definition at line 599 of file PCMNetwork.h.

7.20.5.28 void ParCompMark::Network::setNodeNumber (const int & *nodenumber*) [inline]

Remarks:
Setter of mNodeNumber.

Sets value of mNodeNumber.

Parameters:
← ***nodenumber*** The value of mNodeNumber

Definition at line 585 of file PCMNetwork.h.

Referenced by ParCompMarkTest::TestNetwork::test_initClient().

7.20.5.29 Network::MessageType ParCompMark::Network::str2enum (const std::string & *strType*) [static]

String convert to enum.

Parameters:
← ***strType*** Type as string.

Returns:
Type as enum.

Definition at line 114 of file PCMNetwork.cpp.

References INIT, INITOK, LOGIN, RESULT, and TIME.

Referenced by recieveBroadcastMessage(), and ParCompMark::HandleClient::task().

7.20.5.30 void ParCompMark::Network::task () [protected, virtual]

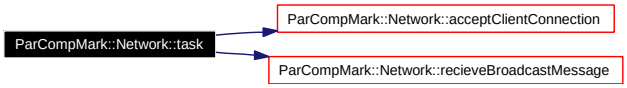
Thread(p. 264) for accept calling.

Reimplemented from **ParCompMark::Thread** (p. 271).

Definition at line 614 of file PCMNetwork.cpp.

References acceptClientConnection(), mCommanderMode, and recieveBroadcastMessage().

Here is the call graph for this function:



7.20.6 Member Data Documentation

7.20.6.1 struct sockaddr_in ParCompMark::Network::mBroadcastAddress [protected]

The broadcast address structure for send later time.

Remarks:
This is own attribute of this class.

Definition at line 179 of file PCMNetwork.h.

Referenced by initBroadcastSend(), and sendBroadcastMessage().

7.20.6.2 unsigned short ParCompMark::Network::mBroadcastPort = 1235 [static, protected]

The broadcast port number.

Remarks:
This is own attribute of this class.

Definition at line 131 of file PCMNetwork.h.

Referenced by initBroadcastRecieve(), initBroadcastSend(), and Network().

7.20.6.3 int ParCompMark::Network::mBroadcastRecieveSocket [protected]

The broadcast socket number.

Remarks:
This is own attribute of this class.

Definition at line 172 of file PCMNetwork.h.

Referenced by initBroadcastRecieve(), Network(), recieveBroadcastMessage(), and ~Network().

7.20.6.4 int ParCompMark::Network::mBroadcastSendSocket [protected]

The broadcast socket number.

Remarks:
This is own attribute of this class.

Definition at line 165 of file PCMNetwork.h.

Referenced by initBroadcastSend(), Network(), sendBroadcastMessage(), and ~Network().

7.20.6.5 int ParCompMark::Network::mClientSocket [protected]

The client socket number.

Remarks:
This is own attribute of this class.

Definition at line 158 of file PCMNetwork.h.

Referenced by closeClient(), initClient(), Network(), recieveMessage(), sendMessage(), and ~Network().

7.20.6.6 bool ParCompMark::Network::mCommanderMode [protected]

Indicates commander mode (default false).

Remarks:
This is own attribute of this class.

Definition at line 144 of file PCMNetwork.h.

Referenced by initNetwork(), Network(), task(), and ParCompMarkTest::TestNetwork::test_initBroadcast-Send().

7.20.6.7 unsigned short ParCompMark::Network::mCommunicationPort = 1234 [static, protected]

The communication port number.

Remarks:
This is own attribute of this class.

Definition at line 124 of file PCMNetwork.h.

Referenced by initClient(), initServer(), and Network().

7.20.6.8 std::string ParCompMark::Network::mMasterNodeIP [protected]

IP address of the master node.

Remarks:
This is own attribute of this class.

Definition at line 193 of file PCMNetwork.h.

Referenced by initClient(), Network(), and recieveBroadcastMessage().

7.20.6.9 int ParCompMark::Network::mNodeNumber [protected]

Number of the nodes in the grid.

Remarks:
This is own attribute of this class.

Definition at line 186 of file PCMNetwork.h.

Referenced by initServer(), and Network().

7.20.6.10 std::string ParCompMark::Network::mOwnIP [protected]

IP address of the master node.

Remarks:
This is own attribute of this class.

Definition at line 200 of file PCMNetwork.h.

7.20.6.11 int ParCompMark::Network::mServerSocket [protected]

The server socket number.

Remarks:
This is own attribute of this class.

Definition at line 151 of file PCMNetwork.h.

Referenced by acceptClientConnection(), initServer(), Network(), and ~Network().

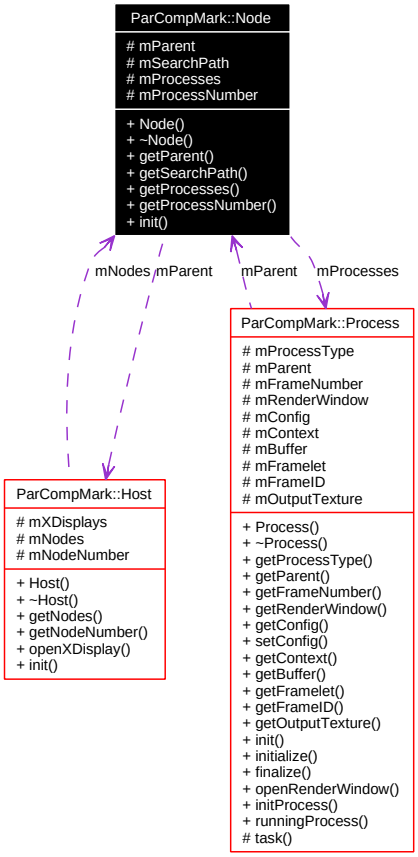
The documentation for this class was generated from the following files:

- **PCMNetwork.h**
- **PCMNetwork.cpp**

7.21 ParCompMark::Node Class Reference

`#include <PCMNode.h>`

Collaboration diagram for ParCompMark::Node:



7.21.1 Detailed Description

Node(p. 150) class. Entity for composite and/or render a framelet.

Definition at line 55 of file PCMNode.h.

Public Member Functions

- **Node** (**Host** ∗parent)
- virtual ∼**Node** ()
- **Host** ∗ **getParent** () const
- const PCstring & **getSearchPath** () const
- **Process** ∗∗ **getProcesses** () const
- const int & **getProcessNumber** () const
- virtual void **init** (const std::string &conf)

Protected Attributes

- **Host** ∗ **mParent**
- PCstring **mSearchPath**
- **Process** ∗∗ **mProcesses**
- int **mProcessNumber**

7.21.2 Constructor & Destructor Documentation

7.21.2.1 ParCompMark::Node::Node (Host ∗ *parent*)

Create node. Normally **Host**(p. 112) calles this constructor.

Parameters:
← ***parent*** Parent host

Definition at line 38 of file PCMNode.cpp.

References Assert, mProcesses, and mProcessNumber.

7.21.2.2 ParCompMark::Node::∼Node () [virtual]

The destructor. This class has virtual destructor.

Definition at line 57 of file PCMNode.cpp.

References mProcesses, and mProcessNumber.

7.21.3 Member Function Documentation

7.21.3.1 Host ∗ ParCompMark::Node::getParent () const [inline]

Remarks:
Getter of mParent.

Returns value of mParent.

Returns:
The value of mParent

Definition at line 212 of file PCMNode.h.

7.21.3.2 Process ∗∗ ParCompMark::Node::getProcesses () const [inline]

Remarks:
Getter of mProcesses.

Returns value of mProcesses.

Returns:
The value of mProcesses

Definition at line 226 of file PCMNode.h.

7.21.3.3 const int & ParCompMark::Node::getProcessNumber () const [inline]

Remarks:
Getter of mProcessNumber.

Returns value of mProcessNumber.

Returns:
The value of mProcessNumber

Definition at line 233 of file PCMNode.h.

7.21.3.4 const PCstring & ParCompMark::Node::getSearchPath () const [inline]

Remarks:
Getter of mSearchPath.

Returns value of mSearchPath.

Returns:
The value of mSearchPath

Definition at line 219 of file PCMNode.h.

7.21.3.5 void ParCompMark::Node::init (const std::string & *conf*) [virtual]

Create and call init functions of the nodes on the hosts.

Parameters:
← ***conf*** Config string.

Definition at line 76 of file PCMNode.cpp.

References mProcesses, and mProcessNumber.

7.21.4 Member Data Documentation

7.21.4.1 Host* ParCompMark::Node::mParent [protected]

Parent host of the node. (Parent reference is standard pointer to avoid circular reference)

Remarks:
This attribute references an attribute.

Definition at line 76 of file PCMNode.h.

7.21.4.2 Process** ParCompMark::Node::mProcesses [protected]

Processes of the node.

Remarks:
This is own attribute of this class.

Definition at line 90 of file PCMNode.h.

Referenced by init(), Node(), and ~Node().

7.21.4.3 int ParCompMark::Node::mProcessNumber [protected]

Number of processes.

Remarks:
This is own attribute of this class.

Definition at line 97 of file PCMNode.h.

Referenced by init(), Node(), and ~Node().

7.21.4.4 PCstring ParCompMark::Node::mSearchPath [protected]

Search path of PC library.

Remarks:
This is own attribute of this class.

Definition at line 83 of file PCMNode.h.

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

- **PCMNode.h**
- **PCMNode.cpp**

7.22 ParCompMark::OldContainer Class Reference

`#include <PCMOldContainer.h>`

7.22.1 Detailed Description

Class for contain something in a hashmap.

Definition at line 55 of file PCMOldContainer.h.

Public Types

- typedef **Pointer**< **OldContainer**, **Mutex** > **Pointer**
- typedef std::map< std::string, **Name** * > **Map**
- typedef Map::iterator **Iterator**

Public Member Functions

- **OldContainer** ()
- virtual ~**OldContainer** ()
- const int & **getElementNumber** () const
- virtual void **add** (**Name** *&element)
- virtual void **add** (std::string name)
- virtual **Name** * **get** (const std::string &name)
- virtual bool **has** (const std::string &name)
- virtual void **remove** (const std::string &name)
- virtual void **remove** (**Name** *element)

Protected Attributes

- **Map mElements**
- int **mElementNumber**

7.22.2 Member Typedef Documentation

7.22.2.1 typedef Map::iterator ParCompMark::OldContainer::Iterator

Type definition for iterator on map of elements.

Definition at line 80 of file PCMOldContainer.h.

7.22.2.2 typedef std::map< std::string, Name * > ParCompMark::OldContainer::Map

Type definition for map of elements.

Definition at line 77 of file PCMOldContainer.h.

7.22.2.3 typedef Pointer< OldContainer, Mutex > ParCompMark::OldContainer::Pointer

Type for pointer on that class.

Definition at line 73 of file PCMOldContainer.h.

7.22.3 Constructor & Destructor Documentation

7.22.3.1 ParCompMark::OldContainer::OldContainer ()

Default constructor.

Definition at line 38 of file PCMOldContainer.cpp.

References mElementNumber.

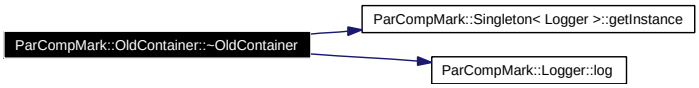
7.22.3.2 ParCompMark::OldContainer::~OldContainer () [virtual]

The destructor. This class has virtual destructor.

Definition at line 51 of file PCMOldContainer.cpp.

References ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Logger::log(), mElementNumber, mElements, and ParCompMark::Logger::NOTICE.

Here is the call graph for this function:



7.22.4 Member Function Documentation

7.22.4.1 void ParCompMark::OldContainer::add (std::string *name*) [virtual]

Add an element by name.

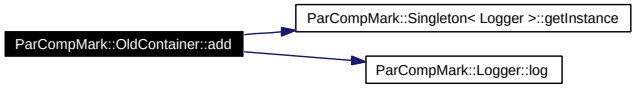
Parameters:

← ***name*** name of the element

Definition at line 84 of file PCMOldContainer.cpp.

References Assert, ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Logger::log(), mElementNumber, mElements, and ParCompMark::Logger::NOTICE.

Here is the call graph for this function:



7.22.4.2 void ParCompMark::OldContainer::add (Name *& *element*) [virtual]

Add an element by object.

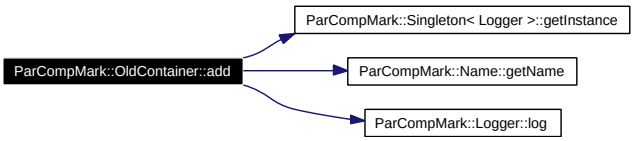
Parameters:
→ ***element*** pointer to the element

Definition at line 69 of file PCMOldContainer.cpp.

References Assert, ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Name::getName(), ParCompMark::Logger::log(), mElementNumber, mElements, and ParCompMark::Logger::NOTICE.

Referenced by ParCompMarkTest::TestOldContainer::test_add_Name_p(), ParCompMarkTest::TestOldContainer::test_add_std__string(), ParCompMarkTest::TestOldContainer::test_get_cstd__string(), and ParCompMarkTest::TestOldContainer::test_has_cstd__string().

Here is the call graph for this function:



7.22.4.3 Name * ParCompMark::OldContainer::get (const std::string & *name*) [virtual]

Get an element by name.

Parameters:
← ***name*** name of the element

Returns:
Pointer(p. 170) to the element

Definition at line 100 of file PCMOldContainer.cpp.

References Assert, and mElements.

Referenced by ParCompMarkTest::TestOldContainer::test_get_cstd__string().

7.22.4.4 const int & ParCompMark::OldContainer::getElementNumber () const [inline]

Remarks:
Getter of mElementNumber.

Returns value of mElementNumber.

Returns:
The value of mElementNumber

Definition at line 213 of file PCMOldContainer.h.

7.22.4.5 bool ParCompMark::OldContainer::has (const std::string & *name*) [virtual]

Search for an element by name. Already exist?

Parameters:
← ***name*** name of the element

Returns:
True is the **OldContainer**(p. 154) has the element.

Definition at line 114 of file PCMOldContainer.cpp.

References mElements.

Referenced by ParCompMarkTest::TestOldContainer::test_has_cstd__string().

7.22.4.6 void ParCompMark::OldContainer::remove (Name * *element*) [virtual]

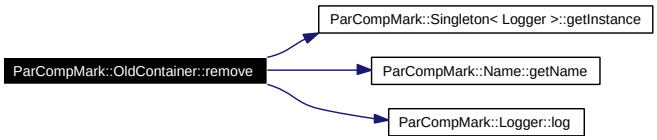
Remove an element by pointer to the removed element.

Parameters:
← ***element*** name of the element

Definition at line 144 of file PCMOldContainer.cpp.

References Assert, ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Name::getName(), ParCompMark::Logger::log(), mElementNumber, mElements, and ParCompMark::Logger::NOTICE.

Here is the call graph for this function:



7.22.4.7 void ParCompMark::OldContainer::remove (const std::string & *name*) [virtual]

Remove an element by name.

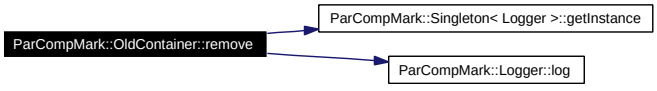
Parameters:
← ***name*** name of the element

Definition at line 125 of file PCMOldContainer.cpp.

References Assert, ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Logger::log(), mElementNumber, mElements, and ParCompMark::Logger::NOTICE.

Referenced by ParCompMarkTest::TestOldContainer::test_add_Name_p(), and ParCompMarkTest::TestOldContainer::test_add_std__string().

Here is the call graph for this function:



7.22.5 Member Data Documentation

7.22.5.1 int ParCompMark::OldContainer::mElementNumber [protected]

Number of elements.

Remarks:
This is own attribute of this class.

Definition at line 100 of file PCMOldContainer.h.

Referenced by add(), OldContainer(), remove(), ParCompMarkTest::TestOldContainer::test_add_std__-string(), and ~OldContainer().

7.22.5.2 Map ParCompMark::OldContainer::mElements [protected]

Map of elements.

Remarks:
This is own attribute of this class.

Definition at line 93 of file PCMOldContainer.h.

Referenced by add(), get(), has(), remove(), and ~OldContainer().

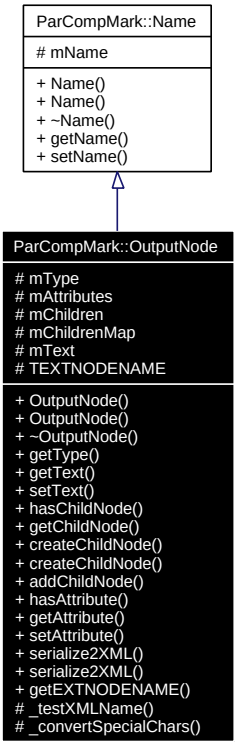
The documentation for this class was generated from the following files:

- **PCMOldContainer.h**
- **PCMOldContainer.cpp**

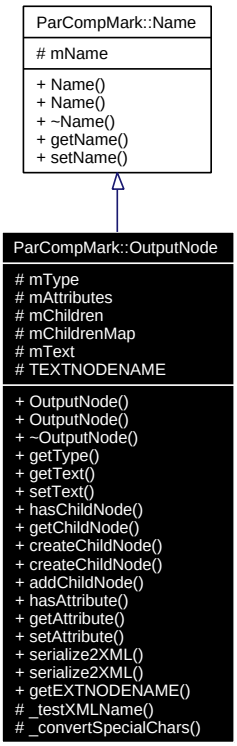
7.23 ParCompMark::OutputNode Class Reference

`#include <PCMOutputNode.h>`

Inheritance diagram for ParCompMark::OutputNode:



Collaboration diagram for ParCompMark::OutputNode:



7.23.1 Detailed Description

Node(p. 150) of the output document.

Definition at line 59 of file PCMOutputNode.h.

Public Types

- typedef **Pointer**< **OutputNode**, **Mutex** > **Pointer**
- typedef std::list< **OutputNode::Pointer** > **ChildNodeList**
- typedef ChildNodeList::iterator **ChildNodeListIterator**
- typedef std::map< std::string, **OutputNode::Pointer** > **ChildNodeMap**
- typedef ChildNodeMap::iterator **ChildNodeMapIterator**
- typedef std::map< std::string, std::string > **AttributeMap**
- typedef AttributeMap::const_iterator **AttributeMapIterator**
- enum **NodeType** {

  DEFINITION, **INFORMATION**, **REFERENCE**, **STATISTICS**,

  TEXT }

Public Member Functions

- **OutputNode** (const std::string &name, const **NodeType** &type)
- **OutputNode** (const std::string &text)
- virtual ~**OutputNode** ()
- const **OutputNode::NodeType** & **getType** () const

- const std::string & **getText** () const
- void **setText** (const std::string &text)
- virtual bool **hasChildNode** (const std::string &name)
- virtual **OutputNode::Pointer getChildNode** (const std::string &name)
- virtual **OutputNode::Pointer createChildNode** (const std::string &name, const **NodeType** &type)
- virtual **OutputNode::Pointer createChildNode** (const std::string &text)
- virtual void **addChildNode** (**OutputNode::Pointer** &child)
- virtual bool **hasAttribute** (const std::string &attribute) const
- virtual const std::string & **getAttribute** (const std::string &attribute) const
- virtual void **setAttribute** (const std::string &attribute, const std::string &value)
- virtual std::ostringstream & **serialize2XML** (std::ostringstream &osstr)
- virtual std::string **serialize2XML** ()

Static Public Member Functions

- static const std::string & **getEXTNODENAME** ()

Static Protected Member Functions

- static bool **_testXMLName** (const std::string &name)
- static std::string **_convertSpecialChars** (const std::string &string)

Protected Attributes

- **NodeType mType**
- **AttributeMap mAttributes**
- **ChildNodeList mChildren**
- **ChildNodeMap mChildrenMap**
- std::string **mText**

Static Protected Attributes

- static const std::string **TEXTNODENAME** = "#text"

7.23.2 Member Typedef Documentation

7.23.2.1 typedef std::map< std::string, std::string > ParCompMark::OutputNode::AttributeMap

Type definition for map of attributes.

Definition at line 120 of file PCMOutputNode.h.

7.23.2.2 typedef AttributeMap::const_iterator ParCompMark::OutputNode::AttributeMap-Iterator

Type definition for iterator on map of attributes.

Definition at line 123 of file PCMOutputNode.h.

7.23.2.3 typedef std::list< OutputNode::Pointer > ParCompMark::OutputNode::ChildNodeList

Type definition for list of child nodes.

Definition at line 106 of file PCMOutputNode.h.

7.23.2.4 typedef ChildNodeList::iterator ParCompMark::OutputNode::ChildNodeListIterator

Type definition for iterator on list of child nodes.

Definition at line 109 of file PCMOutputNode.h.

7.23.2.5 typedef std::map< std::string, OutputNode::Pointer > ParCompMark::OutputNode::ChildNodeMap

Type definition for map of child nodes (works in collaboration with ChildNodeList, and easies the element test in the list).

Definition at line 113 of file PCMOutputNode.h.

7.23.2.6 typedef ChildNodeMap::iterator ParCompMark::OutputNode::ChildNodeMapIterator

Type definition for iterator on map of child nodes.

Definition at line 116 of file PCMOutputNode.h.

7.23.2.7 typedef Pointer< OutputNode, Mutex > ParCompMark::OutputNode::Pointer

Type for pointer on that class.

Definition at line 103 of file PCMOutputNode.h.

7.23.3 Member Enumeration Documentation

7.23.3.1 enum ParCompMark::OutputNode::NodeType

Node(p. 150) type definitions.

Enumerator:

DEFINITION Define something (cluster, host, node, etc.), anything that can be described in the script or in the GUI.

INFORMATION Information about the hardware or software element. The user cannot redefine its value.

REFERENCE A reference, new element cannot be defined, only referencing an existing element is possible.

STATISTICS Statistics, it cannot be defined, referenced, or determined from the hardware directly. These values are measured during the benchmark process.

TEXT The node is a text field.

Definition at line 76 of file PCMOutputNode.h.

7.23.4 Constructor & Destructor Documentation

7.23.4.1 ParCompMark::OutputNode::OutputNode (const std::string & *name*, const NodeType & *type*)

Creates output node.

Parameters:
← ***name*** **Name**(p. 131) of the node.
← ***type*** Type of the node.

Definition at line 44 of file PCMOutputNode.cpp.

References _testXMLName(), and Assert.

Referenced by createChildNode().

Here is the call graph for this function:

7.23.4.2 ParCompMark::OutputNode::OutputNode (const std::string & *text*)

Creates text field.

Parameters:
← ***text*** Text data.

Definition at line 65 of file PCMOutputNode.cpp.

7.23.4.3 ParCompMark::OutputNode::~OutputNode () `[virtual]`

The destructor. This class has virtual destructor.

Definition at line 87 of file PCMOutputNode.cpp.

7.23.5 Member Function Documentation

7.23.5.1 std::string ParCompMark::OutputNode::_convertSpecialChars (const std::string & *string*) `[static, protected]`

Convert special characters to XML entites.

Parameters:
← ***string*** String to convert.

Returns:
Converted string

Definition at line 107 of file PCMOutputNode.cpp.

Referenced by serialize2XML().

7.23.5.2 bool ParCompMark::OutputNode::_testXMLName (const std::string & *name*) [static, protected]

Test name if its a correct xml name.

Parameters:
← ***name*** **Name**(p. 131) to test.

Returns:
True if the specified name is a correct XML name

Definition at line 98 of file PCMOutputNode.cpp.

Referenced by OutputNode(), and setAttribute().

7.23.5.3 void ParCompMark::OutputNode::addChildNode (OutputNode::Pointer & *child*) [virtual]

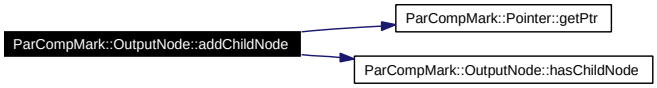
Add child node.

Parameters:
→ ***child*** Child node.

Definition at line 171 of file PCMOutputNode.cpp.

References Assert, ParCompMark::Pointer< T, Lock >::getPtr(), hasChildNode(), mChildren, mChildrenMap, mType, and TEXT.

Here is the call graph for this function:



7.23.5.4 OutputNode::Pointer ParCompMark::OutputNode::createChildNode (const std::string & *text*) [virtual]

Create textual child node.

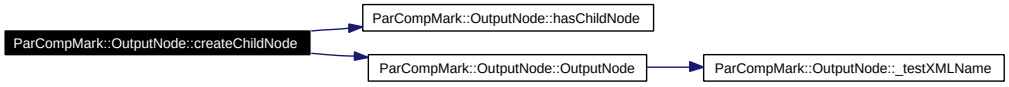
Parameters:
← ***text*** Text data.

Returns:
Created child node

Definition at line 156 of file PCMOutputNode.cpp.

References Assert, hasChildNode(), mChildren, mChildrenMap, mType, OutputNode(), TEXT, and TEXTNODENAME.

Here is the call graph for this function:



7.23.5.5 OutputNode::Pointer ParCompMark::OutputNode::createChildNode (const std::string & *name*, const NodeType & *type*) [virtual]

Create nontextual child node.

Parameters:

← ***name*** **Name**(p. 131) of the child node.

← ***type*** Type of the child node.

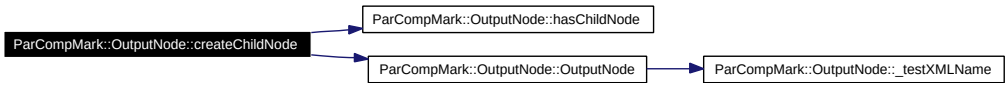
Returns:

Created child node

Definition at line 141 of file PCMOutputNode.cpp.

References Assert, hasChildNode(), mChildren, mChildrenMap, mType, OutputNode(), and TEXT.

Here is the call graph for this function:



7.23.5.6 const std::string & ParCompMark::OutputNode::getAttribute (const std::string & *attribute*) const [virtual]

Get an attribute by name.

Parameters:

← ***attribute*** **Name**(p. 131) of the attribute.

Returns:

Attribute value.

Definition at line 192 of file PCMOutputNode.cpp.

References Assert, mAttributes, mType, and TEXT.

7.23.5.7 OutputNode::Pointer ParCompMark::OutputNode::getChildNode (const std::string & *name*) [virtual]

Get child node by name.

Parameters:

← ***name*** **Name**(p. 131) of the child node.

Returns:

Found child node

Definition at line 128 of file PCMOutputNode.cpp.

References Assert, mChildrenMap, mType, and TEXT.

7.23.5.8 const std::string & ParCompMark::OutputNode::getEXTNODENAME () [inline, static]

Remarks:
Getter of TEXTNODENAME.

Returns value of TEXTNODENAME.

Returns:
The value of TEXTNODENAME

Definition at line 382 of file PCMOutputNode.h.

7.23.5.9 const std::string & ParCompMark::OutputNode::getText () const [inline]

Remarks:
Getter of mText.

Returns value of mText.

Returns:
The value of mText

Definition at line 396 of file PCMOutputNode.h.

7.23.5.10 const OutputNode::NodeType & ParCompMark::OutputNode::getType () const [inline]

Remarks:
Getter of mType.

Returns value of mType.

Returns:
The value of mType

Definition at line 389 of file PCMOutputNode.h.

7.23.5.11 bool ParCompMark::OutputNode::hasAttribute (const std::string & *attribute*) const [virtual]

Return true if the node has attribute with the specified name.

Parameters:
← ***attribute*** **Name**(p. 131) of the attribute.

Returns:
The node has the specified attribute.

Definition at line 184 of file PCMOutputNode.cpp.

References Assert, mAttributes, mType, and TEXT.

7.23.5.12 bool ParCompMark::OutputNode::hasChildNode (const std::string & *name*) [virtual]

Return true if the node has a child with the specified name.

Parameters:
← ***name*** **Name**(p. 131) of the child node.

Returns:
The node has a child with the specified name

Definition at line 120 of file PCMOutputNode.cpp.

References Assert, mChildrenMap, mType, and TEXT.

Referenced by addChildNode(), and createChildNode().

7.23.5.13 std::string ParCompMark::OutputNode::serialize2XML () [virtual]

Serialize the node to XML (this is a recursive method, calls the same methods of the children too). Use the other version of this method with parameter std::ostringstream if it is possible for memory saving.

Returns:
Serialized node.

Definition at line 271 of file PCMOutputNode.cpp.

7.23.5.14 std::ostringstream & ParCompMark::OutputNode::serialize2XML (std::ostringstream & *osstr*) [virtual]

Serialize the node to XML (this is a recursive method, calls the same methods of the children too).

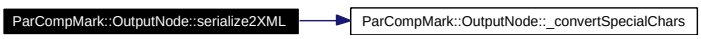
Parameters:
→ ***osstr*** Output string stream.

Returns:
Serialized node (added to osstr).

Definition at line 215 of file PCMOutputNode.cpp.

References _convertSpecialChars(), DEFINITION, INFORMATION, mAttributes, mChildren, ParCompMark::Name::mName, mText, mType, REFERENCE, STATISTICS, and TEXT.

Here is the call graph for this function:



7.23.5.15 void ParCompMark::OutputNode::setAttribute (const std::string & *attribute*, const std::string & *value*) [virtual]

Set attribute, also create if does not exists, overwrite otherwise.

Parameters:
← ***attribute*** **Name**(p. 131) of the attribute.

← ***value*** Value of the attribute.

Definition at line 205 of file PCMOutputNode.cpp.

References _testXMLName(), Assert, mAttributes, mType, and TEXT.

Here is the call graph for this function:

7.23.5.16 void ParCompMark::OutputNode::setText (const std::string & *text*) [inline]

Remarks:
Setter of mText.

Sets value of mText.

Parameters:
← ***text*** The value of mText

Definition at line 403 of file PCMOutputNode.h.

7.23.6 Member Data Documentation

7.23.6.1 AttributeMap ParCompMark::OutputNode::mAttributes [protected]

Attributes of the node.

Remarks:
This is own attribute of this class.

Definition at line 156 of file PCMOutputNode.h.

Referenced by getAttribute(), hasAttribute(), serialize2XML(), and setAttribute().

7.23.6.2 ChildNodeList ParCompMark::OutputNode::mChildren [protected]

Child nodes.

Remarks:
This is own attribute of this class.

Definition at line 163 of file PCMOutputNode.h.

Referenced by addChildNode(), createChildNode(), and serialize2XML().

7.23.6.3 ChildNodeMap ParCompMark::OutputNode::mChildrenMap [protected]

Map of child nodes (works in collaboration with ChildNodeList, and easies the element test in the list).

Remarks:
This is own attribute of this class.

Definition at line 170 of file PCMOutputNode.h.

Referenced by addChildNode(), createChildNode(), getChildNode(), and hasChildNode().

7.23.6.4 std::string ParCompMark::OutputNode::mText [protected]

Text data.

Remarks:
This is own attribute of this class.

Definition at line 177 of file PCMOutputNode.h.

Referenced by serialize2XML(), ParCompMarkTest::TestOutputNode::test_constructor_cstd__string(), and ParCompMarkTest::TestOutputNode::test_constructor_cstd__string_cNodeType().

7.23.6.5 NodeType ParCompMark::OutputNode::mType [protected]

Type of the node.

Remarks:
This is own attribute of this class.

Definition at line 149 of file PCMOutputNode.h.

Referenced by addChildNode(), createChildNode(), getAttribute(), getChildNode(), hasAttribute(), hasChildNode(), serialize2XML(), setAttribute(), ParCompMarkTest::TestOutputNode::test_constructor_cstd__string(), and ParCompMarkTest::TestOutputNode::test_constructor_cstd__string_cNodeType().

7.23.6.6 const std::string ParCompMark::OutputNode::TEXTNODENAME = "#text" [static, protected]

String constant for node name for text data nodes.

Remarks:
This is own attribute of this class.

Definition at line 136 of file PCMOutputNode.h.

Referenced by createChildNode().

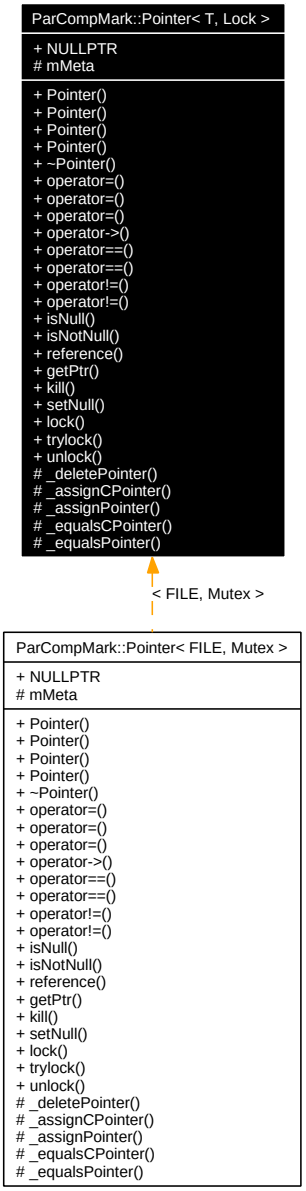
The documentation for this class was generated from the following files:

- **PCMOutputNode.h**
- **PCMOutputNode.cpp**

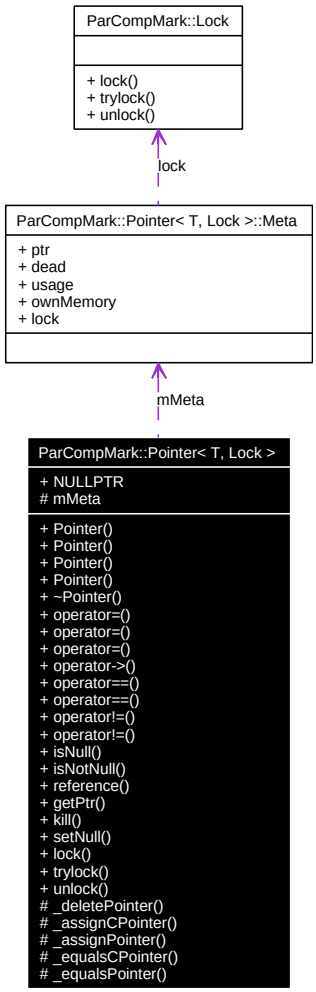
7.24 ParCompMark::Pointer< T, Lock > Class Template Reference

`#include <PCMPointer.h>`

Inheritance diagram for ParCompMark::Pointer< T, Lock >:



Collaboration diagram for ParCompMark::Pointer< T, Lock >:



7.24.1 Detailed Description

template<typename T, class Lock> class ParCompMark::Pointer< T, Lock >

Template smart pointer class.

Remarks:

This class provides: exception safe, garbage colletion, thread safeness, and more efficiency

Definition at line 56 of file PCMPointer.h.

Public Member Functions

- **Pointer** ()
- **Pointer** (const T *pointer, const bool &takeOwnership=true)
- **Pointer** (**Pointer**< T, **Lock** > &pointer)
- **Pointer** (const **Pointer**< T, **Lock** > &pointer)

- virtual ~**Pointer** ()
- virtual const **Pointer**< T, **Lock** > & **operator=** (const T *pointer)
- virtual const **Pointer**< T, **Lock** > & **operator=** (**Pointer**< T, **Lock** > &pointer)
- virtual const **Pointer**< T, **Lock** > & **operator=** (const **Pointer**< T, **Lock** > &pointer)
- virtual T *& **operator** → ()
- virtual bool **operator==** (const T *pointer)
- virtual bool **operator==** (**Pointer**< T, **Lock** > &pointer)
- virtual bool **operator!=** (const T *pointer)
- virtual bool **operator!=** (**Pointer**< T, **Lock** > &pointer)
- virtual bool **isNull** () const
- virtual bool **isNotNull** () const
- virtual void **reference** (const T *pointer)
- virtual T * **getPtr** ()
- virtual void **kill** (const bool &force=false)
- virtual void **setNull** (const bool &force=false)
- virtual void **lock** ()
- virtual bool **trylock** ()
- virtual void **unlock** ()

Static Public Attributes

- static const **Pointer**< T, **Lock** > * **NULLPTR**

Protected Member Functions

- virtual void **_deletePointer** (const bool &force=false)
- virtual void **_assignCPointer** (const T *pointer, const bool &takeOwnership=true)
- virtual void **_assignPointer** (**Pointer**< T, **Lock** > &pointer)
- virtual bool **_equalsCPointer** (const T *pointer) const
- virtual bool **_equalsPointer** (**Pointer**< T, **Lock** > &pointer) const

Protected Attributes

- **Meta** * **mMeta**

Classes

- struct **Meta**

7.24.2 Constructor & Destructor Documentation

7.24.2.1 template<typename T, class Lock> ParCompMark::Pointer< T, Lock >::Pointer () [inline]

Create a NULL pointer.

Definition at line 346 of file PCMPointer.h.

7.24.2.2 template<typename T, class Lock> ParCompMark::Pointer< T, Lock >::Pointer (const T * *pointer*, const bool & *takeOwnership* = true) [inline, explicit]

Create smart pointer from a C style pointer.

Parameters:

← ***pointer*** C style pointer

← ***takeOwnership*** Takes ownership of the assigned object. It will be deleted when smart pointer dies.

Definition at line 359 of file PCMPointer.h.

7.24.2.3 template<typename T, class Lock> ParCompMark::Pointer< T, Lock >::Pointer (Pointer< T, Lock > & *pointer*) [inline]

Create smart pointer from an another smart pointer. (Copy constructor)

Parameters:

→ ***pointer*** Another smart pointer to assign.

Definition at line 369 of file PCMPointer.h.

7.24.2.4 template<typename T, class Lock> ParCompMark::Pointer< T, Lock >::Pointer (const Pointer< T, Lock > & *pointer*) [inline]

Create smart pointer from an another const smart pointer. (Constantant copy constructor)

Parameters:

← ***pointer*** Another smart pointer to assign.

Definition at line 378 of file PCMPointer.h.

7.24.2.5 template<typename T, class Lock> ParCompMark::Pointer< T, Lock >::~Pointer () [virtual]

The destructor. This class has virtual destructor.

Definition at line 388 of file PCMPointer.h.

7.24.3 Member Function Documentation

7.24.3.1 template<typename T, class Lock> void ParCompMark::Pointer< T, Lock >::_assignCPointer (const T * *pointer*, const bool & *takeOwnership* = true) [inline, protected, virtual]

Assign a C style pointer.

Parameters:

← ***pointer*** C style pointer

← ***takeOwnership*** Takes ownership of the assigned object. It will be deleted when smart pointer dies.

Definition at line 629 of file PCMPointer.h.

7.24.3.2 template<typename T, class Lock> void ParCompMark::Pointer< T, Lock >::_assignPointer (Pointer< T, Lock > & *pointer*) [inline, protected, virtual]

Assign another smart pointer.

Parameters:
→ ***pointer*** Another smart pointer to assign.

Definition at line 642 of file PCMPointer.h.

7.24.3.3 template<typename T, class Lock> void ParCompMark::Pointer< T, Lock >::_deletePointer (const bool & *force* = false) [inline, protected, virtual]

Delete the pointer. Decrease reference counter, relese lock, free memory if needed.

Parameters:
← ***force*** If false, the object will only be deallocated when one it has one reference.

Definition at line 603 of file PCMPointer.h.

7.24.3.4 template<typename T, class Lock> bool ParCompMark::Pointer< T, Lock >::_equalsCPointer (const T * *pointer*) const [inline, protected, virtual]

Test indentity.

Parameters:
← ***pointer*** C style pointer

Returns:
True if the data in parameter is identical to the smart pointer.

Definition at line 653 of file PCMPointer.h.

7.24.3.5 template<typename T, class Lock> bool ParCompMark::Pointer< T, Lock >::_equalsPointer (Pointer< T, Lock > & *pointer*) const [inline, protected, virtual]

Test indentity.

Parameters:
→ ***pointer*** Another smart pointer to assign.

Returns:
True if the data in parameter is identical to the smart pointer.

Definition at line 660 of file PCMPointer.h.

7.24.3.6 template<typename T, class Lock> T * ParCompMark::Pointer< T, Lock >::getPtr () [inline, virtual]

Return a C style pointer to the referenced object.

Returns:
Pointer(p. 170) to the referenced object.

Definition at line 550 of file PCMPointer.h.

Referenced by ParCompMark::OutputNode::addChildNode(), ParCompMark::Logger::init(), and ParCompMark::Logger::~Logger().

7.24.3.7 template<typename T, class Lock> bool ParCompMark::Pointer< T, Lock >::isNotNull () const [inline, virtual]

Return true if the pointer does reference something.

Returns:
True if the pointer does reference something.

Definition at line 531 of file PCMPointer.h.

Referenced by ParCompMark::Process::initProcess().

7.24.3.8 template<typename T, class Lock> bool ParCompMark::Pointer< T, Lock >::isNull () const [inline, virtual]

Return true if the pointer does not reference anything.

Returns:
True if the pointer does not reference anything.

Definition at line 524 of file PCMPointer.h.

7.24.3.9 template<typename T, class Lock> void ParCompMark::Pointer< T, Lock >::kill (const bool & *force* = false) [inline, virtual]

Force deallocating referenced object.

Parameters:
← ***force*** If false, the object will only be deallocated when one it has one reference.

Definition at line 558 of file PCMPointer.h.

7.24.3.10 template<typename T, class Lock> void ParCompMark::Pointer< T, Lock >::lock () [inline, virtual]

Lock(p. 116) the referenced object.

Definition at line 580 of file PCMPointer.h.

Referenced by ParCompMark::Network::buildCluster(), ParCompMark::GLXRenderWindow::createWindow(), ParCompMark::GLXGLContext::setCurrent(), ParCompMark::HandleClient::task(), and ParCompMark::Logger::~Logger().

7.24.3.11 template<typename T, class Lock> bool ParCompMark::Pointer< T, Lock >::operator!= (Pointer< T, Lock > & *pointer*) [inline, virtual]

Equality test operator.

Parameters:
→ ***pointer*** Another smart pointer to test.

Returns:
True if the referenced objects are not identical.

Definition at line 512 of file PCMPointer.h.

7.24.3.12 template<typename T, class Lock> bool ParCompMark::Pointer< T, Lock >::operator!= (const T * *pointer*) [inline, virtual]

Equality test operator on a C style pointer.

Parameters:
← ***pointer*** C style pointer

Returns:
True if the referenced object is not identical to the parameter.

Definition at line 504 of file PCMPointer.h.

7.24.3.13 template<typename T, class Lock> T *& ParCompMark::Pointer< T, Lock >::operator → () [inline, virtual]

Get referenced object.

Returns:
Referenced object.

Definition at line 447 of file PCMPointer.h.

7.24.3.14 template<typename T, class Lock> const Pointer< T, Lock > & ParCompMark::Pointer< T, Lock >::operator= (const Pointer< T, Lock > & *pointer*) [inline, virtual]

Assign another smart pointer (const version).

Parameters:
← ***pointer*** Another smart pointer to assign.

Returns:
Self reference.

Definition at line 431 of file PCMPointer.h.

7.24.3.15 template<typename T, class Lock> const Pointer< T, Lock > & ParCompMark::Pointer< T, Lock >::operator= (Pointer< T, Lock > & *pointer*) [inline, virtual]

Assign another smart pointer.

Parameters:
→ ***pointer*** Another smart pointer to assign.

Returns:
Self reference.

Definition at line 414 of file PCMPointer.h.

7.24.3.16 template<typename T, class Lock> const Pointer< T, Lock > & ParCompMark::Pointer< T, Lock >::operator= (const T * *pointer*) [inline, virtual]

Assign a C style pointer.

Parameters:
← ***pointer*** C style pointer

Returns:
Self reference.

Definition at line 400 of file PCMPointer.h.

7.24.3.17 template<typename T, class Lock> bool ParCompMark::Pointer< T, Lock >::operator== (Pointer< T, Lock > & *pointer*) [inline, virtual]

Equality test operator.

Parameters:
→ ***pointer*** Another smart pointer to test.

Returns:
True if the referenced objects are identical.

Definition at line 496 of file PCMPointer.h.

7.24.3.18 template<typename T, class Lock> bool ParCompMark::Pointer< T, Lock >::operator== (const T * *pointer*) [inline, virtual]

Equality test operator on a C style pointer.

Parameters:
← ***pointer*** C style pointer

Returns:
True if the referenced object is identical to the parameter.

Definition at line 488 of file PCMPointer.h.

7.24.3.19 template<typename T, class Lock> void ParCompMark::Pointer< T, Lock >::reference (const T * *pointer*) [inline, virtual]

Take reference from a C style pointer (does not delete referenced object when smart pointer dies).

Parameters:
← ***pointer*** C style pointer

Definition at line 539 of file PCMPointer.h.

7.24.3.20 template<typename T, class Lock> void ParCompMark::Pointer< T, Lock >::setNull (const bool & *force* = false) [inline, virtual]

Set the referenced object to null without trying to deallocate the actual object.

Remarks:

The typical usage of this method the external deallocating a pointer, like fclose.

Parameters:
← ***force*** If false, the reference will only be set to null when one it has one reference.

Definition at line 570 of file PCMPointer.h.

Referenced by ParCompMark::Logger::~Logger().

7.24.3.21 template<typename T, class Lock> bool ParCompMark::Pointer< T, Lock >::trylock () [inline, virtual]

Try locking referenced object.

Returns:
True if the locking was successful.

Definition at line 587 of file PCMPointer.h.

7.24.3.22 template<typename T, class Lock> void ParCompMark::Pointer< T, Lock >::unlock () [inline, virtual]

Unlock the referenced object.

Definition at line 595 of file PCMPointer.h.

Referenced by ParCompMark::Network::buildCluster(), ParCompMark::GLXGLContext::setCurrent(), ParCompMark::HandleClient::task(), and ParCompMark::Logger::~Logger().

7.24.4 Member Data Documentation

7.24.4.1 template<typename T, class Lock> Meta* ParCompMark::Pointer< T, Lock >::mMeta [protected]

Meta(p. 180) field for the referenced object.

Remarks:
This is own attribute of this class.

Definition at line 112 of file PCMPointer.h.

7.24.4.2 template<typename T, class Lock> const Pointer< T, Lock > * ParCompMark::Pointer< T, Lock >::NULLPTR [static]

Initial value:

```
Pointer < T, DummyLock > ()
```

Null pointer constant.

Remarks:
This is own attribute of this class.

Definition at line 99 of file PCMPointer.h.

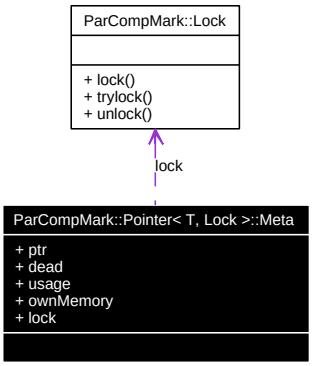
The documentation for this class was generated from the following files:

- **PCMPointer.h**
- **PCMPointer.cpp**

7.25 ParCompMark::Pointer< T, Lock >::Meta Struct Reference

`#include <PCMPointer.h>`

Collaboration diagram for ParCompMark::Pointer< T, Lock >::Meta:



7.25.1 Detailed Description

template<typename T, class Lock> struct ParCompMark::Pointer< T, Lock >::Meta

Meta(p. 180) field type for the referenced object.

Definition at line 73 of file PCMPointer.h.

Public Attributes

- T * **ptr**
- bool **dead**
- **u32 usage**
- bool **ownMemory**
- **Lock lock**

7.25.2 Member Data Documentation

7.25.2.1 template<typename T, class Lock> bool ParCompMark::Pointer< T, Lock >::Meta::dead

Indicates that the object is deleted

Definition at line 79 of file PCMPointer.h.

7.25.2.2 template<typename T, class Lock> Lock ParCompMark::Pointer< T, Lock >::Meta::lock

Lock(p. 116) for the object

Definition at line 84 of file PCMPointer.h.

7.25.2.3 template<typename T, class Lock> bool ParCompMark::Pointer< T, Lock >::Meta::ownMemory

Indicates that the referenced memory should deallocated by the smart pointer

Definition at line 83 of file PCMPointer.h.

7.25.2.4 template<typename T, class Lock> T* ParCompMark::Pointer< T, Lock >::Meta::ptr

Pointer(p. 170) to the referenced object

Definition at line 77 of file PCMPointer.h.

7.25.2.5 template<typename T, class Lock> u32 ParCompMark::Pointer< T, Lock >::Meta::usage

Usage counter

Definition at line 81 of file PCMPointer.h.

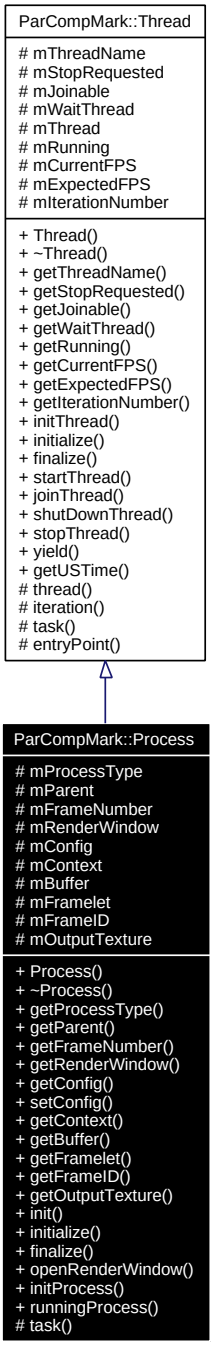
The documentation for this struct was generated from the following file:

- **PCMPointer.h**

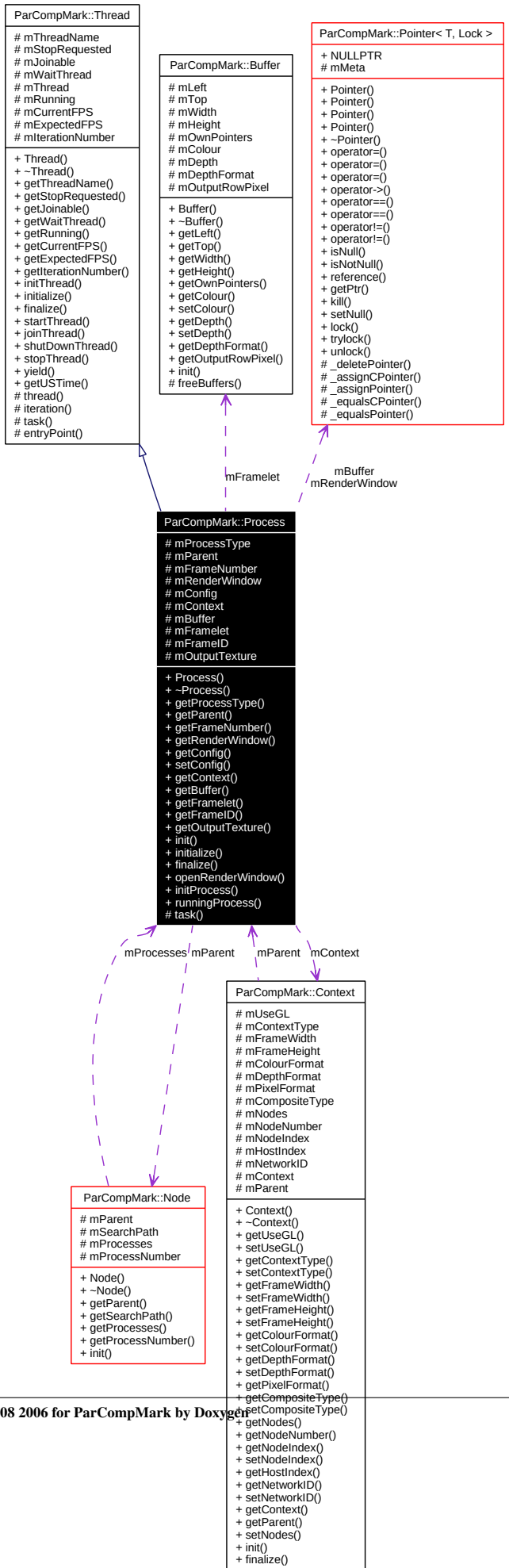
7.26 ParCompMark::Process Class Reference

`#include <PCMProcess.h>`

Inheritance diagram for ParCompMark::Process:



Collaboration diagram for ParCompMark::Process:


ParCompMark::Thread
mThreadName
mStopRequested
mJoinable
mWaitThread
mThread
mRunning
mCurrentFPS
mExpectedFPS
mIterationNumber
+ Thread()
+ ~Thread()
+ getThreadName()
+ getStopRequested()
+ getJoinable()
+ getWaitThread()
+ getRunning()
+ getCurrentFPS()
+ getExpectedFPS()
+ getIterationNumber()
+ initThread()
+ initialize()
+ finalize()
+ startThread()
+ joinThread()
+ shutDownThread()
+ stopThread()
+ yield()
+ getUSTime()
# thread()
# iteration()
# task()
# entryPoint()
ParCompMark::Buffer
mLeft
mTop
mWidth
mHeight
mOwnPointers
mColour
mDepth
mDepthFormat
mOutputRowPixel
+ Buffer()
+ ~Buffer()
+ getLeft()
+ getTop()
+ getWidth()
+ getHeight()
+ getOwnPointers()
+ getColour()
+ setColour()
+ getDepth()
+ setDepth()
+ getDepthFormat()
+ getOutputRowPixel()
+ init()
# freeBuffers()
ParCompMark::Pointer< T, Lock >
+ NULLPTR
# mMeta
+ Pointer()
+ Pointer()
+ Pointer()
+ Pointer()
+ ~Pointer()
+ operator=()
+ operator=()
+ operator=()
+ operator->()
+ operator==()
+ operator==()
+ operator!=()
+ operator!=()
+ isNull()
+ isNotNull()
+ reference()
+ getPtr()
+ kill()
+ setNull()
+ lock()
+ trylock()
+ unlock()
# _deletePointer()
# _assignCPointer()
# _assignPointer()
# _equalsCPointer()
# _equalsPointer()
mFramelet
mBuffer
mRenderWindow
ParCompMark::Process
mProcessType
mParent
mFrameNumber
mRenderWindow
mConfig
mContext
mBuffer
mFramelet
mFrameID
mOutputTexture
+ Process()
+ ~Process()
+ getProcessType()
+ getParent()
+ getFrameNumber()
+ getRenderWindow()
+ getConfig()
+ setConfig()
+ getContext()
+ getBuffer()
+ getFramelet()
+ getFrameID()
+ getOutputTexture()
+ init()
+ initialize()
+ finalize()
+ openRenderWindow()
+ initProcess()
+ runningProcess()
# task()
mProcesses
mParent
mParent
mContext
ParCompMark::Node
mParent
mSearchPath
mProcesses
mProcessNumber
+ Node()
+ ~Node()
+ getParent()
+ getSearchPath()
+ getProcesses()
+ getProcessNumber()
+ init()
ParCompMark::Context
mUseGL
mContextType
mFrameWidth
mFrameHeight
mColourFormat
mDepthFormat
mPixelFormat
mCompositeType
mNodes
mNodeNumber
mNodeIndex
mHostIndex
mNetworkID
mContext
mParent
+ Context()
+ ~Context()
+ getUseGL()
+ setUseGL()
+ getContextType()
+ setContextType()
+ getFrameWidth()
+ setFrameWidth()
+ getFrameHeight()
+ setFrameHeight()
+ getColourFormat()
+ setColourFormat()
+ getDepthFormat()
+ setDepthFormat()
+ getPixelFormat()
+ getCompositeType()
+ setCompositeType()
+ getNodes()
+ getNodeNumber()
+ getNodeIndex()
+ setNodeIndex()
+ getHostIndex()
+ getNetworkID()
+ setNetworkID()
+ getContext()
+ getParent()
+ setNodes()
+ init()
+ finalize()

7.26.1 Detailed Description

Entity that composite or render something.

Definition at line 65 of file PCMProcess.h.

Public Types

- enum **ProcessType** { **COMPOSITE**, **RENDER** }

Public Member Functions

- **Process** (const std::string &name, **Node** *parent)
- virtual ~**Process** ()
- const **Process::ProcessType** & **getProcessType** () const
- **Node** * **getParent** () const
- const **u32** & **getFrameNumber** () const
- **GLXRenderWindow::Pointer** & **getRenderWindow** ()
- const std::string & **getConfig** () const
- void **setConfig** (const std::string &config)
- **Context** * **getContext** () const
- const **Buffer::Pointer** & **getBuffer** () const
- **Buffer** * **getFramelet** () const
- const PCid & **getFrameID** () const
- const GLuint & **getOutputTexture** () const
- virtual void **init** ()
- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual void **openRenderWindow** (const std::string caption="PCM Framework", const std::string &displayName="", const bool &fullScreen=true, const **u32** &colourDepth=0, const **s32** &left=**GLXRenderWindow::CENTERED**, const **s32** &top=**GLXRenderWindow::CENTERED**, const **u32** &width=**GLXRenderWindow::MAXIMALSIZE**, const **u32** &height=**GLXRenderWindow::MAXIMALSIZE**, const **u32** &fsaaSamples=0)
- virtual void **initProcess** ()
- virtual void **runningProcess** (const **u32** &frameNumber, const **Real** &time)

Protected Member Functions

- virtual void **task** ()

Protected Attributes

- **ProcessType mProcessType**
- **Node** * **mParent**
- **u32 mFrameNumber**
- **GLXRenderWindow::Pointer mRenderWindow**
- std::string **mConfig**
- **Context** * **mContext**
- **Buffer::Pointer mBuffer**

- **Buffer** ∗ **mFramelet**
- PCid **mFrameID**
- GLuint **mOutputTexture**

7.26.2 Member Enumeration Documentation

7.26.2.1 enum ParCompMark::Process::ProcessType

The control type of PC context (Local, Global).

Enumerator:

COMPOSITE **Process**(p. 182) composite and provides frame and requires framelets.

RENDER **Process**(p. 182) provides framelets.

Definition at line 82 of file PCMProcess.h.

7.26.3 Constructor & Destructor Documentation

7.26.3.1 ParCompMark::Process::Process (const std::string & *name*, Node ∗ *parent*)

Creates a process with a specified name.

Parameters:

← ***name*** **Name**(p. 131) of the host.

← ***parent*** Parent node

Definition at line 42 of file PCMProcess.cpp.

References Assert, mConfig, mContext, mFrameID, mFramelet, mFrameNumber, mProcessType, and RENDER.

7.26.3.2 ParCompMark::Process::∼Process () [virtual]

The destructor. This class has virtual destructor.

Definition at line 75 of file PCMProcess.cpp.

References mContext, and mFramelet.

7.26.4 Member Function Documentation

7.26.4.1 void ParCompMark::Process::finalize () [virtual]

Some finitiate step as thread.

Reimplemented from **ParCompMark::Thread** (p. 267).

Definition at line 232 of file PCMProcess.cpp.

References ParCompMark::Context::finalize(), and mContext.

Here is the call graph for this function:

7.26.4.2 const Buffer::Pointer & ParCompMark::Process::getBuffer () const [inline]

Remarks:
Getter of mBuffer.

Returns value of mBuffer.

Returns:
The value of mBuffer

Definition at line 462 of file PCMProcess.h.

7.26.4.3 const std::string & ParCompMark::Process::getConfig () const [inline]

Remarks:
Getter of mConfig.

Returns value of mConfig.

Returns:
The value of mConfig

Definition at line 441 of file PCMProcess.h.

7.26.4.4 Context * ParCompMark::Process::getContext () const [inline]

Remarks:
Getter of mContext.

Returns value of mContext.

Returns:
The value of mContext

Definition at line 455 of file PCMProcess.h.

7.26.4.5 const PCid & ParCompMark::Process::getFrameID () const [inline]

Remarks:
Getter of mFrameID.

Returns value of mFrameID.

Returns:
The value of mFrameID

Definition at line 476 of file PCMProcess.h.

7.26.4.6 Buffer * ParCompMark::Process::getFramelet () const [inline]

Remarks:
Getter of mFramelet.

Returns value of mFramelet.

Returns:
The value of mFramelet

Definition at line 469 of file PCMProcess.h.

7.26.4.7 const u32 & ParCompMark::Process::getFrameNumber () const [inline]

Remarks:
Getter of mFrameNumber.

Returns value of mFrameNumber.

Returns:
The value of mFrameNumber

Definition at line 427 of file PCMProcess.h.

7.26.4.8 const GLuint & ParCompMark::Process::getOutputTexture () const [inline]

Remarks:
Getter of mOutputTexture.

Returns value of mOutputTexture.

Returns:
The value of mOutputTexture

Definition at line 483 of file PCMProcess.h.

7.26.4.9 Node * ParCompMark::Process::getParent () const [inline]

Remarks:
Getter of mParent.

Returns value of mParent.

Returns:
The value of mParent

Definition at line 420 of file PCMProcess.h.

Referenced by openRenderWindow().

7.26.4.10 const Process::ProcessType & ParCompMark::Process::getProcessType () const [inline]

Remarks:
Getter of mProcessType.

Returns value of mProcessType.

Returns:
The value of mProcessType

Definition at line 413 of file PCMProcess.h.

7.26.4.11 GLXRenderWindow::Pointer & ParCompMark::Process::getRenderWindow () [inline]

Remarks:
Getter of mRenderWindow.

Returns value of mRenderWindow.

Returns:
The value of mRenderWindow

Definition at line 434 of file PCMProcess.h.

Referenced by ParCompMark::Context::init().

7.26.4.12 void ParCompMark::Process::init () [virtual]

Init the process.

Definition at line 90 of file PCMProcess.cpp.

References COMPOSITE, mConfig, and mProcessType.

Referenced by initialize().

7.26.4.13 void ParCompMark::Process::initialize () [virtual]

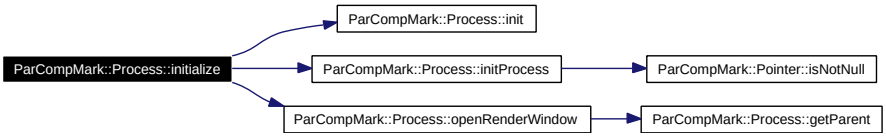
Some init step as thread.

Reimplemented from **ParCompMark::Thread** (p. 269).

Definition at line 217 of file PCMProcess.cpp.

References init(), initProcess(), and openRenderWindow().

Here is the call graph for this function:



7.26.4.14 void ParCompMark::Process::initProcess () [virtual]

This is an initializing method. Called at the start of the working process. Starts a Squirrel VM and executes the initialization code.

Definition at line 262 of file PCMProcess.cpp.

References COMPOSITE, ParCompMark::Pointer< T, Lock >::isNotNull(), mOutputTexture, and mRenderWindow.

Referenced by initialize().

Here is the call graph for this function:

7.26.4.15 void ParCompMark::Process::openRenderWindow (const std::string *caption* = "PCM Framework", const std::string & *displayName* = "", const bool & *fullScreen* = true, const u32 & *colourDepth* = 0, const s32 & *left* = GLXRenderWindow::CENTERED, const s32 & *top* = GLXRenderWindow::CENTERED, const u32 & *width* = GLXRenderWindow::MAXIMALSIZE, const u32 & *height* = GLXRenderWindow::MAXIMALSIZE, const u32 & *fsaaSamples* = 0) [virtual]

Open GLX render window for rendering.

Parameters:

← ***caption*** Window caption

← ***displayName*** X display name

← ***fullScreen*** The window appears in full screen mode

← ***colourDepth*** Colour depth of the window

← ***left*** Horizontal position

← ***top*** Vertical position

← ***width*** Horizontal size

← ***height*** Vertical size

← ***fsaaSamples*** Number of fullscreen antialasing samples (Do not use FSAA samples other than 0 now with nVidia cards!)

Definition at line 243 of file PCMProcess.cpp.

References getParent(), and mRenderWindow.

Referenced by initialize().

Here is the call graph for this function:

7.26.4.16 void ParCompMark::Process::runningProcess (const u32 & *frameNumber*, const Real & *time*) [virtual]

This method achieves the "real functionality" of the process. Called at every frame. Starts a Squirrel VM and executes the initialization code.

Parameters:

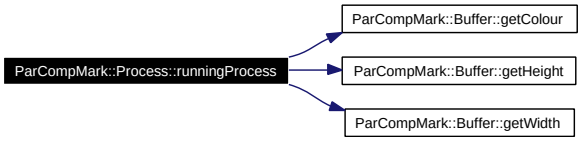
← ***frameNumber*** Number of frame to produce

← ***time*** Elapsed time since starting the benchmark

Definition at line 293 of file PCMProcess.cpp.

References COMPOSITE, ParCompMark::Buffer::getColour(), ParCompMark::Buffer::getHeight(), ParCompMark::Buffer::getWidth(), mBuffer, mFramelet, mFrameNumber, mOutputTexture, mRenderWindow, and RENDER.

Here is the call graph for this function:



7.26.4.17 void ParCompMark::Process::setConfig (const std::string & *config*) [inline]

Remarks:

Setter of mConfig.

Sets value of mConfig.

Parameters:

← ***config*** The value of mConfig

Definition at line 448 of file PCMProcess.h.

7.26.4.18 void ParCompMark::Process::task () [protected, virtual]

What we do during rendering/compositing.

Reimplemented from **ParCompMark::Thread** (p. 271).

Definition at line 353 of file PCMProcess.cpp.

References COMPOSITE, Except, ParCompMark::Context::getContext(), mContext, and mFrameID.

Here is the call graph for this function:

7.26.5 Member Data Documentation

7.26.5.1 Buffer::Pointer ParCompMark::Process::mBuffer [protected]

Memory buffer for output (if has) This is lockable.

Remarks:
This is own attribute of this class.

Definition at line 145 of file PCMProcess.h.

Referenced by runningProcess().

7.26.5.2 std::string ParCompMark::Process::mConfig [protected]

Config string.

Remarks:
This is own attribute of this class.

Definition at line 131 of file PCMProcess.h.

Referenced by init(), and Process().

7.26.5.3 Context* ParCompMark::Process::mContext [protected]

Context(p. 53) description.

Remarks:
This is own attribute of this class.

Definition at line 138 of file PCMProcess.h.

Referenced by finalize(), Process(), task(), and ~Process().

7.26.5.4 PCid ParCompMark::Process::mFrameID [protected]

The current frame id. It sets by PC.

Remarks:
This is own attribute of this class.

Definition at line 159 of file PCMProcess.h.

Referenced by Process(), and task().

7.26.5.5 Buffer* ParCompMark::Process::mFramelet [protected]

Memory buffers for framelets (if has).

Remarks:
This is own attribute of this class.

Definition at line 152 of file PCMProcess.h.

Referenced by Process(), runningProcess(), and ∼Process().

7.26.5.6 u32 ParCompMark::Process::mFrameNumber [protected]

Number of actual frame.

Remarks:
This is own attribute of this class.

Definition at line 117 of file PCMProcess.h.

Referenced by Process(), and runningProcess().

7.26.5.7 GLuint ParCompMark::Process::mOutputTexture [protected]

OpenGL texture for displaying output.

Remarks:
This is own attribute of this class.

Definition at line 166 of file PCMProcess.h.

Referenced by initProcess(), and runningProcess().

7.26.5.8 Node∗ ParCompMark::Process::mParent [protected]

Parent node of the process. (Parent reference is standard pointer to avoid circular reference)

Remarks:
This attribute references an attribute.

Definition at line 110 of file PCMProcess.h.

7.26.5.9 ProcessType ParCompMark::Process::mProcessType [protected]

Composite or render process.

Remarks:
This is own attribute of this class.

Definition at line 103 of file PCMProcess.h.

Referenced by init(), and Process().

7.26.5.10 GLXRenderWindow::Pointer ParCompMark::Process::mRenderWindow [protected]

Renderwindow of this process. Each process can have zero or one renderwindow.

Remarks:
This is own attribute of this class.

Definition at line 124 of file PCMProcess.h.

Referenced by initProcess(), openRenderWindow(), and runningProcess().

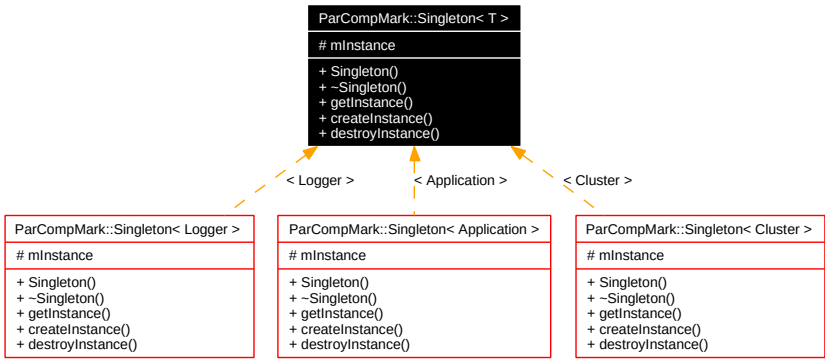
The documentation for this class was generated from the following files:

- **PCMProcess.h**
- **PCMProcess.cpp**

7.27 ParCompMark::Singleton< T > Class Template Reference

`#include <PCMSingleton.h>`

Inheritance diagram for ParCompMark::Singleton< T >:



7.27.1 Detailed Description

template<typename T> class ParCompMark::Singleton< T >

Template class for creating single-instance global classes.

Definition at line 46 of file PCMSingleton.h.

Public Member Functions

- **Singleton** ()
- virtual ~**Singleton** ()

Static Public Member Functions

- static T * **getInstance** ()
- static void **createInstance** ()
- static void **destroyInstance** ()

Static Protected Attributes

- static T * **mInstance**

7.27.2 Constructor & Destructor Documentation

7.27.2.1 template<typename T> ParCompMark::Singleton< T >::Singleton ()

Default constructor.

Definition at line 141 of file PCMSingleton.h.

7.27.2.2 template<typename T> ParCompMark::Singleton< T >::~Singleton () [virtual]

The destructor. This class has virtual destructor.

Definition at line 150 of file PCMSingleton.h.

7.27.3 Member Function Documentation

7.27.3.1 template<typename T> void ParCompMark::Singleton< T >::createInstance () [inline, static]

Create singleton instance.

Definition at line 170 of file PCMSingleton.h.

7.27.3.2 template<typename T> void ParCompMark::Singleton< T >::destroyInstance () [inline, static]

Destroy singleton instance.

Definition at line 177 of file PCMSingleton.h.

7.27.3.3 template<typename T> T * ParCompMark::Singleton< T >::getInstance () [inline, static]

Gets the instance of the singleton class. You have to construct singleton object (createInstance) before calling this method.

Returns:
Instance of the class

Definition at line 163 of file PCMSingleton.h.

7.27.4 Member Data Documentation

7.27.4.1 template<typename T> T * ParCompMark::Singleton< T >::mInstance [static, protected]

Class level instance.

Remarks:
This is own attribute of this class.

Definition at line 67 of file PCMSingleton.h.

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

- **PCMSingleton.h**
- **PCMSingleton.cpp**

7.28 ParCompMark::SqVM Class Reference

`#include <PCMSqVM.h>`

7.28.1 Detailed Description

Squirrel virtual machine.

Definition at line 57 of file PCMSqVM.h.

Static Protected Member Functions

- static void **printFunction** (::HSQUIRRELVM sqVM, const SQChar *s,...)

7.28.2 Member Function Documentation

7.28.2.1 void ParCompMark::SqVM::printFunction (::HSQUIRRELVM *sqVM*, const SQChar * *s*, ...) [static, protected]

The print function of the virtual machine. This function is used by the built-in function 'print()' to output text.

Parameters:

← ***sqVM*** Squirrel VM

← ***s*** Format string

← **...** Additional parameters

Definition at line 44 of file PCMSqVM.cpp.

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

- **PCMSqVM.h**
- **PCMSqVM.cpp**

7.29 ParCompMarkTest::TestApplication Class Reference

`#include <TestApplication.h>`

7.29.1 Detailed Description

Remarks:

Test suite for **ParCompMark::Application**(p. 21).

This is a `CppUnit` test suite for class **ParCompMark::Application**(p. 21) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestApplication.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor** ()
- void **test_destructor** ()
- void **test_parseCommandLine_cu32_cchar_pp** ()
- void **test_showHelp_cstd__string** ()
- void **test_showVersion_cstd__string** ()
- void **test_setCommanderOn_cstd__string** ()
- void **test_setGUIOn_cstd__string** ()
- void **test_setLowLevelOn_cstd__string** ()
- void **test_setCluster_cstd__string** ()
- void **test_setParameters_cstd__string** ()
- void **test_setInput_cstd__string** ()
- void **test_setOutput_cstd__string** ()
- void **test_terminateHandler** ()
- void **test_unexpectedHandler** ()
- void **test_segfaultHandler_int** ()
- void **test_setupHandlers** ()
- void **test_initalize** ()
- void **test_finalize** ()
- void **test_NetworkTest** ()

7.29.2 Member Function Documentation

7.29.2.1 void ParCompMarkTest::TestApplication::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestApplication.cpp.

7.29.2.2 void ParCompMarkTest::TestApplication::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestApplication.cpp.

7.29.2.3 void ParCompMarkTest::TestApplication::test_constructor () [protected]

Tests constructor Application::Application().

Definition at line 46 of file TestApplication.cpp.

7.29.2.4 void ParCompMarkTest::TestApplication::test_destructor () [protected]

Tests destructor Application::∼Application().

Definition at line 60 of file TestApplication.cpp.

7.29.2.5 void ParCompMarkTest::TestApplication::test_finalize () [protected]

Tests method Application::finalize().

Definition at line 289 of file TestApplication.cpp.

7.29.2.6 void ParCompMarkTest::TestApplication::test_initalize () [protected]

Tests method Application::initalize().

Definition at line 275 of file TestApplication.cpp.

7.29.2.7 void ParCompMarkTest::TestApplication::test_NetworkTest () [protected]

Tests method Application::NetworkTest().

Definition at line 303 of file TestApplication.cpp.

7.29.2.8 void ParCompMarkTest::TestApplication::test_parseCommandLine_cu32_cchar_pp () [protected]

Tests method Application::parseCommandLine(const u32 &argc, const char ∗∗ &argv).

Definition at line 75 of file TestApplication.cpp.

7.29.2.9 void ParCompMarkTest::TestApplication::test_segfaultHandler_int () [protected]

Tests method Application::segfaultHandler(int value).

Definition at line 243 of file TestApplication.cpp.

7.29.2.10 void ParCompMarkTest::TestApplication::test_setCluster_cstd__string ()
[protected]

Tests method Application::setCluster(const std::string &strarg).

Definition at line 159 of file TestApplication.cpp.

7.29.2.11 void ParCompMarkTest::TestApplication::test_setCommanderOn_cstd__string ()
[protected]

Tests method Application::setCommanderOn(const std::string &strarg).

Definition at line 117 of file TestApplication.cpp.

7.29.2.12 void ParCompMarkTest::TestApplication::test_setGUIOn_cstd__string ()
[protected]

Tests method Application::setGUIOn(const std::string &strarg).

Definition at line 131 of file TestApplication.cpp.

7.29.2.13 void ParCompMarkTest::TestApplication::test_setInput_cstd__string ()
[protected]

Tests method Application::setInput(const std::string &strarg).

Definition at line 187 of file TestApplication.cpp.

7.29.2.14 void ParCompMarkTest::TestApplication::test_setLowLevelOn_cstd__string ()
[protected]

Tests method Application::setLowLevelOn(const std::string &strarg).

Definition at line 145 of file TestApplication.cpp.

7.29.2.15 void ParCompMarkTest::TestApplication::test_setOutput_cstd__string ()
[protected]

Tests method Application::setOutput(const std::string &strarg).

Definition at line 201 of file TestApplication.cpp.

7.29.2.16 void ParCompMarkTest::TestApplication::test_setParameters_cstd__string ()
[protected]

Tests method Application::setParameters(const std::string &strarg).

Definition at line 173 of file TestApplication.cpp.

7.29.2.17 void ParCompMarkTest::TestApplication::test_setupHandlers () [protected]

Tests method Application::setupHandlers().

Definition at line 261 of file TestApplication.cpp.

7.29.2.18 void ParCompMarkTest::TestApplication::test_showHelp_cstd__string () [protected]

Tests method Application::showHelp(const std::string &strarg).

Definition at line 89 of file TestApplication.cpp.

7.29.2.19 void ParCompMarkTest::TestApplication::test_showVersion_cstd__string () [protected]

Tests method Application::showVersion(const std::string &strarg).

Definition at line 103 of file TestApplication.cpp.

7.29.2.20 void ParCompMarkTest::TestApplication::test_terminateHandler () [protected]

Tests method Application::terminateHandler().

Definition at line 215 of file TestApplication.cpp.

7.29.2.21 void ParCompMarkTest::TestApplication::test_unexpectedHandler () [protected]

Tests method Application::unexpectedHandler().

Definition at line 229 of file TestApplication.cpp.

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

- **TestApplication.h**
- **TestApplication.cpp**

7.30 ParCompMarkTest::TestBuffer Class Reference

`#include <TestBuffer.h>`

7.30.1 Detailed Description

Remarks:

Test suite for **ParCompMark::Buffer**(p. 39).

This is a `CppUnit` test suite for class **ParCompMark::Buffer**(p. 39) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestBuffer.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor** ()
- void **test_destructor** ()
- void **test_init_cint_cint_cint_cint_cint** ()
- void **test_freeBuffers** ()

7.30.2 Member Function Documentation

7.30.2.1 void ParCompMarkTest::TestBuffer::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestBuffer.cpp.

7.30.2.2 void ParCompMarkTest::TestBuffer::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestBuffer.cpp.

7.30.2.3 void ParCompMarkTest::TestBuffer::test_constructor () [protected]

Tests constructor Buffer::Buffer().

Definition at line 46 of file TestBuffer.cpp.

7.30.2.4 void ParCompMarkTest::TestBuffer::test_destructor () [protected]

Tests destructor Buffer::∼Buffer().

Definition at line 54 of file TestBuffer.cpp.

7.30.2.5 void ParCompMarkTest::TestBuffer::test_freeBuffers () [protected]

Tests method Buffer::freeBuffers().

Definition at line 80 of file TestBuffer.cpp.

7.30.2.6 void ParCompMarkTest::TestBuffer::test_init_cint_cint_cint_cint_cint () [protected]

Tests method Buffer::init(const int &left, const int &top, const int &width, const int &height, const int &depthFormat).

Definition at line 66 of file TestBuffer.cpp.

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

- **TestBuffer.h**
- **TestBuffer.cpp**

7.31 ParCompMarkTest::TestCluster Class Reference

`#include <TestCluster.h>`

7.31.1 Detailed Description

Remarks:

Test suite for **ParCompMark::Cluster**(p. 46).

This is a `CppUnit` test suite for class **ParCompMark::Cluster**(p. 46) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestCluster.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor** ()
- void **test_destructor** ()

7.31.2 Member Function Documentation

7.31.2.1 void ParCompMarkTest::TestCluster::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestCluster.cpp.

7.31.2.2 void ParCompMarkTest::TestCluster::tearDown ()

Cleans up after the test run.

Definition at line 40 of file TestCluster.cpp.

7.31.2.3 void ParCompMarkTest::TestCluster::test_constructor () [protected]

Tests constructor Cluster::Cluster().

Definition at line 50 of file TestCluster.cpp.

7.31.2.4 void ParCompMarkTest::TestCluster::test_destructor () [protected]

Tests destructor Cluster::∼Cluster().

Definition at line 61 of file TestCluster.cpp.

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

- **TestCluster.h**
- **TestCluster.cpp**

7.32 ParCompMarkTest::TestContainer Class Reference

`#include <TestContainer.h>`

7.32.1 Detailed Description

Remarks:

Test suite for **ParCompMark::Container**(p. 49).

This is a `CppUnit` test suite for class **ParCompMark::Container**(p. 49) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestContainer.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor** ()
- void **test_destructor** ()
- void **test_add_std__string_ElementPointer** ()
- void **test_get_cstd__string** ()
- void **test_has_cstd__string** ()
- void **test_remove_cstd__string** ()
- void **test_getSize** ()
- void **test_isEmpty** ()

7.32.2 Member Function Documentation

7.32.2.1 void ParCompMarkTest::TestContainer::setUp ()

Sets up context before running a test.

Definition at line 42 of file TestContainer.cpp.

7.32.2.2 void ParCompMarkTest::TestContainer::tearDown ()

Cleans up after the test run.

Definition at line 51 of file TestContainer.cpp.

7.32.2.3 void ParCompMarkTest::TestContainer::test_add_std__string_ElementPointer () [protected]

Tests method Container::add(std::string name, ElementPointer element).

Definition at line 79 of file TestContainer.cpp.

7.32.2.4 void ParCompMarkTest::TestContainer::test_constructor () [protected]

Tests constructor Container::Container().

Definition at line 61 of file TestContainer.cpp.

7.32.2.5 void ParCompMarkTest::TestContainer::test_destructor () [protected]

Tests destructor Container::∼Container().

Definition at line 68 of file TestContainer.cpp.

7.32.2.6 void ParCompMarkTest::TestContainer::test_get_cstd__string () [protected]

Tests method Container::get(const std::string &name).

Definition at line 106 of file TestContainer.cpp.

7.32.2.7 void ParCompMarkTest::TestContainer::test_getSize () [protected]

Tests method Container::getSize().

Definition at line 167 of file TestContainer.cpp.

7.32.2.8 void ParCompMarkTest::TestContainer::test_has_cstd__string () [protected]

Tests method Container::has(const std::string &name).

Definition at line 129 of file TestContainer.cpp.

7.32.2.9 void ParCompMarkTest::TestContainer::test_isEmpty () [protected]

Tests method Container::isEmpty().

Definition at line 200 of file TestContainer.cpp.

7.32.2.10 void ParCompMarkTest::TestContainer::test_remove_cstd__string () [protected]

Tests method Container::remove(const std::string &name).

Definition at line 143 of file TestContainer.cpp.

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

- **TestContainer.h**
- **TestContainer.cpp**

7.33 ParCompMarkTest::TestContext Class Reference

`#include <TestContext.h>`

7.33.1 Detailed Description

Remarks:

Test suite for **ParCompMark::Context**(p. 53).

This is a `CppUnit` test suite for class **ParCompMark::Context**(p. 53) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestContext.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor_Process_p** ()
- void **test_destructor** ()
- void **test_setNodes_cstd__string** ()
- void **test_init** ()
- void **test_finalize** ()

7.33.2 Member Function Documentation

7.33.2.1 void ParCompMarkTest::TestContext::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestContext.cpp.

7.33.2.2 void ParCompMarkTest::TestContext::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestContext.cpp.

7.33.2.3 void ParCompMarkTest::TestContext::test_constructor_Process_p () [protected]

Tests constructor Context::Context(Process * parent).

Definition at line 46 of file TestContext.cpp.

7.33.2.4 void ParCompMarkTest::TestContext::test_destructor () [protected]

Tests destructor Context::∼Context().

Definition at line 60 of file TestContext.cpp.

7.33.2.5 void ParCompMarkTest::TestContext::test_finalize () [protected]

Tests method Context::finalize().

Definition at line 100 of file TestContext.cpp.

7.33.2.6 void ParCompMarkTest::TestContext::test_init () [protected]

Tests method Context::init().

Definition at line 86 of file TestContext.cpp.

7.33.2.7 void ParCompMarkTest::TestContext::test_setNodes_cstd__string () [protected]

Tests method Context::setNodes(const std::string &nodes).

Definition at line 72 of file TestContext.cpp.

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

- **TestContext.h**
- **TestContext.cpp**

7.34 ParCompMarkTest::TestDummyLock Class Reference

`#include <TestDummyLock.h>`

7.34.1 Detailed Description

Remarks:

Test suite for **ParCompMark::DummyLock**(p. 67).

This is a `CppUnit` test suite for class **ParCompMark::DummyLock**(p. 67) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestDummyLock.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_lock** ()
- void **test_trylock** ()
- void **test_unlock** ()

7.34.2 Member Function Documentation

7.34.2.1 void ParCompMarkTest::TestDummyLock::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestDummyLock.cpp.

7.34.2.2 void ParCompMarkTest::TestDummyLock::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestDummyLock.cpp.

7.34.2.3 void ParCompMarkTest::TestDummyLock::test_lock () [protected]

Tests method DummyLock::lock().

Definition at line 46 of file TestDummyLock.cpp.

7.34.2.4 void ParCompMarkTest::TestDummyLock::test_trylock () [protected]

Tests method DummyLock::trylock().

Definition at line 60 of file TestDummyLock.cpp.

7.34.2.5 void ParCompMarkTest::TestDummyLock::test_unlock () [protected]

Tests method DummyLock::unlock().

Definition at line 74 of file TestDummyLock.cpp.

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

- **TestDummyLock.h**
- **TestDummyLock.cpp**

7.35 ParCompMarkTest::TestDynLoad Class Reference

`#include <TestDynLoad.h>`

7.35.1 Detailed Description

Remarks:

Test suite for **ParCompMark::DynLoad**(p. 69).

This is a `CppUnit` test suite for class **ParCompMark::DynLoad**(p. 69) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestDynLoad.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor_cstd__string** ()
- void **test_destructor** ()
- void **test_getFunc_cstd__string** ()
- void **test_load** ()
- void **test_unload** ()

7.35.2 Member Function Documentation

7.35.2.1 void ParCompMarkTest::TestDynLoad::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestDynLoad.cpp.

7.35.2.2 void ParCompMarkTest::TestDynLoad::tearDown ()

Cleans up after the test run.

Definition at line 41 of file TestDynLoad.cpp.

7.35.2.3 void ParCompMarkTest::TestDynLoad::test_constructor_cstd__string () [protected]

Tests constructor DynLoad::DynLoad(const std::string &libName).

Definition at line 51 of file TestDynLoad.cpp.

7.35.2.4 void ParCompMarkTest::TestDynLoad::test_destructor () [protected]

Tests destructor DynLoad::∼DynLoad().

Definition at line 58 of file TestDynLoad.cpp.

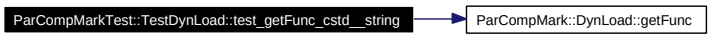
7.35.2.5 void ParCompMarkTest::TestDynLoad::test_getFunc_cstd__string () [protected]

Tests method DynLoad::getFunc(const std::string &funcName).

Definition at line 75 of file TestDynLoad.cpp.

References ParCompMark::DynLoad::getFunc().

Here is the call graph for this function:



7.35.2.6 void ParCompMarkTest::TestDynLoad::test_load () [protected]

Tests method DynLoad::load().

Definition at line 97 of file TestDynLoad.cpp.

7.35.2.7 void ParCompMarkTest::TestDynLoad::test_unload () [protected]

Tests method DynLoad::unload().

Definition at line 111 of file TestDynLoad.cpp.

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

- **TestDynLoad.h**
- **TestDynLoad.cpp**

7.36 ParCompMarkTest::TestException Class Reference

`#include <TestException.h>`

7.36.1 Detailed Description

Remarks:

Test suite for **ParCompMark::Exception**(p. 73).

This is a `CppUnit` test suite for class **ParCompMark::Exception**(p. 73) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestException.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor_cExceptionType_cstd__string_cstd__string_cstd__string_cu32** ()
- void **test_translateType_cException__ExceptionType** ()

7.36.2 Member Function Documentation

7.36.2.1 void ParCompMarkTest::TestException::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestException.cpp.

7.36.2.2 void ParCompMarkTest::TestException::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestException.cpp.

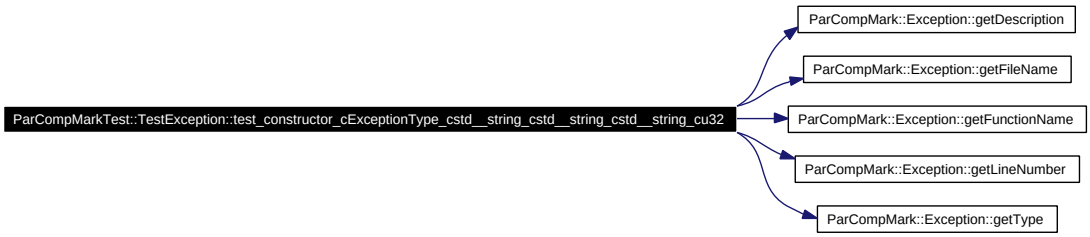
7.36.2.3 void ParCompMarkTest::TestException::test_constructor_cExceptionType_cstd__string_cstd__string_cstd__string_cu32 () `[protected]`

Tests constructor Exception::Exception(const ExceptionType &type = INTERNAL_ERROR, const std::string &description = unknown, const std::string &fileName = unknown, const std::string &functionName = unknown, const u32 &lineNumber = 0).

Definition at line 46 of file TestException.cpp.

References ParCompMark::Exception::getDescription(), ParCompMark::Exception::getFileName(), ParCompMark::Exception::getFunctionName(), ParCompMark::Exception::getLineNumber(), and ParCompMark::Exception::getType().

Here is the call graph for this function:



7.36.2.4 void ParCompMarkTest::TestException::test_translateType_cException__ExceptionType () [protected]

Tests method Exception::translateType(const Exception::ExceptionType &type).

Definition at line 66 of file TestException.cpp.

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

- **TestException.h**
- **TestException.cpp**

7.37 ParCompMarkTest::TestGLXGLContext Class Reference

`#include <TestGLXGLContext.h>`

7.37.1 Detailed Description

Remarks:

Test suite for **ParCompMark::GLXGLContext**(p. 79).

This is a `CppUnit` test suite for class **ParCompMark::GLXGLContext**(p. 79) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestGLXGLContext.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor_XDisplay__Pointer___GLXDrawable___XVisualInfo_p** ()
- void **test_destructor** ()
- void **test_initialize** ()
- void **test_finalize** ()
- void **test_setCurrent** ()

7.37.2 Member Function Documentation

7.37.2.1 void ParCompMarkTest::TestGLXGLContext::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestGLXGLContext.cpp.

7.37.2.2 void ParCompMarkTest::TestGLXGLContext::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestGLXGLContext.cpp.

7.37.2.3 void ParCompMarkTest::TestGLXGLContext::test_constructor_XDisplay__Pointer___GLXDrawable___XVisualInfo_p () [protected]

Tests constructor GLXGLContext::GLXGLContext(XDisplay::Pointer &display, ::GLXDrawable glxDrawable, ::XVisualInfo * visualInfo).

Definition at line 46 of file TestGLXGLContext.cpp.

7.37.2.4 void ParCompMarkTest::TestGLXGLContext::test_destructor () [protected]

Tests destructor GLXGLContext::∼GLXGLContext().

Definition at line 60 of file TestGLXGLContext.cpp.

7.37.2.5 void ParCompMarkTest::TestGLXGLContext::test_finalize () [protected]

Tests method GLXGLContext::finalize().

Definition at line 86 of file TestGLXGLContext.cpp.

7.37.2.6 void ParCompMarkTest::TestGLXGLContext::test_initialize () [protected]

Tests method GLXGLContext::initialize().

Definition at line 72 of file TestGLXGLContext.cpp.

7.37.2.7 void ParCompMarkTest::TestGLXGLContext::test_setCurrent () [protected]

Tests method GLXGLContext::setCurrent().

Definition at line 100 of file TestGLXGLContext.cpp.

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

- **TestGLXGLContext.h**
- **TestGLXGLContext.cpp**

7.38 ParCompMarkTest::TestGLXRenderWindow Class Reference

`#include <TestGLXRenderWindow.h>`

7.38.1 Detailed Description

Remarks:

Test suite for **ParCompMark::GLXRenderWindow**(p. 85).

This is a `CppUnit` test suite for class **ParCompMark::GLXRenderWindow**(p. 85) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestGLXRenderWindow.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor_XDisplay__Pointer_cstd__string_cbool_cu32_cs32_cs32_cu32_cu32_cu32** ()
- void **test_destructor** ()
- void **test_reposition_cs32_cs32** ()
- void **test_resize_cu32_cu32** ()
- void **test_startFrame** ()
- void **test_finishFrame** ()
- void **test_setCurrent** ()
- void **test_initialize** ()
- void **test_finalize** ()
- void **test_createWindow** ()
- void **test_destroyWindow** ()
- void **test_resetStatistics** ()
- void **test_updateStatistics** ()
- void **test__reposition** ()
- void **test__resize** ()

7.38.2 Member Function Documentation

7.38.2.1 void ParCompMarkTest::TestGLXRenderWindow::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestGLXRenderWindow.cpp.

7.38.2.2 void ParCompMarkTest::TestGLXRenderWindow::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestGLXRenderWindow.cpp.

7.38.2.3 void ParCompMarkTest::TestGLXRenderWindow::test__reposition () [protected]

Tests method GLXRenderWindow::_reposition().

Definition at line 230 of file TestGLXRenderWindow.cpp.

7.38.2.4 void ParCompMarkTest::TestGLXRenderWindow::test__resize () [protected]

Tests method GLXRenderWindow::_resize().

Definition at line 244 of file TestGLXRenderWindow.cpp.

7.38.2.5 void ParCompMarkTest::TestGLXRenderWindow::test_constructor_-XDisplay__Pointer_cstd__string_cbool_cu32_cs32_cs32_cu32_cu32_cu32 () [protected]

Tests constructor GLXRenderWindow::GLXRenderWindow(XDisplay::Pointer &display, const std::string caption = PCM Framework, const bool &fullScreen = true, const u32 &colourDepth = 0, const s32 &left = GLXRenderWindow::CENTERED, const s32 &top = GLXRenderWindow::CENTERED, const u32 &width = GLXRenderWindow::MAXIMALSIZE, const u32 &height = GLXRenderWindow::MAXIMALSIZE, const u32 &fsaaSamples = 0).

Definition at line 47 of file TestGLXRenderWindow.cpp.

7.38.2.6 void ParCompMarkTest::TestGLXRenderWindow::test_createWindow () [protected]

Tests method GLXRenderWindow::createWindow().

Definition at line 174 of file TestGLXRenderWindow.cpp.

7.38.2.7 void ParCompMarkTest::TestGLXRenderWindow::test_destroyWindow () [protected]

Tests method GLXRenderWindow::destroyWindow().

Definition at line 188 of file TestGLXRenderWindow.cpp.

7.38.2.8 void ParCompMarkTest::TestGLXRenderWindow::test_destructor () [protected]

Tests destructor GLXRenderWindow::∼GLXRenderWindow().

Definition at line 61 of file TestGLXRenderWindow.cpp.

7.38.2.9 void ParCompMarkTest::TestGLXRenderWindow::test_finalize () [protected]

Tests method GLXRenderWindow::finalize().

Definition at line 160 of file TestGLXRenderWindow.cpp.

7.38.2.10 void ParCompMarkTest::TestGLXRenderWindow::test_finishFrame () [protected]

Tests method GLXRenderWindow::finishFrame().

Definition at line 115 of file TestGLXRenderWindow.cpp.

7.38.2.11 void ParCompMarkTest::TestGLXRenderWindow::test_initialize () [protected]

Tests method GLXRenderWindow::initialize().

Definition at line 146 of file TestGLXRenderWindow.cpp.

7.38.2.12 void ParCompMarkTest::TestGLXRenderWindow::test_reposition_cs32_cs32 () [protected]

Tests method GLXRenderWindow::reposition(const s32 &left, const s32 &top).

Definition at line 73 of file TestGLXRenderWindow.cpp.

7.38.2.13 void ParCompMarkTest::TestGLXRenderWindow::test_resetStatistics () [protected]

Tests method GLXRenderWindow::resetStatistics().

Definition at line 202 of file TestGLXRenderWindow.cpp.

7.38.2.14 void ParCompMarkTest::TestGLXRenderWindow::test_resize_cu32_cu32 () [protected]

Tests method GLXRenderWindow::resize(const u32 &width, const u32 &height).

Definition at line 87 of file TestGLXRenderWindow.cpp.

7.38.2.15 void ParCompMarkTest::TestGLXRenderWindow::test_setCurrent () [protected]

Tests method GLXRenderWindow::setCurrent().

Definition at line 129 of file TestGLXRenderWindow.cpp.

7.38.2.16 void ParCompMarkTest::TestGLXRenderWindow::test_startFrame () [protected]

Tests method GLXRenderWindow::startFrame().

Definition at line 101 of file TestGLXRenderWindow.cpp.

7.38.2.17 void ParCompMarkTest::TestGLXRenderWindow::test_updateStatistics () [protected]

Tests method GLXRenderWindow::updateStatistics().

Definition at line 216 of file TestGLXRenderWindow.cpp.

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

- **TestGLXRenderWindow.h**
- **TestGLXRenderWindow.cpp**

7.39 ParCompMarkTest::TestHandleClient Class Reference

`#include <TestHandleClient.h>`

7.39.1 Detailed Description

Remarks:

Test suite for **ParCompMark::HandleClient**(p. 107).

This is a `CppUnit` test suite for class **ParCompMark::HandleClient**(p. 107) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestHandleClient.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor** ()
- void **test_destructor** ()
- void **test_sendMessage_cstd__string_cstd__string** ()
- void **test_task** ()

7.39.2 Member Function Documentation

7.39.2.1 void ParCompMarkTest::TestHandleClient::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestHandleClient.cpp.

7.39.2.2 void ParCompMarkTest::TestHandleClient::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestHandleClient.cpp.

7.39.2.3 void ParCompMarkTest::TestHandleClient::test_constructor () [protected]

Tests constructor HandleClient::HandleClient().

Definition at line 46 of file TestHandleClient.cpp.

7.39.2.4 void ParCompMarkTest::TestHandleClient::test_destructor () [protected]

Tests destructor HandleClient::∼HandleClient().

Definition at line 60 of file TestHandleClient.cpp.

7.39.2.5 void ParCompMarkTest::TestHandleClient::test_sendMessage_cstd__string_cstd__string () [protected]

Tests method HandleClient::sendMessage(const std::string &mType, const std::string &message).

Definition at line 72 of file TestHandleClient.cpp.

7.39.2.6 void ParCompMarkTest::TestHandleClient::test_task () [protected]

Tests method HandleClient::task().

Definition at line 86 of file TestHandleClient.cpp.

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

- **TestHandleClient.h**
- **TestHandleClient.cpp**

7.40 ParCompMarkTest::TestHost Class Reference

`#include <TestHost.h>`

7.40.1 Detailed Description

Remarks:

Test suite for **ParCompMark::Host**(p. 112).

This is a `CppUnit` test suite for class **ParCompMark::Host**(p. 112) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestHost.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor_cstd__string** ()
- void **test_destructor** ()
- void **test_openXDisplay_cstd__string** ()
- void **test_init_cstd__string** ()

7.40.2 Member Function Documentation

7.40.2.1 void ParCompMarkTest::TestHost::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestHost.cpp.

7.40.2.2 void ParCompMarkTest::TestHost::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestHost.cpp.

7.40.2.3 void ParCompMarkTest::TestHost::test_constructor_cstd__string () [protected]

Tests constructor Host::Host(const std::string &name).

Definition at line 46 of file TestHost.cpp.

7.40.2.4 void ParCompMarkTest::TestHost::test_destructor () [protected]

Tests destructor Host::~Host().

Definition at line 60 of file TestHost.cpp.

7.40.2.5 void ParCompMarkTest::TestHost::test_init_cstd__string () [protected]

Tests method Host::init(const std::string &conf).

Definition at line 90 of file TestHost.cpp.

7.40.2.6 void ParCompMarkTest::TestHost::test_openXDisplay_cstd__string () [protected]

Tests method Host::openXDisplay(const std::string &displayName =).

Definition at line 72 of file TestHost.cpp.

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

- **TestHost.h**
- **TestHost.cpp**

7.41 ParCompMarkTest::TestLock Class Reference

`#include <TestLock.h>`

7.41.1 Detailed Description

Remarks:

Test suite for **ParCompMark::Lock**(p. 116).

This is a `CppUnit` test suite for class **ParCompMark::Lock**(p. 116) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestLock.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

7.41.2 Member Function Documentation

7.41.2.1 void ParCompMarkTest::TestLock::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestLock.cpp.

7.41.2.2 void ParCompMarkTest::TestLock::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestLock.cpp.

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

- **TestLock.h**
- **TestLock.cpp**

7.42 ParCompMarkTest::TestLogger Class Reference

`#include <TestLogger.h>`

7.42.1 Detailed Description

Remarks:

Test suite for **ParCompMark::Logger**(p. 118).

This is a `CppUnit` test suite for class **ParCompMark::Logger**(p. 118) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestLogger.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor_cstd__string** ()
- void **test_destructor** ()
- void **test_translateLogLevel_cLogLevel** ()
- void **test_init** ()
- void **test_log_cLogLevel_cstd__string** ()
- void **test_log_cException** ()

7.42.2 Member Function Documentation

7.42.2.1 void ParCompMarkTest::TestLogger::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestLogger.cpp.

7.42.2.2 void ParCompMarkTest::TestLogger::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestLogger.cpp.

7.42.2.3 void ParCompMarkTest::TestLogger::test_constructor_cstd__string () [protected]

Tests constructor Logger::Logger(const std::string &logFileName = pcm.log).

Definition at line 46 of file TestLogger.cpp.

7.42.2.4 void ParCompMarkTest::TestLogger::test_destructor () [protected]

Tests destructor Logger::∼Logger().

Definition at line 61 of file TestLogger.cpp.

7.42.2.5 void ParCompMarkTest::TestLogger::test_init () [protected]

Tests method Logger::init().

Definition at line 94 of file TestLogger.cpp.

7.42.2.6 void ParCompMarkTest::TestLogger::test_log_cException () [protected]

Tests method Logger::log(const Exception &exception).

Definition at line 137 of file TestLogger.cpp.

7.42.2.7 void ParCompMarkTest::TestLogger::test_log_cLogLevel_cstd__string () [protected]

Tests method Logger::log(const LogLevel &loglevel, const std::string &message).

Definition at line 112 of file TestLogger.cpp.

7.42.2.8 void ParCompMarkTest::TestLogger::test_translateLogLevel_cLogLevel () [protected]

Tests method Logger::translateLogLevel(const LogLevel &loglevel).

Definition at line 72 of file TestLogger.cpp.

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

- **TestLogger.h**
- **TestLogger.cpp**

7.43 ParCompMarkTest::TestMutex Class Reference

`#include <TestMutex.h>`

7.43.1 Detailed Description

Remarks:

Test suite for **ParCompMark::Mutex**(p. 127).

This is a `CppUnit` test suite for class **ParCompMark::Mutex**(p. 127) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestMutex.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor** ()
- void **test_destructor** ()
- void **test_lock** ()
- void **test_trylock** ()
- void **test_unlock** ()

7.43.2 Member Function Documentation

7.43.2.1 void ParCompMarkTest::TestMutex::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestMutex.cpp.

7.43.2.2 void ParCompMarkTest::TestMutex::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestMutex.cpp.

7.43.2.3 void ParCompMarkTest::TestMutex::test_constructor () [protected]

Tests constructor Mutex::Mutex().

Definition at line 46 of file TestMutex.cpp.

References ParCompMark::Mutex::getLocked().

Here is the call graph for this function:

7.43.2.4 void ParCompMarkTest::TestMutex::test_destructor () [protected]

Tests destructor Mutex::∼Mutex().

Definition at line 57 of file TestMutex.cpp.

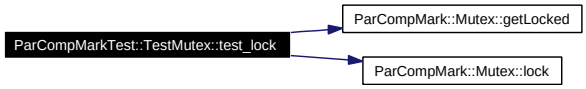
7.43.2.5 void ParCompMarkTest::TestMutex::test_lock () [protected]

Tests method Mutex::lock().

Definition at line 68 of file TestMutex.cpp.

References ParCompMark::Mutex::getLocked(), and ParCompMark::Mutex::lock().

Here is the call graph for this function:



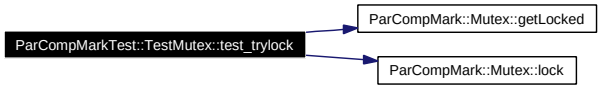
7.43.2.6 void ParCompMarkTest::TestMutex::test_trylock () [protected]

Tests method Mutex::trylock().

Definition at line 85 of file TestMutex.cpp.

References ParCompMark::Mutex::getLocked(), and ParCompMark::Mutex::lock().

Here is the call graph for this function:



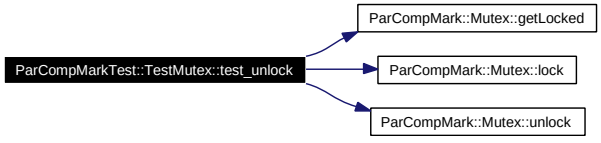
7.43.2.7 void ParCompMarkTest::TestMutex::test_unlock () [protected]

Tests method Mutex::unlock().

Definition at line 103 of file TestMutex.cpp.

References ParCompMark::Mutex::getLocked(), ParCompMark::Mutex::lock(), and ParCompMark::Mutex::unlock().

Here is the call graph for this function:



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

- **TestMutex.h**
- **TestMutex.cpp**

7.44 ParCompMarkTest::TestName Class Reference

`#include <TestName.h>`

7.44.1 Detailed Description

Remarks:

Test suite for **ParCompMark::Name**(p. 131).

This is a `CppUnit` test suite for class **ParCompMark::Name**(p. 131) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestName.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor** ()
- void **test_constructor_cstd__string** ()
- void **test_destructor** ()

7.44.2 Member Function Documentation

7.44.2.1 void ParCompMarkTest::TestName::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestName.cpp.

7.44.2.2 void ParCompMarkTest::TestName::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestName.cpp.

7.44.2.3 void ParCompMarkTest::TestName::test_constructor () [protected]

Tests constructor Name::Name().

Definition at line 46 of file TestName.cpp.

References ParCompMark::Name::mName.

7.44.2.4 void ParCompMarkTest::TestName::test_constructor_cstd__string () [protected]

Tests constructor Name::Name(const std::string &name).

Definition at line 55 of file TestName.cpp.

References ParCompMark::Name::mName.

7.44.2.5 void ParCompMarkTest::TestName::test_destructor () [protected]

Tests destructor Name::~Name().

Definition at line 64 of file TestName.cpp.

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

- **TestName.h**
- **TestName.cpp**

7.45 ParCompMarkTest::TestNetwork Class Reference

`#include <TestNetwork.h>`

7.45.1 Detailed Description

Remarks:

Test suite for **ParCompMark::Network**(p. 134).

This is a `CppUnit` test suite for class **ParCompMark::Network**(p. 134) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestNetwork.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor_cstd__string** ()
- void **test_destructor** ()
- void **test_str2enum_cstd__string** ()
- void **test_enum2str_cNetwork__MessageType** ()
- void **test_initServer** ()
- void **test_initClient** ()
- void **test_closeClient** ()
- void **test_initBroadcastSend** ()
- void **test_initBroadcastRecieve** ()
- void **test_sendMessage_cMessageType_cstd__string** ()
- void **test_sendBroadcastMessage_cMessageType_cstd__string** ()
- void **test_getIP** ()
- void **test_recieveBroadcastMessage** ()
- void **test_recieveMessage** ()
- void **test_acceptClientConnection** ()
- void **test_initNetwork** ()
- void **test_buildCluster** ()
- void **test_task** ()

7.45.2 Member Function Documentation

7.45.2.1 void ParCompMarkTest::TestNetwork::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestNetwork.cpp.

7.45.2.2 void ParCompMarkTest::TestNetwork::tearDown ()

Cleans up after the test run.

Definition at line 39 of file TestNetwork.cpp.

7.45.2.3 void ParCompMarkTest::TestNetwork::test_acceptClientConnection () [protected]

Tests method Network::acceptClientConnection().

Definition at line 310 of file TestNetwork.cpp.

7.45.2.4 void ParCompMarkTest::TestNetwork::test_buildCluster () [protected]

Tests method Network::buildCluster().

Definition at line 338 of file TestNetwork.cpp.

7.45.2.5 void ParCompMarkTest::TestNetwork::test_closeClient () [protected]

Tests method Network::closeClient().

Definition at line 175 of file TestNetwork.cpp.

7.45.2.6 void ParCompMarkTest::TestNetwork::test_constructor_cstd__string () [protected]

Tests constructor Network::Network(const std::string &name).

Definition at line 49 of file TestNetwork.cpp.

7.45.2.7 void ParCompMarkTest::TestNetwork::test_destructor () [protected]

Tests destructor Network::∼Network().

Definition at line 63 of file TestNetwork.cpp.

7.45.2.8 void ParCompMarkTest::TestNetwork::test_enum2str_cNetwork__MessageType () [protected]

Tests method Network::enum2str(const Network::MessageType &enumType).

Definition at line 93 of file TestNetwork.cpp.

7.45.2.9 void ParCompMarkTest::TestNetwork::test_getIP () [protected]

Tests method Network::getIP().

Definition at line 268 of file TestNetwork.cpp.

7.45.2.10 void ParCompMarkTest::TestNetwork::test_initBroadcastRecieve () [protected]

Tests method Network::initBroadcastRecieve().

Definition at line 225 of file TestNetwork.cpp.

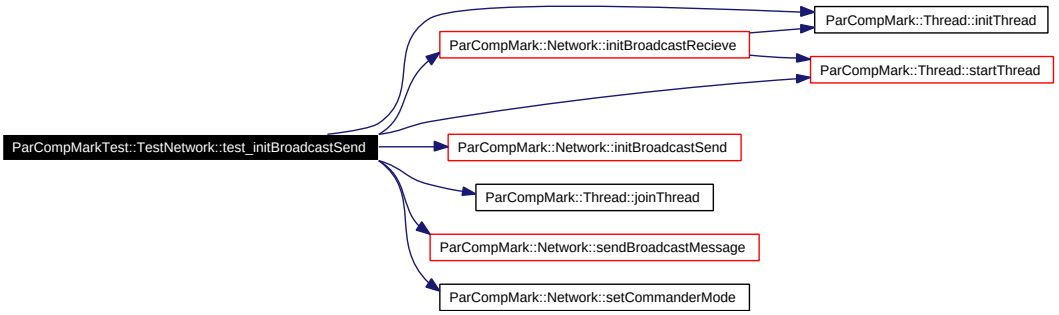
7.45.2.11 void ParCompMarkTest::TestNetwork::test_initBroadcastSend () [protected]

Tests method Network::initBroadcastSend().

Definition at line 189 of file TestNetwork.cpp.

References ParCompMark::Network::initBroadcastRecieve(), ParCompMark::Network::initBroadcastSend(), ParCompMark::Thread::initThread(), ParCompMark::Thread::joinThread(), ParCompMark::Network::mCommanderMode, ParCompMark::Network::sendBroadcastMessage(), ParCompMark::Network::setCommanderMode(), and ParCompMark::Thread::startThread().

Here is the call graph for this function:



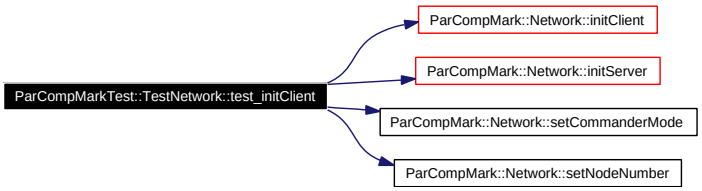
7.45.2.12 void ParCompMarkTest::TestNetwork::test_initClient () [protected]

Tests method Network::initClient().

Definition at line 126 of file TestNetwork.cpp.

References ParCompMark::Network::initClient(), ParCompMark::Network::initServer(), ParCompMark::Network::setCommanderMode(), and ParCompMark::Network::setNodeNumber().

Here is the call graph for this function:



7.45.2.13 void ParCompMarkTest::TestNetwork::test_initNetwork () [protected]

Tests method Network::initNetwork().

Definition at line 324 of file TestNetwork.cpp.

7.45.2.14 void ParCompMarkTest::TestNetwork::test_initServer () [protected]

Tests method Network::initServer().

Definition at line 111 of file TestNetwork.cpp.

7.45.2.15 void ParCompMarkTest::TestNetwork::test_recieveBroadcastMessage () [protected]

Tests method Network::recieveBroadcastMessage().

Definition at line 282 of file TestNetwork.cpp.

7.45.2.16 void ParCompMarkTest::TestNetwork::test_recieveMessage () [protected]

Tests method Network::recieveMessage().

Definition at line 296 of file TestNetwork.cpp.

7.45.2.17 void ParCompMarkTest::TestNetwork::test_sendBroadcastMessage_cMessageType_-cstd__string () [protected]

Tests method Network::sendBroadcastMessage(const MessageType &mType, const std::string &message).

Definition at line 254 of file TestNetwork.cpp.

7.45.2.18 void ParCompMarkTest::TestNetwork::test_sendMessage_cMessageType_cstd__string () [protected]

Tests method Network::sendMessage(const MessageType &mType, const std::string &message).

Definition at line 240 of file TestNetwork.cpp.

7.45.2.19 void ParCompMarkTest::TestNetwork::test_str2enum_cstd__string () [protected]

Tests method Network::str2enum(const std::string &strType).

Definition at line 79 of file TestNetwork.cpp.

7.45.2.20 void ParCompMarkTest::TestNetwork::test_task () [protected]

Tests method Network::task().

Definition at line 352 of file TestNetwork.cpp.

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

- **TestNetwork.h**
- **TestNetwork.cpp**

7.46 ParCompMarkTest::TestNode Class Reference

`#include <TestNode.h>`

7.46.1 Detailed Description

Remarks:

Test suite for **ParCompMark::Node**(p. 150).

This is a `CppUnit` test suite for class **ParCompMark::Node**(p. 150) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestNode.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor_Host_p** ()
- void **test_destructor** ()
- void **test_init_cstd__string** ()

7.46.2 Member Function Documentation

7.46.2.1 void ParCompMarkTest::TestNode::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestNode.cpp.

7.46.2.2 void ParCompMarkTest::TestNode::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestNode.cpp.

7.46.2.3 void ParCompMarkTest::TestNode::test_constructor_Host_p () [protected]

Tests constructor Node::Node(Host $*$ parent).

Definition at line 46 of file TestNode.cpp.

7.46.2.4 void ParCompMarkTest::TestNode::test_destructor () [protected]

Tests destructor Node::$\sim$Node().

Definition at line 60 of file TestNode.cpp.

7.46.2.5 void ParCompMarkTest::TestNode::test_init_cstd__string () [protected]

Tests method Node::init(const std::string &conf).

Definition at line 72 of file TestNode.cpp.

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

- **TestNode.h**
- **TestNode.cpp**

7.47 ParCompMarkTest::TestOldContainer Class Reference

`#include <TestOldContainer.h>`

7.47.1 Detailed Description

Remarks:

Test suite for **ParCompMark::OldContainer**(p. 154).

This is a `CppUnit` test suite for class **ParCompMark::OldContainer**(p. 154) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestOldContainer.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor** ()
- void **test_destructor** ()
- void **test_add_Name_p** ()
- void **test_add_std__string** ()
- void **test_get_cstd__string** ()
- void **test_has_cstd__string** ()
- void **test_remove_cstd__string** ()
- void **test_remove_Name_p** ()

7.47.2 Member Function Documentation

7.47.2.1 void ParCompMarkTest::TestOldContainer::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestOldContainer.cpp.

7.47.2.2 void ParCompMarkTest::TestOldContainer::tearDown ()

Cleans up after the test run.

Definition at line 39 of file TestOldContainer.cpp.

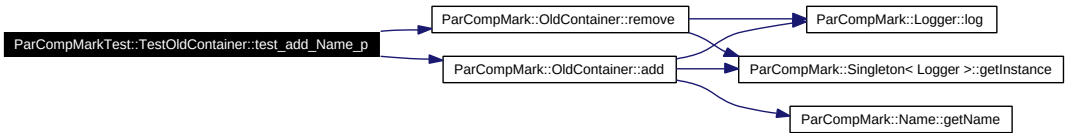
7.47.2.3 void ParCompMarkTest::TestOldContainer::test_add_Name_p () [protected]

Tests method OldContainer::add(Name * &element).

Definition at line 72 of file TestOldContainer.cpp.

References ParCompMark::OldContainer::add(), and ParCompMark::OldContainer::remove().

Here is the call graph for this function:



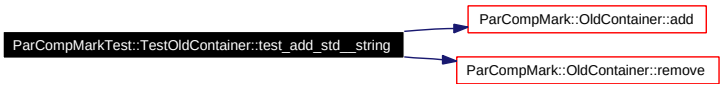
7.47.2.4 void ParCompMarkTest::TestOldContainer::test_add_std__string () [protected]

Tests method OldContainer::add(std::string name).

Definition at line 96 of file TestOldContainer.cpp.

References ParCompMark::OldContainer::add(), ParCompMark::OldContainer::mElementNumber, and ParCompMark::OldContainer::remove().

Here is the call graph for this function:



7.47.2.5 void ParCompMarkTest::TestOldContainer::test_constructor () [protected]

Tests constructor OldContainer::OldContainer().

Definition at line 49 of file TestOldContainer.cpp.

7.47.2.6 void ParCompMarkTest::TestOldContainer::test_destructor () [protected]

Tests destructor OldContainer::∼OldContainer().

Definition at line 57 of file TestOldContainer.cpp.

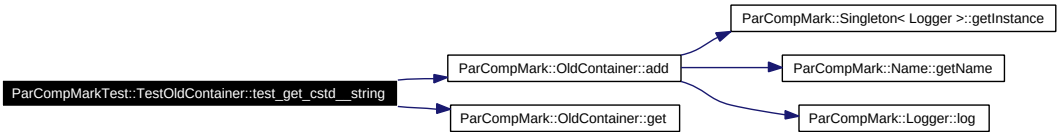
7.47.2.7 void ParCompMarkTest::TestOldContainer::test_get_cstd__string () [protected]

Tests method OldContainer::get(const std::string &name).

Definition at line 118 of file TestOldContainer.cpp.

References ParCompMark::OldContainer::add(), and ParCompMark::OldContainer::get().

Here is the call graph for this function:



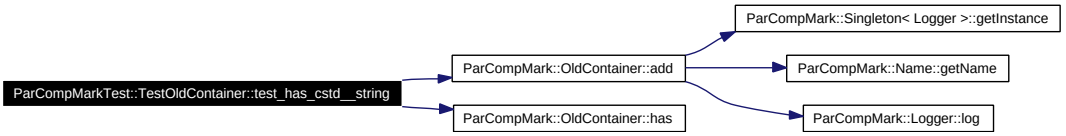
7.47.2.8 void ParCompMarkTest::TestOldContainer::test_has_cstd__string () [protected]

Tests method OldContainer::has(const std::string &name).

Definition at line 134 of file TestOldContainer.cpp.

References ParCompMark::OldContainer::add(), and ParCompMark::OldContainer::has().

Here is the call graph for this function:



7.47.2.9 void ParCompMarkTest::TestOldContainer::test_remove_cstd__string () [protected]

Tests method OldContainer::remove(const std::string &name).

Definition at line 155 of file TestOldContainer.cpp.

7.47.2.10 void ParCompMarkTest::TestOldContainer::test_remove_Name_p () [protected]

Tests method OldContainer::remove(Name * element).

Definition at line 169 of file TestOldContainer.cpp.

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

- **TestOldContainer.h**
- **TestOldContainer.cpp**

7.48 ParCompMarkTest::TestOutputNode Class Reference

`#include <TestOutputNode.h>`

7.48.1 Detailed Description

Remarks:

Test suite for **ParCompMark::OutputNode**(p. 159).

This is a `CppUnit` test suite for class **ParCompMark::OutputNode**(p. 159) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestOutputNode.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor_cstd__string_cNodeType** ()
- void **test_constructor_cstd__string** ()
- void **test_destructor** ()
- void **test__testXMLName_cstd__string** ()
- void **test__convertSpecialChars_cstd__string** ()
- void **test_hasChildNode_cstd__string** ()
- void **test_getChildNode_cstd__string** ()
- void **test_createChildNode_cstd__string_cNodeType** ()
- void **test_createChildNode_cstd__string** ()
- void **test_addChildNode_OutputNode__Pointer** ()
- void **test_hasAttribute_cstd__string** ()
- void **test_getAttribute_cstd__string** ()
- void **test_setAttribute_cstd__string_cstd__string** ()
- void **test_serialize2XML_std__ostringstream** ()
- void **test_serialize2XML** ()

7.48.2 Member Function Documentation

7.48.2.1 void ParCompMarkTest::TestOutputNode::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestOutputNode.cpp.

7.48.2.2 void ParCompMarkTest::TestOutputNode::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestOutputNode.cpp.

7.48.2.3 void ParCompMarkTest::TestOutputNode::test__convertSpecialChars_cstd__string () [protected]

Tests method OutputNode::_convertSpecialChars(const std::string &string).

Definition at line 129 of file TestOutputNode.cpp.

7.48.2.4 void ParCompMarkTest::TestOutputNode::test__testXMLName_cstd__string () [protected]

Tests method OutputNode::_testXMLName(const std::string &name).

Definition at line 115 of file TestOutputNode.cpp.

7.48.2.5 void ParCompMarkTest::TestOutputNode::test_addChildNode_OutputNode__Pointer () [protected]

Tests method OutputNode::addChildNode(OutputNode::Pointer &child).

Definition at line 248 of file TestOutputNode.cpp.

7.48.2.6 void ParCompMarkTest::TestOutputNode::test_constructor_cstd__string () [protected]

Tests constructor OutputNode::OutputNode(const std::string &text).

Definition at line 75 of file TestOutputNode.cpp.

References ParCompMark::Name::mName, ParCompMark::OutputNode::mText, and ParCompMark::OutputNode::mType.

7.48.2.7 void ParCompMarkTest::TestOutputNode::test_constructor_cstd__string_cNodeType () [protected]

Tests constructor OutputNode::OutputNode(const std::string &name, const NodeType &type).

Definition at line 46 of file TestOutputNode.cpp.

References ParCompMark::Name::mName, ParCompMark::OutputNode::mText, and ParCompMark::OutputNode::mType.

7.48.2.8 void ParCompMarkTest::TestOutputNode::test_createChildNode_cstd__string () [protected]

Tests method OutputNode::createChildNode(const std::string &text).

Definition at line 216 of file TestOutputNode.cpp.

7.48.2.9 void ParCompMarkTest::TestOutputNode::test_createChildNode_cstd__string_cNodeType () [protected]

Tests method OutputNode::createChildNode(const std::string &name, const NodeType &type).

Definition at line 181 of file TestOutputNode.cpp.

7.48.2.10 void ParCompMarkTest::TestOutputNode::test_destructor () [protected]

Tests destructor OutputNode::∼OutputNode().

Definition at line 104 of file TestOutputNode.cpp.

7.48.2.11 void ParCompMarkTest::TestOutputNode::test_getAttribute_cstd__string () [protected]

Tests method OutputNode::getAttribute(const std::string &attribute).

Definition at line 312 of file TestOutputNode.cpp.

7.48.2.12 void ParCompMarkTest::TestOutputNode::test_getChildNode_cstd__string () [protected]

Tests method OutputNode::getChildNode(const std::string &name).

Definition at line 167 of file TestOutputNode.cpp.

7.48.2.13 void ParCompMarkTest::TestOutputNode::test_hasAttribute_cstd__string () [protected]

Tests method OutputNode::hasAttribute(const std::string &attribute).

Definition at line 288 of file TestOutputNode.cpp.

7.48.2.14 void ParCompMarkTest::TestOutputNode::test_hasChildNode_cstd__string () [protected]

Tests method OutputNode::hasChildNode(const std::string &name).

Definition at line 147 of file TestOutputNode.cpp.

7.48.2.15 void ParCompMarkTest::TestOutputNode::test_serialize2XML () [protected]

Tests method OutputNode::serialize2XML().

Definition at line 392 of file TestOutputNode.cpp.

7.48.2.16 void ParCompMarkTest::TestOutputNode::test_serialize2XML_std__ostringstream () [protected]

Tests method OutputNode::serialize2XML(std::ostringstream &osstr).

Definition at line 360 of file TestOutputNode.cpp.

7.48.2.17 void ParCompMarkTest::TestOutputNode::test_setAttribute_cstd__string_cstd__string () [protected]

Tests method OutputNode::setAttribute(const std::string &attribute, const std::string &value).

Definition at line 336 of file TestOutputNode.cpp.

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

- **TestOutputNode.h**
- **TestOutputNode.cpp**

7.49 ParCompMarkTest::TestPointer Class Reference

`#include <TestPointer.h>`

7.49.1 Detailed Description

Remarks:

Test suite for **ParCompMark::Pointer**(p. 170).

This is a `CppUnit` test suite for class **ParCompMark::Pointer**(p. 170) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestPointer.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor** ()
- void **test_constructor_cT_p_cbool** ()
- void **test_constructor_Pointer___T__Lock__** ()
- void **test_constructor_cPointer___T__Lock__** ()
- void **test_destructor** ()
- void **test_operator_assignment_cT_p** ()
- void **test_operator_assignment_Pointer___T__Lock__** ()
- void **test_operator_assignment_cPointer___T__Lock__** ()
- void **test_operator_memberSelection** ()
- void **test_operator_equality_cT_p** ()
- void **test_operator_equality_Pointer___T__Lock__** ()
- void **test_operator_inequality_cT_p** ()
- void **test_operator_inequality_Pointer___T__Lock__** ()
- void **test_isNull** ()
- void **test_isNotNull** ()
- void **test_reference_cT_p** ()
- void **test_getPtr** ()
- void **test_kill_cbool** ()
- void **test_setNull_cbool** ()
- void **test_lock** ()
- void **test_trylock** ()
- void **test_unlock** ()
- void **test__deletePointer_cbool** ()
- void **test__assignCPointer_cT_p_cbool** ()
- void **test__assignPointer_Pointer___T__Lock__** ()
- void **test__equalsCPointer_cT_p** ()
- void **test__equalsPointer_Pointer___T__Lock__** ()

7.49.2 Member Function Documentation

7.49.2.1 void ParCompMarkTest::TestPointer::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestPointer.cpp.

7.49.2.2 void ParCompMarkTest::TestPointer::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestPointer.cpp.

7.49.2.3 void ParCompMarkTest::TestPointer::test__assignCPointer_cT_p_cbool () [protected]

Tests method Pointer::_assignCPointer(const T * pointer, const bool &takeOwnership = true).

Definition at line 631 of file TestPointer.cpp.

7.49.2.4 void ParCompMarkTest::TestPointer::test__assignPointer_Pointer___T__Lock__ () [protected]

Tests method Pointer::_assignPointer(Pointer < T, Lock > &pointer).

Definition at line 645 of file TestPointer.cpp.

7.49.2.5 void ParCompMarkTest::TestPointer::test__deletePointer_cbool () [protected]

Tests method Pointer::_deletePointer(const bool &force = false).

Definition at line 617 of file TestPointer.cpp.

7.49.2.6 void ParCompMarkTest::TestPointer::test__equalsCPointer_cT_p () [protected]

Tests method Pointer::_equalsCPointer(const T * pointer).

Definition at line 659 of file TestPointer.cpp.

7.49.2.7 void ParCompMarkTest::TestPointer::test__equalsPointer_Pointer___T__Lock__ () [protected]

Tests method Pointer::_equalsPointer(Pointer < T, Lock > &pointer).

Definition at line 673 of file TestPointer.cpp.

7.49.2.8 void ParCompMarkTest::TestPointer::test_constructor () [protected]

Tests constructor Pointer::Pointer().

Definition at line 46 of file TestPointer.cpp.

7.49.2.9 void ParCompMarkTest::TestPointer::test_constructor_cPointer___T__Lock__ () [protected]

Tests constructor Pointer::Pointer(const Pointer < T, Lock > &pointer).

Definition at line 118 of file TestPointer.cpp.

7.49.2.10 void ParCompMarkTest::TestPointer::test_constructor_cT_p_cbool () [protected]

Tests constructor Pointer::Pointer(const T * pointer, const bool &takeOwnership = true).

Definition at line 66 of file TestPointer.cpp.

7.49.2.11 void ParCompMarkTest::TestPointer::test_constructor_Pointer___T__Lock__ () [protected]

Tests constructor Pointer::Pointer(Pointer < T, Lock > &pointer).

Definition at line 97 of file TestPointer.cpp.

7.49.2.12 void ParCompMarkTest::TestPointer::test_destructor () [protected]

Tests destructor Pointer::~Pointer().

Definition at line 139 of file TestPointer.cpp.

7.49.2.13 void ParCompMarkTest::TestPointer::test_getPtr () [protected]

Tests method Pointer::getPtr().

Definition at line 499 of file TestPointer.cpp.

7.49.2.14 void ParCompMarkTest::TestPointer::test_isNotNull () [protected]

Tests method Pointer::isNotNull().

Definition at line 444 of file TestPointer.cpp.

7.49.2.15 void ParCompMarkTest::TestPointer::test_isNull () [protected]

Tests method Pointer::isNull().

Definition at line 422 of file TestPointer.cpp.

7.49.2.16 void ParCompMarkTest::TestPointer::test_kill_cbool () [protected]

Tests method Pointer::kill(const bool &force = false).

Definition at line 517 of file TestPointer.cpp.

7.49.2.17 void ParCompMarkTest::TestPointer::test_lock () [protected]

Tests method Pointer::lock().

Definition at line 575 of file TestPointer.cpp.

7.49.2.18 void ParCompMarkTest::TestPointer::test_operator_assignment_cPointer___T__-Lock__ () [protected]

Tests operator Pointer::operator=(const Pointer < T, Lock > &pointer).

Definition at line 244 of file TestPointer.cpp.

7.49.2.19 void ParCompMarkTest::TestPointer::test_operator_assignment_cT_p () [protected]

Tests operator Pointer::operator=(const T * pointer).

Definition at line 159 of file TestPointer.cpp.

7.49.2.20 void ParCompMarkTest::TestPointer::test_operator_assignment_Pointer___T__Lock_-_ () [protected]

Tests operator Pointer::operator=(Pointer < T, Lock > &pointer).

Definition at line 192 of file TestPointer.cpp.

7.49.2.21 void ParCompMarkTest::TestPointer::test_operator_equality_cT_p () [protected]

Tests operator Pointer::operator==(const T * pointer).

Definition at line 336 of file TestPointer.cpp.

7.49.2.22 void ParCompMarkTest::TestPointer::test_operator_equality_Pointer___T__Lock__ () [protected]

Tests operator Pointer::operator==(Pointer < T, Lock > &pointer).

Definition at line 353 of file TestPointer.cpp.

7.49.2.23 void ParCompMarkTest::TestPointer::test_operator_inequality_cT_p () [protected]

Tests operator Pointer::operator!=(const T * pointer).

Definition at line 377 of file TestPointer.cpp.

7.49.2.24 void ParCompMarkTest::TestPointer::test_operator_inequality_Pointer___T__Lock__ () [protected]

Tests operator Pointer::operator!=(Pointer < T, Lock > &pointer).

Definition at line 394 of file TestPointer.cpp.

7.49.2.25 void ParCompMarkTest::TestPointer::test_operator_memberSelection () [protected]

Tests operator Pointer::operator → ().

Definition at line 258 of file TestPointer.cpp.

7.49.2.26 void ParCompMarkTest::TestPointer::test_reference_cT_p () [protected]

Tests method Pointer::reference(const T ∗ pointer).

Definition at line 466 of file TestPointer.cpp.

7.49.2.27 void ParCompMarkTest::TestPointer::test_setNull_cbool () [protected]

Tests method Pointer::setNull(const bool &force = false).

Definition at line 546 of file TestPointer.cpp.

7.49.2.28 void ParCompMarkTest::TestPointer::test_trylock () [protected]

Tests method Pointer::trylock().

Definition at line 589 of file TestPointer.cpp.

7.49.2.29 void ParCompMarkTest::TestPointer::test_unlock () [protected]

Tests method Pointer::unlock().

Definition at line 603 of file TestPointer.cpp.

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

- **TestPointer.h**
- **TestPointer.cpp**

7.50 ParCompMarkTest::TestProcess Class Reference

`#include <TestProcess.h>`

7.50.1 Detailed Description

Remarks:

Test suite for **ParCompMark::Process**(p. 182).

This is a `CppUnit` test suite for class **ParCompMark::Process**(p. 182) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestProcess.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor_cstd__string_Node_p** ()
- void **test_destructor** ()
- void **test_init** ()
- void **test_initialize** ()
- void **test_finalize** ()
- void **test_openRenderWindow_cstd__string_cstd__string_cbool_cu32_cs32_cs32_cu32_cu32_cu32** ()
- void **test_initProcess** ()
- void **test_runningProcess_cu32_cReal** ()
- void **test_task** ()

7.50.2 Member Function Documentation

7.50.2.1 void ParCompMarkTest::TestProcess::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestProcess.cpp.

7.50.2.2 void ParCompMarkTest::TestProcess::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestProcess.cpp.

7.50.2.3 void ParCompMarkTest::TestProcess::test_constructor_cstd__string_Node_p () [protected]

Tests constructor Process::Process(const std::string &name, Node * parent).

Definition at line 46 of file TestProcess.cpp.

7.50.2.4 void ParCompMarkTest::TestProcess::test_destructor () [protected]

Tests destructor Process::~Process().

Definition at line 60 of file TestProcess.cpp.

7.50.2.5 void ParCompMarkTest::TestProcess::test_finalize () [protected]

Tests method Process::finalize().

Definition at line 100 of file TestProcess.cpp.

7.50.2.6 void ParCompMarkTest::TestProcess::test_init () [protected]

Tests method Process::init().

Definition at line 72 of file TestProcess.cpp.

7.50.2.7 void ParCompMarkTest::TestProcess::test_initialize () [protected]

Tests method Process::initialize().

Definition at line 86 of file TestProcess.cpp.

7.50.2.8 void ParCompMarkTest::TestProcess::test_initProcess () [protected]

Tests method Process::initProcess().

Definition at line 128 of file TestProcess.cpp.

7.50.2.9 void ParCompMarkTest::TestProcess::test_openRenderWindow_cstd__string_cstd__string_cbool_cu32_cs32_cs32_cu32_cu32_cu32 () [protected]

Tests method Process::openRenderWindow(const std::string caption = PCM Framework, const std::string &displayName = , const bool &fullScreen = true, const u32 &colourDepth = 0, const s32 &left = GLXRenderWindow::CENTERED, const s32 &top = GLXRenderWindow::CENTERED, const u32 &width = GLXRenderWindow::MAXIMALSIZE, const u32 &height = GLXRenderWindow::MAXIMALSIZE, const u32 &fsaaSamples = 0).

Definition at line 114 of file TestProcess.cpp.

7.50.2.10 void ParCompMarkTest::TestProcess::test_runningProcess_cu32_cReal () [protected]

Tests method Process::runningProcess(const u32 &frameNumber, const Real &time).

Definition at line 145 of file TestProcess.cpp.

7.50.2.11 void ParCompMarkTest::TestProcess::test_task () [protected]

Tests method Process::task().

Definition at line 162 of file TestProcess.cpp.

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

- **TestProcess.h**
- **TestProcess.cpp**

7.51 ParCompMarkTest::TestSingleton Class Reference

`#include <TestSingleton.h>`

7.51.1 Detailed Description

Remarks:

Test suite for **ParCompMark::Singleton**(p. 194).

This is a `CppUnit` test suite for class **ParCompMark::Singleton**(p. 194) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestSingleton.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor** ()
- void **test_destructor** ()
- void **test_getInstance** ()
- void **test_createInstance** ()
- void **test_destroyInstance** ()

7.51.2 Member Function Documentation

7.51.2.1 void ParCompMarkTest::TestSingleton::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestSingleton.cpp.

7.51.2.2 void ParCompMarkTest::TestSingleton::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestSingleton.cpp.

7.51.2.3 void ParCompMarkTest::TestSingleton::test_constructor () [protected]

Tests constructor Singleton::Singleton().

Definition at line 46 of file TestSingleton.cpp.

7.51.2.4 void ParCompMarkTest::TestSingleton::test_createInstance () [protected]

Tests method Singleton::createInstance().

Definition at line 78 of file TestSingleton.cpp.

7.51.2.5 void ParCompMarkTest::TestSingleton::test_destroyInstance () [protected]

Tests method Singleton::destroyInstance().

Definition at line 92 of file TestSingleton.cpp.

7.51.2.6 void ParCompMarkTest::TestSingleton::test_destructor () [protected]

Tests destructor Singleton::∼Singleton().

Definition at line 53 of file TestSingleton.cpp.

7.51.2.7 void ParCompMarkTest::TestSingleton::test_getInstance () [protected]

Tests method Singleton::getInstance().

Definition at line 64 of file TestSingleton.cpp.

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

- **TestSingleton.h**
- **TestSingleton.cpp**

7.52 ParCompMarkTest::TestSqVM Class Reference

`#include <TestSqVM.h>`

7.52.1 Detailed Description

Remarks:

Test suite for **ParCompMark::SqVM**(p. 196).

This is a `CppUnit` test suite for class **ParCompMark::SqVM**(p. 196) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestSqVM.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_printFunction___HSQUIRRELVM_cSQChar_p_** ()

7.52.2 Member Function Documentation

7.52.2.1 void ParCompMarkTest::TestSqVM::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestSqVM.cpp.

7.52.2.2 void ParCompMarkTest::TestSqVM::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestSqVM.cpp.

7.52.2.3 void ParCompMarkTest::TestSqVM::test_printFunction___HSQUIRRELVM_cSQChar_p_ () [protected]

Tests method SqVM::printFunction(HSQUIRRELVM sqVM, const SQChar * s, ...).

Definition at line 46 of file TestSqVM.cpp.

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

- **TestSqVM.h**
- **TestSqVM.cpp**

7.53 ParCompMarkTest::TestThread Class Reference

`#include <TestThread.h>`

7.53.1 Detailed Description

Remarks:

Test suite for **ParCompMark::Thread**(p. 264).

This is a `CppUnit` test suite for class **ParCompMark::Thread**(p. 264) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestThread.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor_cstd__string** ()
- void **test_destructor** ()
- void **test_yield** ()
- void **test_getUSTime** ()
- void **test_entryPoint_void_p** ()
- void **test_initThread_cu32_cu32_cbool_cbool** ()
- void **test_initialize** ()
- void **test_finalize** ()
- void **test_startThread** ()
- void **test_joinThread** ()
- void **test_shutDownThread** ()
- void **test_stopThread** ()
- void **test_thread** ()
- void **test_iteration** ()
- void **test_task** ()

7.53.2 Member Function Documentation

7.53.2.1 void ParCompMarkTest::TestThread::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestThread.cpp.

7.53.2.2 void ParCompMarkTest::TestThread::tearDown ()

Cleans up after the test run.

Definition at line 39 of file TestThread.cpp.

7.53.2.3 void ParCompMarkTest::TestThread::test_constructor_cstd__string () [protected]

Tests constructor Thread::Thread(const std::string &name).

Definition at line 49 of file TestThread.cpp.

7.53.2.4 void ParCompMarkTest::TestThread::test_destructor () [protected]

Tests destructor Thread::∼Thread().

Definition at line 66 of file TestThread.cpp.

References ParCompMark::Thread::initThread().

Here is the call graph for this function:

7.53.2.5 void ParCompMarkTest::TestThread::test_entryPoint_void_p () [protected]

Tests method Thread::entryPoint(void ∗ arg).

Definition at line 131 of file TestThread.cpp.

7.53.2.6 void ParCompMarkTest::TestThread::test_finalize () [protected]

Tests method Thread::finalize().

Definition at line 187 of file TestThread.cpp.

7.53.2.7 void ParCompMarkTest::TestThread::test_getUSTime () [protected]

Tests method Thread::getUSTime().

Definition at line 107 of file TestThread.cpp.

7.53.2.8 void ParCompMarkTest::TestThread::test_initialize () [protected]

Tests method Thread::initialize().

Definition at line 173 of file TestThread.cpp.

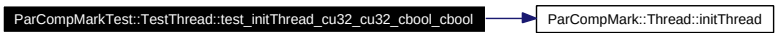
7.53.2.9 void ParCompMarkTest::TestThread::test_initThread_cu32_cu32_cbool_cbool () [protected]

Tests method Thread::initThread(const u32 &iterationNumber = 0, const u32 &expectedFPS = 0, const bool &joinable = false, const bool &waitThread = false).

Definition at line 149 of file TestThread.cpp.

References ParCompMark::Thread::initThread(), ParCompMark::Thread::mExpectedFPS, and ParCompMark::Thread::mIterationNumber.

Here is the call graph for this function:



7.53.2.10 void ParCompMarkTest::TestThread::test_iteration () [protected]

Tests method Thread::iteration().

Definition at line 340 of file TestThread.cpp.

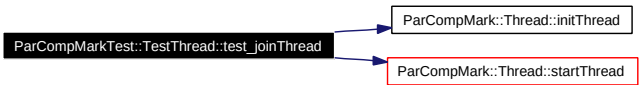
7.53.2.11 void ParCompMarkTest::TestThread::test_joinThread () [protected]

Tests method Thread::joinThread().

Definition at line 239 of file TestThread.cpp.

References ParCompMark::Thread::initThread(), and ParCompMark::Thread::startThread().

Here is the call graph for this function:



7.53.2.12 void ParCompMarkTest::TestThread::test_shutDownThread () [protected]

Tests method Thread::shutDownThread().

Definition at line 290 of file TestThread.cpp.

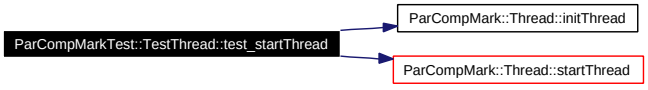
7.53.2.13 void ParCompMarkTest::TestThread::test_startThread () [protected]

Tests method Thread::startThread().

Definition at line 201 of file TestThread.cpp.

References ParCompMark::Thread::initThread(), and ParCompMark::Thread::startThread().

Here is the call graph for this function:



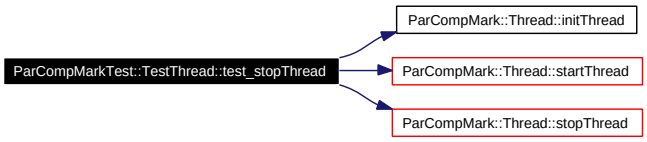
7.53.2.14 void ParCompMarkTest::TestThread::test_stopThread () [protected]

Tests method Thread::stopThread().

Definition at line 304 of file TestThread.cpp.

References ParCompMark::Thread::initThread(), ParCompMark::Thread::startThread(), and ParCompMark::Thread::stopThread().

Here is the call graph for this function:



7.53.2.15 void ParCompMarkTest::TestThread::test_task () [protected]

Tests method Thread::task().

Definition at line 354 of file TestThread.cpp.

7.53.2.16 void ParCompMarkTest::TestThread::test_thread () [protected]

Tests method Thread::thread().

Definition at line 326 of file TestThread.cpp.

7.53.2.17 void ParCompMarkTest::TestThread::test_yield () [protected]

Tests method Thread::yield().

Definition at line 93 of file TestThread.cpp.

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

- **TestThread.h**
- **TestThread.cpp**

7.54 ParCompMarkTest::TestTimer Class Reference

`#include <TestTimer.h>`

7.54.1 Detailed Description

Remarks:

Test suite for **ParCompMark::Timer**(p. 275).

This is a `CppUnit` test suite for class **ParCompMark::Timer**(p. 275) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestTimer.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_getSystemTime** ()

7.54.2 Member Function Documentation

7.54.2.1 void ParCompMarkTest::TestTimer::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestTimer.cpp.

7.54.2.2 void ParCompMarkTest::TestTimer::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestTimer.cpp.

7.54.2.3 void ParCompMarkTest::TestTimer::test_getSystemTime () [protected]

Tests method Timer::getSystemTime().

Definition at line 46 of file TestTimer.cpp.

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

- **TestTimer.h**
- **TestTimer.cpp**

7.55 ParCompMarkTest::TestXDisplay Class Reference

`#include <TestXDisplay.h>`

7.55.1 Detailed Description

Remarks:

Test suite for **ParCompMark::XDisplay**(p. 276).

This is a `CppUnit` test suite for class **ParCompMark::XDisplay**(p. 276) of project **ParCompMark**(p. 13).

Definition at line 57 of file TestXDisplay.h.

Public Member Functions

- void **setUp** ()
- void **tearDown** ()

Protected Member Functions

- void **test_constructor_cstd__string** ()
- void **test_destructor** ()
- void **test_initialize** ()
- void **test_finalize** ()
- void **test_findBestVisual_cs32_cs32** ()
- void **test_getVisualAttribs_XVisualInfo_p_VisualAttribs** ()

7.55.2 Member Function Documentation

7.55.2.1 void ParCompMarkTest::TestXDisplay::setUp ()

Sets up context before running a test.

Definition at line 30 of file TestXDisplay.cpp.

7.55.2.2 void ParCompMarkTest::TestXDisplay::tearDown ()

Cleans up after the test run.

Definition at line 36 of file TestXDisplay.cpp.

7.55.2.3 void ParCompMarkTest::TestXDisplay::test_constructor_cstd__string () [protected]

Tests constructor XDisplay::XDisplay(const std::string &displayName =).

Definition at line 46 of file TestXDisplay.cpp.

7.55.2.4 void ParCompMarkTest::TestXDisplay::test_destructor () [protected]

Tests destructor XDisplay::∼XDisplay().

Definition at line 60 of file TestXDisplay.cpp.

7.55.2.5 void ParCompMarkTest::TestXDisplay::test_finalize () [protected]

Tests method XDisplay::finalize().

Definition at line 86 of file TestXDisplay.cpp.

7.55.2.6 void ParCompMarkTest::TestXDisplay::test_findBestVisual_cs32_cs32 () [protected]

Tests method XDisplay::findBestVisual(const s32 &screenNumber, const s32 &multiSample = XDisplay::IGNOREMULTISAMPLE).

Definition at line 100 of file TestXDisplay.cpp.

7.55.2.7 void ParCompMarkTest::TestXDisplay::test_getVisualAttribs_XVisualInfo_p_VisualAttribs () [protected]

Tests method XDisplay::getVisualAttribs(XVisualInfo ∗ vInfo, VisualAttribs &attribs).

Definition at line 116 of file TestXDisplay.cpp.

7.55.2.8 void ParCompMarkTest::TestXDisplay::test_initialize () [protected]

Tests method XDisplay::initialize().

Definition at line 72 of file TestXDisplay.cpp.

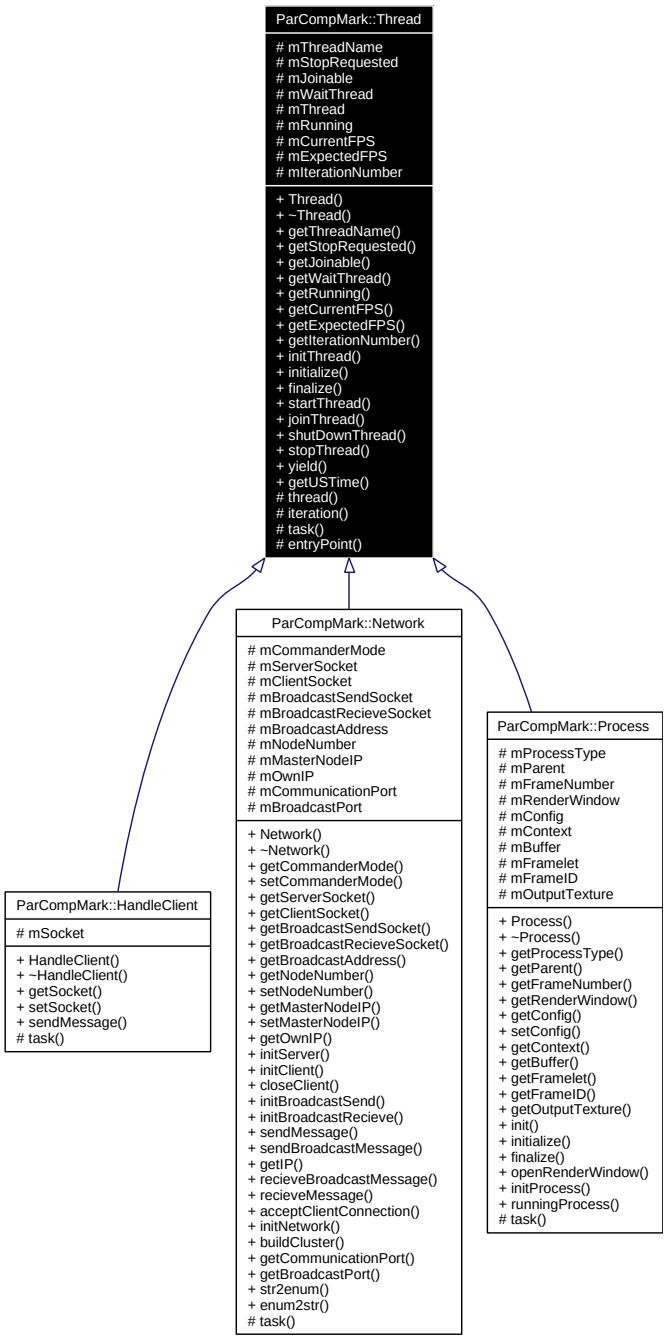
The documentation for this class was generated from the following files:

- **TestXDisplay.h**
- **TestXDisplay.cpp**

7.56 ParCompMark::Thread Class Reference

`#include <PCMThread.h>`

Inheritance diagram for ParCompMark::Thread:



7.56.1 Detailed Description

This is a unique thread class.

Definition at line 53 of file PCMThread.h.

Public Member Functions

- **Thread** (const std::string &name)
- virtual ~**Thread** ()
- const std::string & **getThreadName** () const
- const bool & **getStopRequested** () const
- const bool & **getJoinable** () const
- const bool & **getWaitThread** () const
- const bool & **getRunning** () const
- const **u32** & **getCurrentFPS** () const
- const **u32** & **getExpectedFPS** () const
- const **u32** & **getIterationNumber** () const
- virtual void **initThread** (const **u32** &iterationNumber=0, const **u32** &expectedFPS=0, const bool &joinable=false, const bool &waitThread=false)
- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual void **startThread** ()
- virtual void **joinThread** ()
- virtual void **shutDownThread** ()
- virtual void **stopThread** ()

Static Public Member Functions

- static void **yield** ()
- static **Real getUSTime** ()

Protected Member Functions

- virtual void **thread** ()
- virtual bool **iteration** ()
- virtual void **task** ()

Static Protected Member Functions

- static void * **entryPoint** (void *arg)

Protected Attributes

- std::string **mThreadName**
- bool **mStopRequested**
- bool **mJoinable**
- bool **mWaitThread**
- pthread_t **mThread**
- bool **mRunning**
- **u32 mCurrentFPS**
- **u32 mExpectedFPS**
- **u32 mIterationNumber**

7.56.2 Constructor & Destructor Documentation

7.56.2.1 ParCompMark::Thread::Thread (const std::string & *name*)

Thread(p. 264) constructor.

Parameters:
← ***name*** **Name**(p. 131) of the thread.

Definition at line 38 of file PCMThread.cpp.

References mCurrentFPS, mExpectedFPS, mIterationNumber, mJoinable, mRunning, mStopRequested, mThread, and mThreadName.

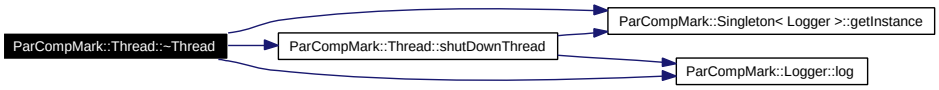
7.56.2.2 ParCompMark::Thread::~Thread () [virtual]

The destructor. This class has virtual destructor.

Definition at line 63 of file PCMThread.cpp.

References ParCompMark::Logger::DEBUG, ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Logger::log(), mRunning, mThreadName, and shutDownThread().

Here is the call graph for this function:



7.56.3 Member Function Documentation

7.56.3.1 void * ParCompMark::Thread::entryPoint (void * *arg*) [static, protected]

Static entry point for the thread.

Parameters:
← ***arg*** This is only for thread.

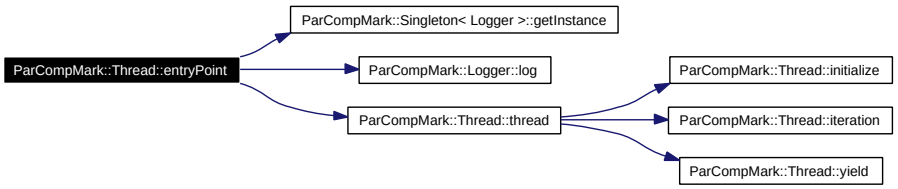
Returns:
This is only for thread.

Definition at line 77 of file PCMThread.cpp.

References ParCompMark::Logger::DEBUG, Except, ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Logger::log(), mRunning, mThreadName, and thread().

Referenced by startThread().

Here is the call graph for this function:



7.56.3.2 void ParCompMark::Thread::finalize () [virtual]

Some finitiate step as thread.

Reimplemented in **ParCompMark::Process** (p. 185).

Definition at line 128 of file PCMThread.cpp.

7.56.3.3 const u32 & ParCompMark::Thread::getCurrentFPS () const [inline]

Remarks:
Getter of mCurrentFPS.

Returns value of mCurrentFPS.

Returns:
The value of mCurrentFPS

Definition at line 412 of file PCMThread.h.

7.56.3.4 const u32 & ParCompMark::Thread::getExpectedFPS () const [inline]

Remarks:
Getter of mExpectedFPS.

Returns value of mExpectedFPS.

Returns:
The value of mExpectedFPS

Definition at line 419 of file PCMThread.h.

7.56.3.5 const u32 & ParCompMark::Thread::getIterationNumber () const [inline]

Remarks:
Getter of mIterationNumber.

Returns value of mIterationNumber.

Returns:
The value of mIterationNumber

Definition at line 426 of file PCMThread.h.

7.56.3.6 const bool & ParCompMark::Thread::getJoinable () const [inline]

Remarks:
Getter of mJoinable.

Returns value of mJoinable.

Returns:
The value of mJoinable

Definition at line 391 of file PCMThread.h.

7.56.3.7 const bool & ParCompMark::Thread::getRunning () const [inline]

Remarks:
Getter of mRunning.

Returns value of mRunning.

Returns:
The value of mRunning

Definition at line 405 of file PCMThread.h.

7.56.3.8 const bool & ParCompMark::Thread::getStopRequested () const [inline]

Remarks:
Getter of mStopRequested.

Returns value of mStopRequested.

Returns:
The value of mStopRequested

Definition at line 384 of file PCMThread.h.

7.56.3.9 const std::string & ParCompMark::Thread::getThreadName () const [inline]

Remarks:
Getter of mThreadName.

Returns value of mThreadName.

Returns:
The value of mThreadName

Definition at line 377 of file PCMThread.h.

7.56.3.10 Real ParCompMark::Thread::getUSTime () [inline, static]

Get system time in us.

Returns:
System time in us.

Definition at line 444 of file PCMThread.h.

7.56.3.11 const bool & ParCompMark::Thread::getWaitThread () const [inline]

Remarks:
Getter of mWaitThread.

Returns value of mWaitThread.

Returns:
The value of mWaitThread

Definition at line 398 of file PCMThread.h.

7.56.3.12 void ParCompMark::Thread::initialize () [virtual]

Some init step as thread.

Reimplemented in **ParCompMark::Process** (p. 188).

Definition at line 122 of file PCMThread.cpp.

Referenced by thread().

7.56.3.13 void ParCompMark::Thread::initThread (const u32 & *iterationNumber* = 0, const u32 & *expectedFPS* = 0, const bool & *joinable* = false, const bool & *waitThread* = false) [virtual]

Start thread.

Parameters:
← ***iterationNumber*** How much is the task running. Zero means nan.
← ***expectedFPS*** Expected FPS.
← ***joinable*** Joinable?
← ***waitThread*** Wait when destroy class?

Definition at line 108 of file PCMThread.cpp.

References Assert, mExpectedFPS, mIterationNumber, mJoinable, and mWaitThread.

Referenced by ParCompMark::Network::acceptClientConnection(), ParCompMark::Network::initBroadcastRecieve(), ParCompMark::Network::initServer(), ParCompMarkTest::TestThread::test_destructor(), ParCompMarkTest::TestNetwork::test_initBroadcastSend(), ParCompMarkTest::TestThread::test_initThread_cu32_cu32_cbool_cbool(), ParCompMarkTest::TestThread::test_joinThread(), ParCompMarkTest::TestThread::test_startThread(), and ParCompMarkTest::TestThread::test_stopThread().

7.56.3.14 bool ParCompMark::Thread::iteration () [inline, protected, virtual]

One iteration step. This method calls the overridable task method.

Returns:
If true, the thread is stopped.

Definition at line 459 of file PCMThread.h.

Referenced by thread().

7.56.3.15 void ParCompMark::Thread::joinThread () [virtual]

Join thread. Wait for its ending.

Remarks:
!!! You can join if the thread will stop (mIterationNumber != 0 or mStopRequested = true).

Definition at line 164 of file PCMThread.cpp.

References Assert, Except, mIterationNumber, mJoinable, mRunning, mStopRequested, mThread, and mThreadName.

Referenced by ParCompMarkTest::TestNetwork::test_initBroadcastSend().

7.56.3.16 void ParCompMark::Thread::shutDownThread () [virtual]

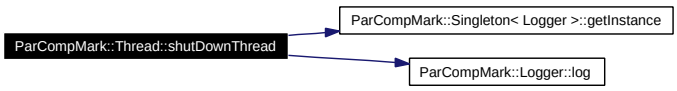
Immediately stop the thread.

Definition at line 185 of file PCMThread.cpp.

References Assert, ParCompMark::Logger::DEBUG, Except, ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Logger::log(), mRunning, mStopRequested, mThread, mThreadName, and mWaitThread.

Referenced by ParCompMark::Network::initServer(), and ~Thread().

Here is the call graph for this function:



7.56.3.17 void ParCompMark::Thread::startThread () [virtual]

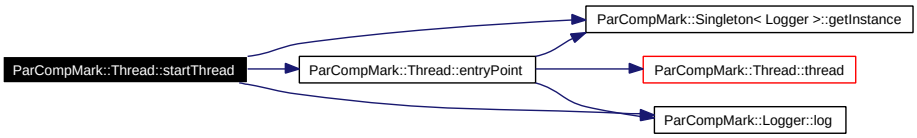
Start thread.

Definition at line 134 of file PCMThread.cpp.

References Assert, ParCompMark::Logger::DEBUG, entryPoint(), ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Logger::log(), mJoinable, mRunning, mThread, and mThreadName.

Referenced by ParCompMark::Network::acceptClientConnection(), ParCompMark::Network::initBroadcastRecieve(), ParCompMark::Network::initServer(), ParCompMarkTest::TestNetwork::test_initBroadcastSend(), ParCompMarkTest::TestThread::test_joinThread(), ParCompMarkTest::TestThread::test_startThread(), and ParCompMarkTest::TestThread::test_stopThread().

Here is the call graph for this function:



7.56.3.18 void ParCompMark::Thread::stopThread () [virtual]

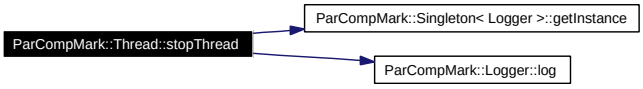
Request thread shut down.

Definition at line 216 of file PCMThread.cpp.

References ParCompMark::Logger::DEBUG, ParCompMark::Singleton< Logger >::getInstance(), ParCompMark::Logger::log(), mStopRequested, and mThreadName.

Referenced by ParCompMarkTest::TestThread::test_stopThread().

Here is the call graph for this function:



7.56.3.19 void ParCompMark::Thread::task () [protected, virtual]

It is the task of the thread. It have to be overiden.

Remarks:
This method should be abstact. It is not, because of unit testing of this class.

Reimplemented in **ParCompMark::HandleClient** (p. 110), **ParCompMark::Network** (p. 146), and **ParCompMark::Process** (p. 190).

Definition at line 253 of file PCMThread.cpp.

7.56.3.20 void ParCompMark::Thread::thread () [protected, virtual]

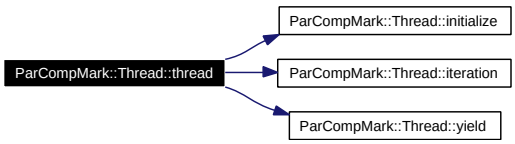
This is the thread method containing one or more (or inifinite) iterations.

Definition at line 224 of file PCMThread.cpp.

References initialize(), iteration(), mIterationNumber, and yield().

Referenced by entryPoint().

Here is the call graph for this function:



7.56.3.21 void ParCompMark::Thread::yield () [inline, static]

Yield current thread.

Definition at line 437 of file PCMThread.h.

Referenced by thread().

7.56.4 Member Data Documentation

7.56.4.1 u32 ParCompMark::Thread::mCurrentFPS [protected]

The thread is running.

Remarks:
This is own attribute of this class.

Definition at line 116 of file PCMThread.h.

Referenced by Thread().

7.56.4.2 u32 ParCompMark::Thread::mExpectedFPS [protected]

The thread is running. Zero means no limit.

Remarks:
This is own attribute of this class.

Definition at line 123 of file PCMThread.h.

Referenced by initThread(), ParCompMarkTest::TestThread::test_initThread_cu32_cu32_cbool_cbool(), and Thread().

7.56.4.3 u32 ParCompMark::Thread::mIterationNumber [protected]

Iteration number of task. Zero means nan.

Remarks:
This is own attribute of this class.

Definition at line 130 of file PCMThread.h.

Referenced by initThread(), joinThread(), ParCompMarkTest::TestThread::test_initThread_cu32_cu32_-cbool_cbool(), thread(), and Thread().

7.56.4.4 bool ParCompMark::Thread::mJoinable [protected]

The thread is created as joinable or not.

Remarks:
This is own attribute of this class.

Definition at line 88 of file PCMThread.h.

Referenced by initThread(), joinThread(), startThread(), and Thread().

7.56.4.5 bool ParCompMark::Thread::mRunning [protected]

The thread is running.

Remarks:
This is own attribute of this class.

Definition at line 109 of file PCMThread.h.

Referenced by entryPoint(), ParCompMark::Network::initServer(), joinThread(), shutDownThread(), startThread(), Thread(), and ∼Thread().

7.56.4.6 bool ParCompMark::Thread::mStopRequested [protected]

The thread is requested to stop.

Remarks:
This is own attribute of this class.

Definition at line 81 of file PCMThread.h.

Referenced by joinThread(), shutDownThread(), stopThread(), and Thread().

7.56.4.7 pthread_t ParCompMark::Thread::mThread [protected]

Thread(p. 264) handle.

Remarks:
This is own attribute of this class.

Definition at line 102 of file PCMThread.h.

Referenced by joinThread(), shutDownThread(), startThread(), and Thread().

7.56.4.8 std::string ParCompMark::Thread::mThreadName [protected]

Name(p. 131) of the thread.

Remarks:
This is own attribute of this class.

Definition at line 74 of file PCMThread.h.

Referenced by entryPoint(), joinThread(), shutDownThread(), startThread(), stopThread(), Thread(), and ∼Thread().

7.56.4.9 bool ParCompMark::Thread::mWaitThread [protected]

When class destructor call, wait thread or not.

Remarks:
This is own attribute of this class.

Definition at line 95 of file PCMThread.h.

Referenced by initThread(), and shutDownThread().

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

- **PCMThread.h**
- **PCMThread.cpp**

7.57 ParCompMark::Timer Class Reference

`#include <PCMTimer.h>`

7.57.1 Detailed Description

Collection of time handling methods.

Definition at line 47 of file PCMTimer.h.

Static Public Member Functions

- static **Real getSystemTime** ()

Static Public Attributes

- static const **Real EPSILONDELAY** = 0.000001

7.57.2 Member Function Documentation

7.57.2.1 Real ParCompMark::Timer::getSystemTime () [inline, static]

Return time in secondes.

Returns:
Time in seconds

Definition at line 116 of file PCMTimer.h.

Referenced by ParCompMark::GLXRenderWindow::resetStatistics(), and ParCompMark::GLXRenderWindow::updateStatistics().

7.57.3 Member Data Documentation

7.57.3.1 const Real ParCompMark::Timer::EPSILONDELAY = 0.000001 [static]

Constant for minimal time delay (1us).

Remarks:
This is own attribute of this class.

Definition at line 68 of file PCMTimer.h.

Referenced by ParCompMark::GLXRenderWindow::updateStatistics().

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

- **PCMTimer.h**
- **PCMTimer.cpp**

7.58 ParCompMark::XDisplay Class Reference

`#include <PCMXDisplay.h>`

7.58.1 Detailed Description

Class that encapsulates an X Display.

Definition at line 55 of file PCMXDisplay.h.

Public Types

- typedef **Pointer**< **XDisplay**, **DummyLock** > **Pointer**

Public Member Functions

- **XDisplay** (const std::string &displayName="")
- virtual ~**XDisplay** ()
- const std::string & **getDisplayName** () const
- ::Display * **getDisplay** () const
- const bool & **getInitialized** () const
- virtual void **initialize** ()
- virtual void **finalize** ()
- virtual **s32** **findBestVisual** (const **s32** &screenNumber, const **s32** &multiSample=**XDisplay::IGNOREMULTISAMPLE**)
- virtual void **getVisualAttribs** (XVisualInfo *vInfo, **VisualAttribs** &attribs)

Static Public Attributes

- static const **s32 IGNOREMULTISAMPLE** = -1

Protected Attributes

- std::string **mDisplayName**
- ::Display * **mDisplay**
- bool **mInitialized**

Classes

- struct **VisualAttribs**

7.58.2 Member Typedef Documentation

7.58.2.1 typedef Pointer< XDisplay, DummyLock > ParCompMark::XDisplay::Pointer

Type for pointer on this class.

Definition at line 73 of file PCMXDisplay.h.

7.58.3 Constructor & Destructor Documentation

7.58.3.1 ParCompMark::XDisplay::XDisplay (const std::string & *displayName* = "")

Create an X display.

Parameters:

← ***displayName*** X display name

Definition at line 44 of file PCMXDisplay.cpp.

References mDisplay, mDisplayName, and mInitialized.

7.58.3.2 ParCompMark::XDisplay::~XDisplay () [virtual]

The destructor. This class has virtual destructor.

Definition at line 57 of file PCMXDisplay.cpp.

References finalize(), and mInitialized.

Here is the call graph for this function:

7.58.4 Member Function Documentation

7.58.4.1 void ParCompMark::XDisplay::finalize () [virtual]

Finalize X display.

Definition at line 83 of file PCMXDisplay.cpp.

References Assert, mDisplay, and mInitialized.

Referenced by ~XDisplay().

7.58.4.2 s32 ParCompMark::XDisplay::findBestVisual (const s32 & *screenNumber*, const s32 & *multiSample* = XDisplay::IGNOREMULTISAMPLE) [virtual]

Examine all visuals to find the so-called best one. We prefer deepest RGBA buffer with depth, stencil and accum that has no caveats. This will only choose formats with a multisample that equals multisample.

Parameters:

← ***screenNumber*** Screen number to test

← ***multiSample*** Select specific multisample value

Returns:

-1 in case of failure, otherwise a valid visual ID

Definition at line 97 of file PCMXDisplay.cpp.

References ParCompMark::XDisplay::VisualAttribs::accumRedSize, ParCompMark::XDisplay::VisualAttribs::alphaSize, ParCompMark::XDisplay::VisualAttribs::blueSize, ParCompMark::XDisplay::VisualAttribs::depthSize, ParCompMark::XDisplay::VisualAttribs::doubleBuffer, getVisualAttribs(), ParCompMark::XDisplay::VisualAttribs::greenSize, mDisplay, ParCompMark::XDisplay::VisualAttribs::redSize, ParCompMark::XDisplay::VisualAttribs::rgba, ParCompMark::XDisplay::VisualAttribs::stencilSize, ParCompMark::XDisplay::VisualAttribs::supportsGL, and ParCompMark::XDisplay::VisualAttribs::visualCaveat.

Here is the call graph for this function:

7.58.4.3 inline::Display * ParCompMark::XDisplay::getDisplay () const

Remarks:
Getter of mDisplay.

Returns value of mDisplay.

Returns:
The value of mDisplay

Definition at line 287 of file PCMXDisplay.h.

7.58.4.4 const std::string & ParCompMark::XDisplay::getDisplayName () const [inline]

Remarks:
Getter of mDisplayName.

Returns value of mDisplayName.

Returns:
The value of mDisplayName

Definition at line 280 of file PCMXDisplay.h.

7.58.4.5 const bool & ParCompMark::XDisplay::getInitialized () const [inline]

Remarks:
Getter of mInitialized.

Returns value of mInitialized.

Returns:
The value of mInitialized

Definition at line 294 of file PCMXDisplay.h.

7.58.4.6 void ParCompMark::XDisplay::getVisualAttribs (XVisualInfo * *vInfo*, VisualAttribs & *attribs*) [virtual]

Get visual attributes for the specified visual info descriptor

Parameters:

← ***vInfo*** Visual info descriptor

→ ***attribs*** Variable to hold the restrieved attributes

Definition at line 161 of file PCMXDisplay.cpp.

References ParCompMark::XDisplay::VisualAttribs::accumAlphaSize, ParCompMark::XDisplay::VisualAttribs::accumBlueSize, ParCompMark::XDisplay::VisualAttribs::accumGreenSize, ParCompMark::XDisplay::VisualAttribs::accumRedSize, ParCompMark::XDisplay::VisualAttribs::alphaSize, ParCompMark::XDisplay::VisualAttribs::auxBuffers, ParCompMark::XDisplay::VisualAttribs::bitsPerRGB, ParCompMark::XDisplay::VisualAttribs::blueMask, ParCompMark::XDisplay::VisualAttribs::blueSize, ParCompMark::XDisplay::VisualAttribs::bufferSize, ParCompMark::XDisplay::VisualAttribs::colormapSize, ParCompMark::XDisplay::VisualAttribs::depth, ParCompMark::XDisplay::VisualAttribs::depthSize, ParCompMark::XDisplay::VisualAttribs::doubleBuffer, ParCompMark::XDisplay::VisualAttribs::greenMask, ParCompMark::XDisplay::VisualAttribs::greenSize, ParCompMark::XDisplay::VisualAttribs::id, ParCompMark::XDisplay::VisualAttribs::klass, ParCompMark::XDisplay::VisualAttribs::level, mDisplay, ParCompMark::XDisplay::VisualAttribs::numMultisample, ParCompMark::XDisplay::VisualAttribs::numSamples, ParCompMark::XDisplay::VisualAttribs::redMask, ParCompMark::XDisplay::VisualAttribs::redSize, ParCompMark::XDisplay::VisualAttribs::rgba, ParCompMark::XDisplay::VisualAttribs::stencilSize, ParCompMark::XDisplay::VisualAttribs::stereo, ParCompMark::XDisplay::VisualAttribs::supportsGL, ParCompMark::XDisplay::VisualAttribs::transparentAlphaValue, ParCompMark::XDisplay::VisualAttribs::transparentBlueValue, ParCompMark::XDisplay::VisualAttribs::transparentGreenValue, ParCompMark::XDisplay::VisualAttribs::transparentIndexValue, ParCompMark::XDisplay::VisualAttribs::transparentRedValue, ParCompMark::XDisplay::VisualAttribs::transparentType, and ParCompMark::XDisplay::VisualAttribs::visualCaveat.

Referenced by findBestVisual().

7.58.4.7 void ParCompMark::XDisplay::initialize () [virtual]

Initialize X display.

Definition at line 69 of file PCMXDisplay.cpp.

References Assert, mDisplay, mDisplayName, and mInitialized.

7.58.5 Member Data Documentation

7.58.5.1 const s32 ParCompMark::XDisplay::IGNOREMULTISAMPLE = -1 [static]

Constant for ignoring multisample.

Remarks:

This is own attribute of this class.

Definition at line 127 of file PCMXDisplay.h.

7.58.5.2 ::Display* ParCompMark::XDisplay::mDisplay [protected]

Wrapped X Display.

Remarks:
This attribute references an attribute.

Definition at line 147 of file PCMXDisplay.h.

Referenced by finalize(), findBestVisual(), getVisualAttribs(), initialize(), and XDisplay().

7.58.5.3 std::string ParCompMark::XDisplay::mDisplayName [protected]

X display name.

Remarks:
This is own attribute of this class.

Definition at line 140 of file PCMXDisplay.h.

Referenced by initialize(), and XDisplay().

7.58.5.4 bool ParCompMark::XDisplay::mInitialized [protected]

The display is initialized.

Remarks:
This is own attribute of this class.

Definition at line 154 of file PCMXDisplay.h.

Referenced by finalize(), initialize(), XDisplay(), and ~XDisplay().

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

- **PCMXDisplay.h**
- **PCMXDisplay.cpp**

7.59 ParCompMark::XDisplay::VisualAttribs Struct Reference

`#include <PCMXDisplay.h>`

7.59.1 Detailed Description

Struct for visual attributes.

Definition at line 78 of file PCMXDisplay.h.

Public Attributes

- int **id**
- int **klass**
- int **depth**
- int **redMask**
- int **greenMask**
- int **blueMask**
- int **colormapSize**
- int **bitsPerRGB**
- int **supportsGL**
- int **transparentType**
- int **transparentRedValue**
- int **transparentGreenValue**
- int **transparentBlueValue**
- int **transparentAlphaValue**
- int **transparentIndexValue**
- int **bufferSize**
- int **level**
- int **rgba**
- int **doubleBuffer**
- int **stereo**
- int **auxBuffers**
- int **redSize**
- int **greenSize**
- int **blueSize**
- int **alphaSize**
- int **depthSize**
- int **stencilSize**
- int **accumRedSize**
- int **accumGreenSize**
- int **accumBlueSize**
- int **accumAlphaSize**
- int **numSamples**
- int **numMultisample**
- int **visualCaveat**

7.59.2 Member Data Documentation

7.59.2.1 int ParCompMark::XDisplay::VisualAttribs::accumAlphaSize

Definition at line 107 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.2 int ParCompMark::XDisplay::VisualAttribs::accumBlueSize

Definition at line 107 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.3 int ParCompMark::XDisplay::VisualAttribs::accumGreenSize

Definition at line 107 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.4 int ParCompMark::XDisplay::VisualAttribs::accumRedSize

Definition at line 107 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::findBestVisual(), and ParCompMark::XDisplay::getVisualAttribs().

7.59.2.5 int ParCompMark::XDisplay::VisualAttribs::alphaSize

Definition at line 101 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::findBestVisual(), and ParCompMark::XDisplay::getVisualAttribs().

7.59.2.6 int ParCompMark::XDisplay::VisualAttribs::auxBuffers

Definition at line 100 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.7 int ParCompMark::XDisplay::VisualAttribs::bitsPerRGB

Definition at line 87 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.8 int ParCompMark::XDisplay::VisualAttribs::blueMask

Definition at line 83 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.9 int ParCompMark::XDisplay::VisualAttribs::blueSize

Definition at line 101 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::findBestVisual(), and ParCompMark::XDisplay::getVisualAttribs().

7.59.2.10 int ParCompMark::XDisplay::VisualAttribs::bufferSize

Definition at line 95 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.11 int ParCompMark::XDisplay::VisualAttribs::colormapSize

Definition at line 86 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.12 int ParCompMark::XDisplay::VisualAttribs::depth

Definition at line 82 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.13 int ParCompMark::XDisplay::VisualAttribs::depthSize

Definition at line 105 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::findBestVisual(), and ParCompMark::XDisplay::getVisualAttribs().

7.59.2.14 int ParCompMark::XDisplay::VisualAttribs::doubleBuffer

Definition at line 98 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::findBestVisual(), and ParCompMark::XDisplay::getVisualAttribs().

7.59.2.15 int ParCompMark::XDisplay::VisualAttribs::greenMask

Definition at line 83 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.16 int ParCompMark::XDisplay::VisualAttribs::greenSize

Definition at line 101 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::findBestVisual(), and ParCompMark::XDisplay::getVisualAttribs().

7.59.2.17 int ParCompMark::XDisplay::VisualAttribs::id

Definition at line 80 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.18 int ParCompMark::XDisplay::VisualAttribs::klass

Definition at line 81 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.19 int ParCompMark::XDisplay::VisualAttribs::level

Definition at line 96 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.20 int ParCompMark::XDisplay::VisualAttribs::numMultisample

Definition at line 111 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.21 int ParCompMark::XDisplay::VisualAttribs::numSamples

Definition at line 111 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.22 int ParCompMark::XDisplay::VisualAttribs::redMask

Definition at line 83 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.23 int ParCompMark::XDisplay::VisualAttribs::redSize

Definition at line 101 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::findBestVisual(), and ParCompMark::XDisplay::getVisualAttribs().

7.59.2.24 int ParCompMark::XDisplay::VisualAttribs::rgba

Definition at line 97 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::findBestVisual(), and ParCompMark::XDisplay::getVisualAttribs().

7.59.2.25 int ParCompMark::XDisplay::VisualAttribs::stencilSize

Definition at line 106 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::findBestVisual(), and ParCompMark::XDisplay::getVisualAttribs().

7.59.2.26 int ParCompMark::XDisplay::VisualAttribs::stereo

Definition at line 99 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.27 int ParCompMark::XDisplay::VisualAttribs::supportsGL

Definition at line 88 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::findBestVisual(), and ParCompMark::XDisplay::getVisualAttribs().

7.59.2.28 int ParCompMark::XDisplay::VisualAttribs::transparentAlphaValue

Definition at line 93 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.29 int ParCompMark::XDisplay::VisualAttribs::transparentBlueValue

Definition at line 92 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.30 int ParCompMark::XDisplay::VisualAttribs::transparentGreenValue

Definition at line 91 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.31 int ParCompMark::XDisplay::VisualAttribs::transparentIndexValue

Definition at line 94 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.32 int ParCompMark::XDisplay::VisualAttribs::transparentRedValue

Definition at line 90 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.33 int ParCompMark::XDisplay::VisualAttribs::transparentType

Definition at line 89 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::getVisualAttribs().

7.59.2.34 int ParCompMark::XDisplay::VisualAttribs::visualCaveat

Definition at line 113 of file PCMXDisplay.h.

Referenced by ParCompMark::XDisplay::findBestVisual(), and ParCompMark::XDisplay::getVisualAttribs().

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

- **PCMXDisplay.h**

Chapter 8

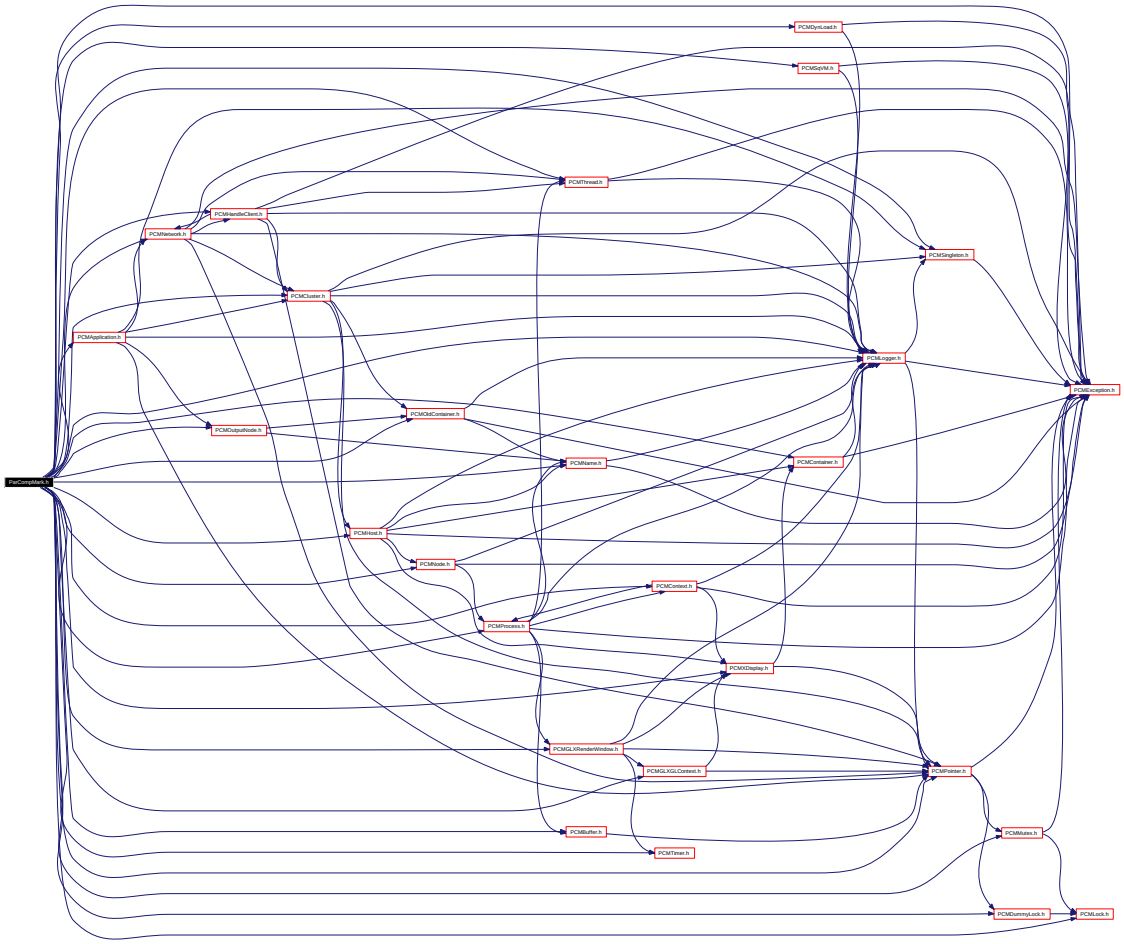
ParCompMark File Documentation

8.1 ParCompMark.h File Reference

```
#include "PCMApplication.h"
#include "PCMBuffer.h"
#include "PCMCluster.h"
#include "PCMContainer.h"
#include "PCMContext.h"
#include "PCMDummyLock.h"
#include "PCMDynLoad.h"
#include "PCMException.h"
#include "PCMGLXGLContext.h"
#include "PCMGLXRenderWindow.h"
#include "PCMHandleClient.h"
#include "PCMHost.h"
#include "PCMLock.h"
#include "PCMLogger.h"
#include "PCMMutex.h"
#include "PCMName.h"
#include "PCMNetwork.h"
#include "PCMNode.h"
#include "PCMOldContainer.h"
#include "PCMOutputNode.h"
#include "PCMPointer.h"
#include "PCMProcess.h"
#include "PCMSingleton.h"
#include "PCMSqVM.h"
```

```
#include "PCMThread.h"
#include "PCMTimer.h"
#include "PCMXDisplay.h"
```

Include dependency graph for ParCompMark.h:

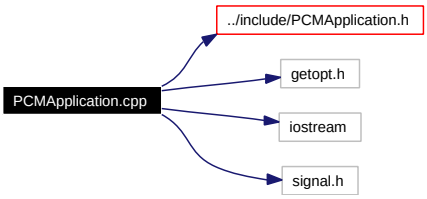


This graph shows which files directly or indirectly include this file:

8.2 PCMApplication.cpp File Reference

```
#include "../include/PCMApplication.h"
#include <getopt.h>
#include <iostream>
#include <signal.h>
```

Include dependency graph for PCMApplication.cpp:



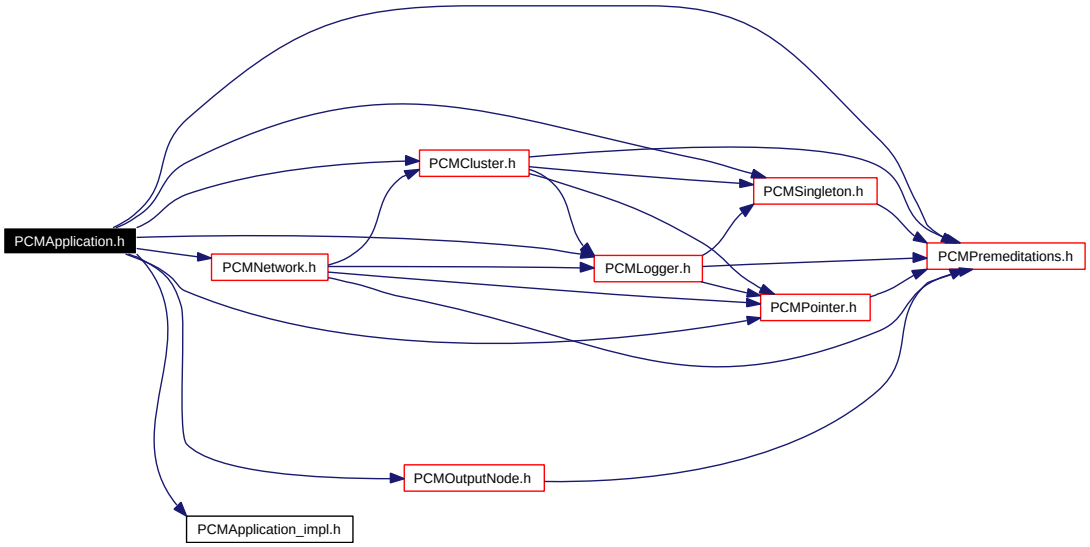
Namespaces

- namespace **ParCompMark**

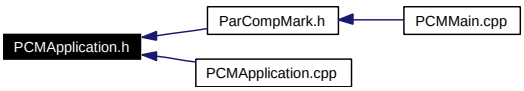
8.3 PCMApplication.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMSingleton.h"
#include "PCMLogger.h"
#include "PCMPointer.h"
#include "PCMNetwork.h"
#include "PCMCluster.h"
#include "PCMOutputNode.h"
#include "PCMApplication_impl.h"
```

Include dependency graph for PCMApplication.h:



This graph shows which files directly or indirectly include this file:



Namespaces

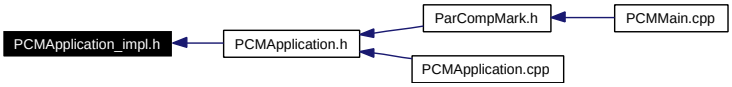
- namespace **ParCompMark**

Classes

- class **ParCompMark::Application**
- struct **ParCompMark::Application::CommandLineOption**

8.4 PCMApplication_impl.h File Reference

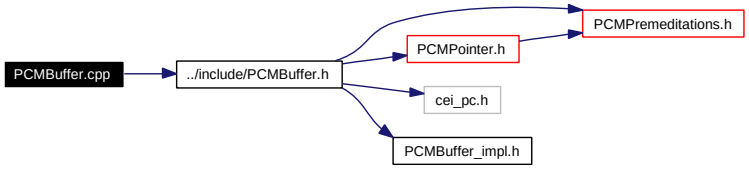
This graph shows which files directly or indirectly include this file:



8.5 PCMBuffer.cpp File Reference

```
#include "../include/PCMBuffer.h"
```

Include dependency graph for PCMBuffer.cpp:



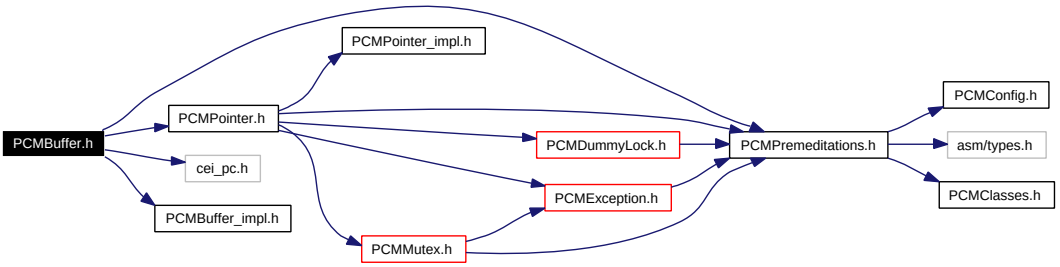
Namespaces

- namespace **ParCompMark**

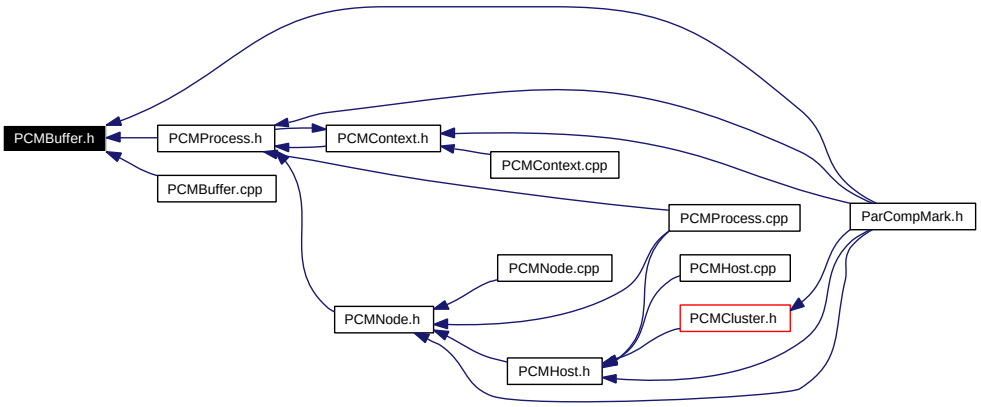
8.6 PCMBuffer.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMPointer.h"
#include <cei_pc.h>
#include "PCMBuffer_impl.h"
```

Include dependency graph for PCMBuffer.h:



This graph shows which files directly or indirectly include this file:



Namespaces

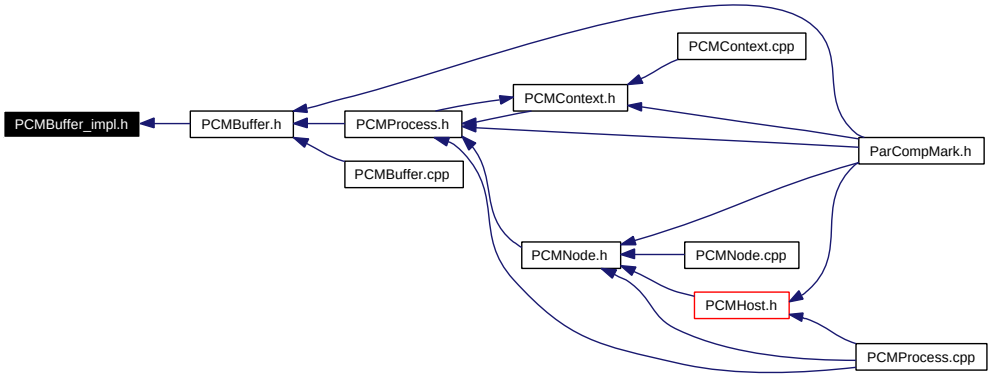
- namespace **ParCompMark**

Classes

- class **ParCompMark::Buffer**

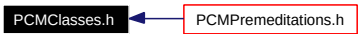
8.7 PCMBuffer_impl.h File Reference

This graph shows which files directly or indirectly include this file:



8.8 PCMClasses.h File Reference

This graph shows which files directly or indirectly include this file:



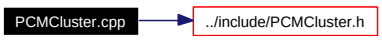
Namespaces

- namespace **ParCompMark**
- namespace **ParCompMarkTest**

8.9 PCMCluster.cpp File Reference

```
#include "../include/PCMCluster.h"
```

Include dependency graph for PCMCluster.cpp:



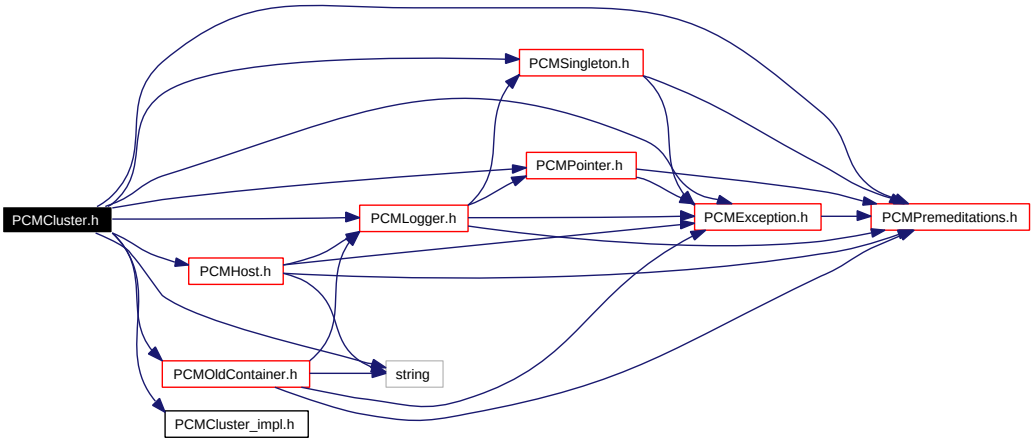
Namespaces

- namespace **ParCompMark**

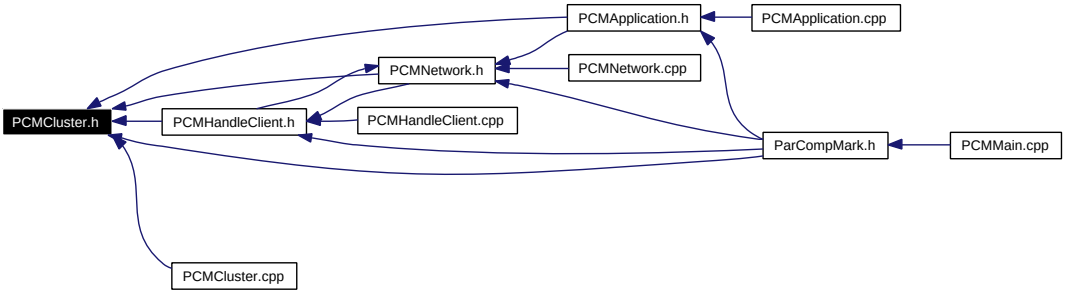
8.10 PCMCluster.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMSingleton.h"
#include "PCMLogger.h"
#include "PCMException.h"
#include "PCMHost.h"
#include "PCMOldContainer.h"
#include "PCMPointer.h"
#include <string>
#include "PCMCluster_impl.h"
```

Include dependency graph for PCMCluster.h:



This graph shows which files directly or indirectly include this file:



Namespaces

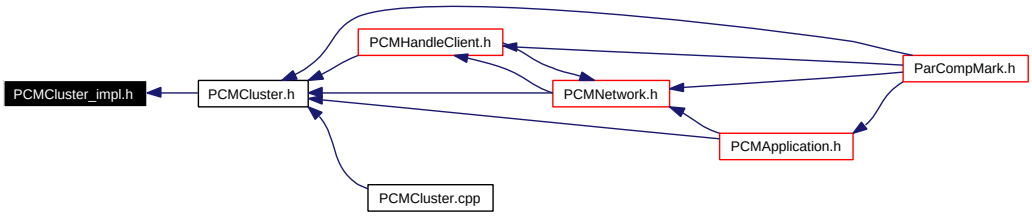
- namespace **ParCompMark**

Classes

- class **ParCompMark::Cluster**

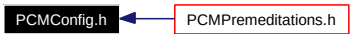
8.11 PCMCluster_impl.h File Reference

This graph shows which files directly or indirectly include this file:



8.12 PCMConfig.h File Reference

This graph shows which files directly or indirectly include this file:



Defines

- #define **PCM_opengl**

8.12.1 Define Documentation

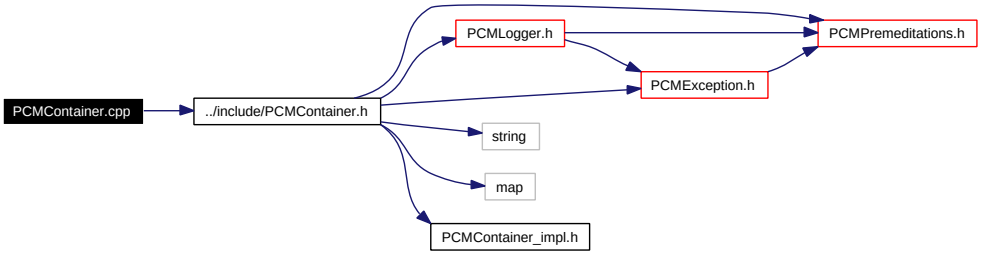
8.12.1.1 #define PCM_opengl

Definition at line 69 of file PCMConfig.h.

8.13 PCMContainer.cpp File Reference

```
#include "../include/PCMContainer.h"
```

Include dependency graph for PCMContainer.cpp:



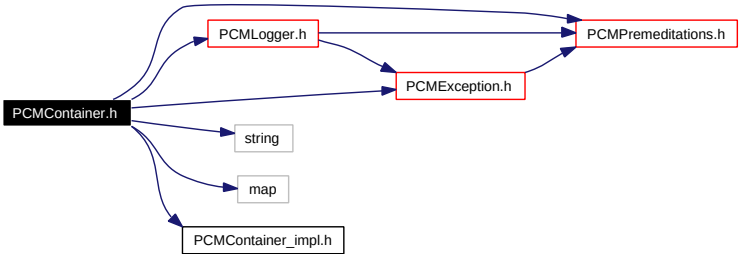
Namespaces

- namespace **ParCompMark**

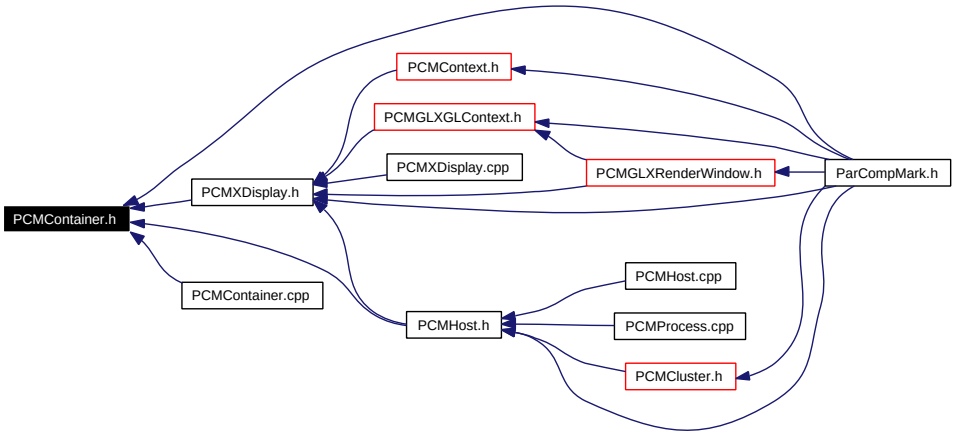
8.14 PCMContainer.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMLogger.h"
#include "PCMException.h"
#include <string>
#include <map>
#include "PCMContainer_impl.h"
```

Include dependency graph for PCMContainer.h:



This graph shows which files directly or indirectly include this file:



Namespaces

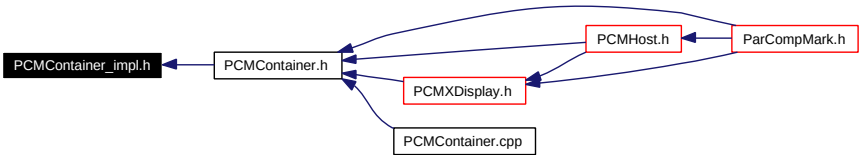
- namespace **ParCompMark**

Classes

- class **ParCompMark::Container**< **ElementType, LockType** >

8.15 PCMContainer_impl.h File Reference

This graph shows which files directly or indirectly include this file:



8.16 PCMContext.cpp File Reference

```
#include "../include/PCMContext.h"
```

Include dependency graph for PCMContext.cpp:

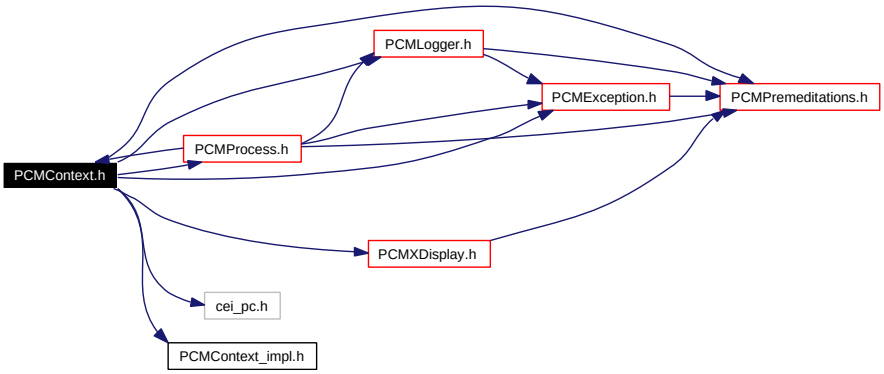
Namespaces

- namespace **ParCompMark**

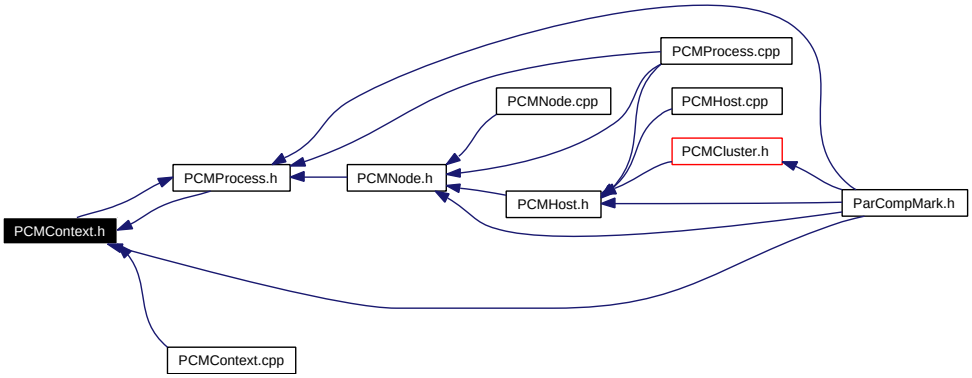
8.17 PCMContext.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMException.h"
#include "PCMLogger.h"
#include "PCMProcess.h"
#include "PCMXDisplay.h"
#include <cei_pc.h>
#include "PCMContext_impl.h"
```

Include dependency graph for PCMContext.h:



This graph shows which files directly or indirectly include this file:



Namespaces

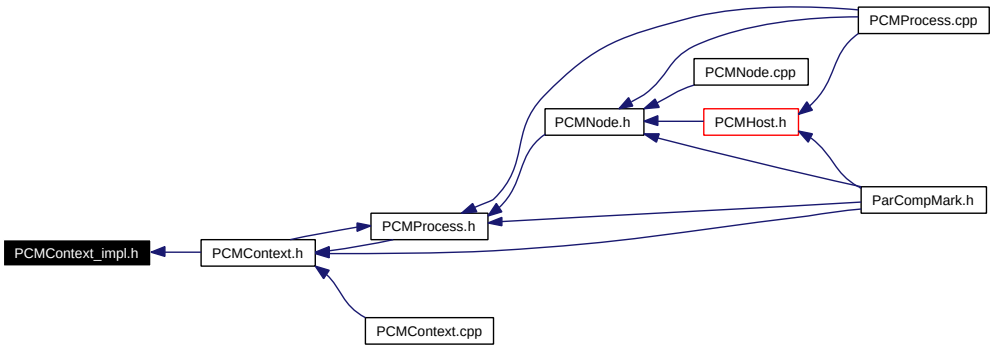
- namespace **ParCompMark**

Classes

- class **ParCompMark::Context**

8.18 PCMContext_impl.h File Reference

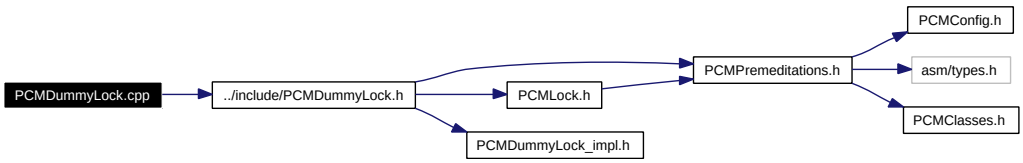
This graph shows which files directly or indirectly include this file:



8.19 PCMDummyLock.cpp File Reference

```
#include "../include/PCMDummyLock.h"
```

Include dependency graph for PCMDummyLock.cpp:



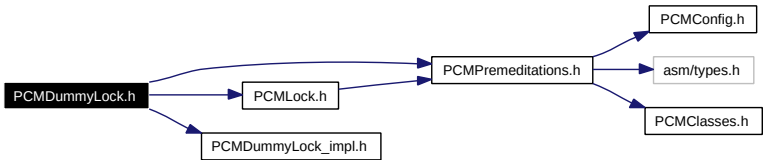
Namespaces

- namespace **ParCompMark**

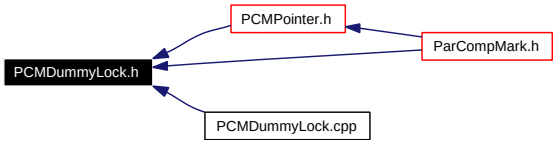
8.20 PCMDummyLock.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMLock.h"
#include "PCMDummyLock_impl.h"
```

Include dependency graph for PCMDummyLock.h:



This graph shows which files directly or indirectly include this file:



Namespaces

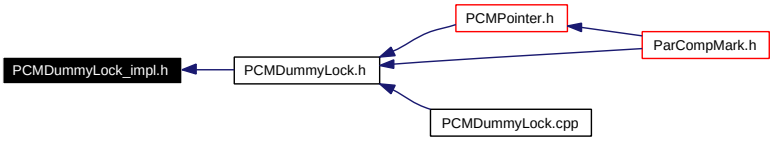
- namespace **ParCompMark**

Classes

- class **ParCompMark::DummyLock**

8.21 PCMDummyLock_impl.h File Reference

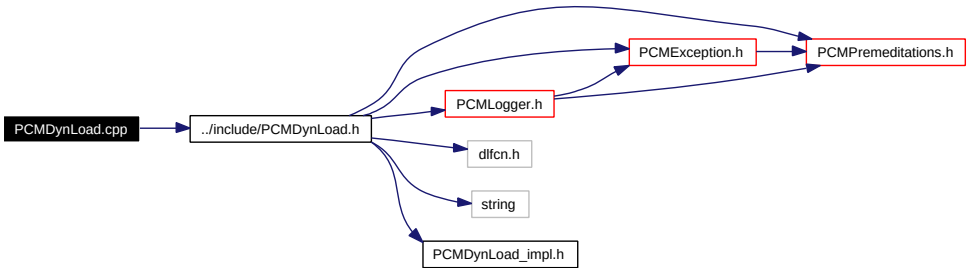
This graph shows which files directly or indirectly include this file:



8.22 PCMDynLoad.cpp File Reference

```
#include "../include/PCMDynLoad.h"
```

Include dependency graph for PCMDynLoad.cpp:



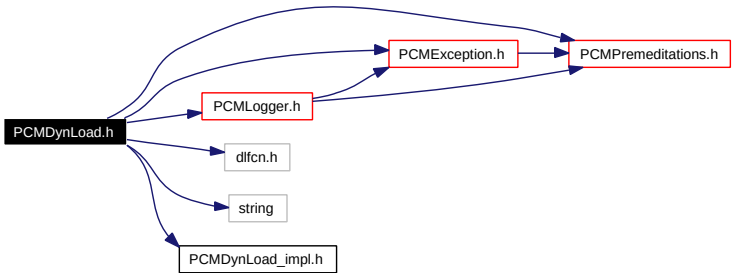
Namespaces

- namespace **ParCompMark**

8.23 PCMDynLoad.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMException.h"
#include "PCMLogger.h"
#include <dlfcn.h>
#include <string>
#include "PCMDynLoad_impl.h"
```

Include dependency graph for PCMDynLoad.h:



This graph shows which files directly or indirectly include this file:

Namespaces

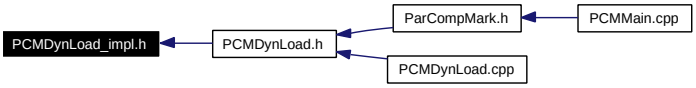
- namespace **ParCompMark**

Classes

- class **ParCompMark::DynLoad**

8.24 PCMDynLoad_impl.h File Reference

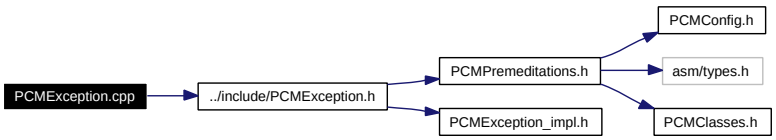
This graph shows which files directly or indirectly include this file:



8.25 PCMException.cpp File Reference

```
#include "../include/PCMException.h"
```

Include dependency graph for PCMException.cpp:



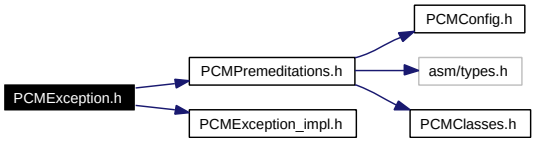
Namespaces

- namespace **ParCompMark**

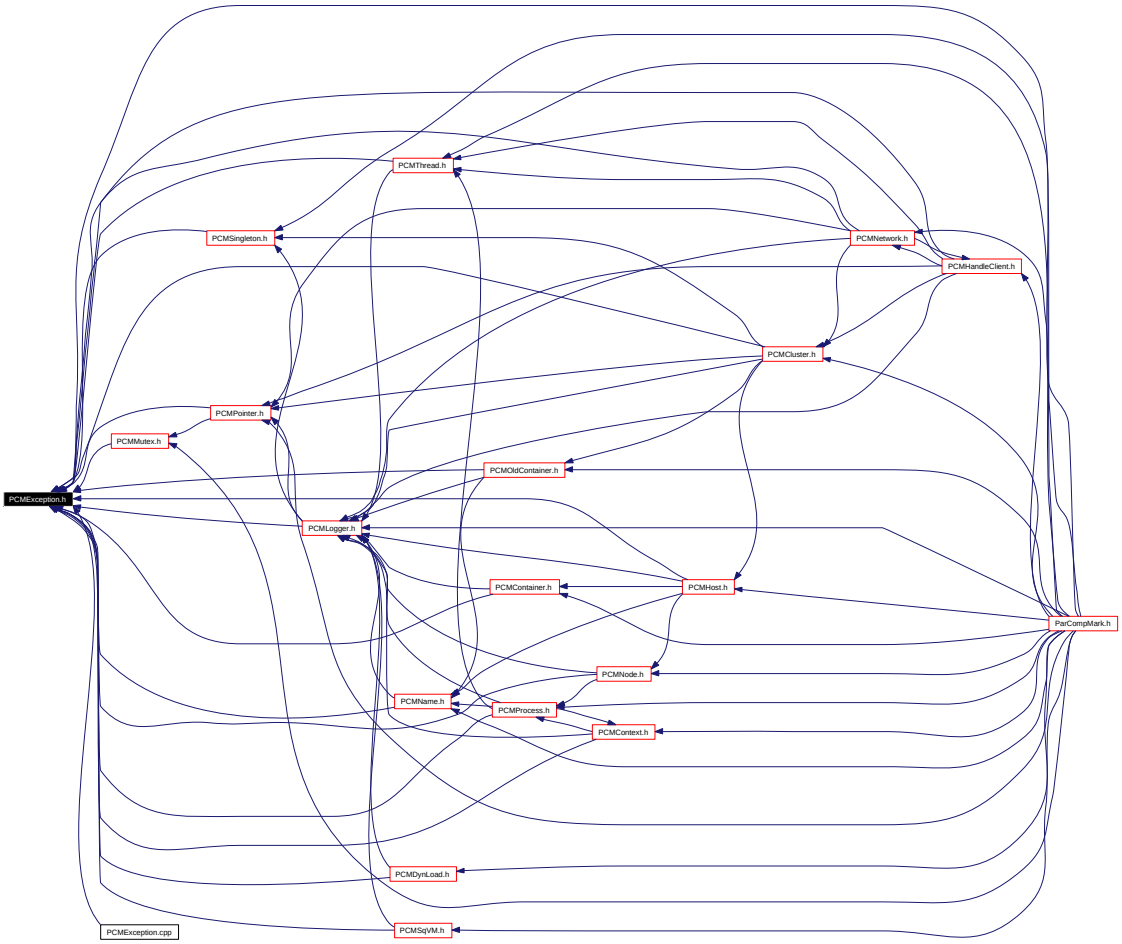
8.26 PCMException.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMException_impl.h"
```

Include dependency graph for PCMException.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMark**

Classes

- class **ParCompMark::Exception**

Defines

- #define **Except**(type, func, desc) throw(**ParCompMark::Exception**(Exception::type, desc, __FILE__, func, __LINE__))

  > *Macro for throwing exception.*

- #define **Assert**(cond, type, func) if(!(cond)) Except(Exception::type, func, "Assertion error.")

  > *Macro for asserting.*

8.26.1 Define Documentation

8.26.1.1 #define Assert(cond, type, func) if(!(cond)) Except(Exception::type, func, "Assertion error.")

> Macro for asserting.

Definition at line 45 of file PCMException.h.

Referenced by ParCompMark::OldContainer::add(), ParCompMark::OutputNode::addChildNode(), ParCompMark::Network::closeClient(), ParCompMark::Context::Context(), ParCompMark::OutputNode::createChildNode(), ParCompMark::GLXRenderWindow::destroyWindow(), ParCompMark::XDisplay::finalize(), ParCompMark::GLXRenderWindow::finalize(), ParCompMark::GLXGLContext::finalize(), ParCompMark::OldContainer::get(), ParCompMark::OutputNode::getAttribute(), ParCompMark::OutputNode::getChildNode(), ParCompMark::DynLoad::getFunc(), ParCompMark::OutputNode::hasAttribute(), ParCompMark::OutputNode::hasChildNode(), ParCompMark::Logger::init(), ParCompMark::Network::initClient(), ParCompMark::XDisplay::initialize(), ParCompMark::GLXRenderWindow::initialize(), ParCompMark::GLXGLContext::initialize(), ParCompMark::Thread::initThread(), ParCompMark::Thread::joinThread(), ParCompMark::Node::Node(), ParCompMark::OutputNode::OutputNode(), ParCompMark::Process::Process(), ParCompMark::Network::recieveMessage(), ParCompMark::OldContainer::remove(), ParCompMark::HandleClient::sendMessage(), ParCompMark::OutputNode::setAttribute(), ParCompMark::GLXGLContext::setCurrent(), ParCompMark::Thread::shutDownThread(), ParCompMark::Thread::startThread(), and ParCompMark::HandleClient::task().

8.26.1.2 #define Except(type, func, desc) throw(ParCompMark::Exception(Exception::type, desc, __FILE__, func, __LINE__))

> Macro for throwing exception.

Definition at line 43 of file PCMException.h.

Referenced by ParCompMark::Network::acceptClientConnection(), ParCompMark::Thread::entryPoint(), ParCompMark::Context::finalize(), ParCompMark::DynLoad::getFunc(), ParCompMark::Host::Host(), ParCompMark::Context::init(), ParCompMark::Network::initBroadcastRecieve(), ParCompMark::Network::initBroadcastSend(), ParCompMark::Network::initClient(), ParCompMark::Network::initServer(), ParCompMark::Thread::joinThread(), ParCompMark::DynLoad::load(), ParCompMark::Network::recieveBroadcastMessage(), ParCompMark::Network::sendBroadcastMessage(), ParCompMark::Network::sendMessage(), ParCompMark::HandleClient::sendMessage(),

ParCompMark::GLXGLContext::setCurrent(), ParCompMark::Thread::shutDownThread(), ParCompMark::Process::task(), and ParCompMark::DynLoad::unload().

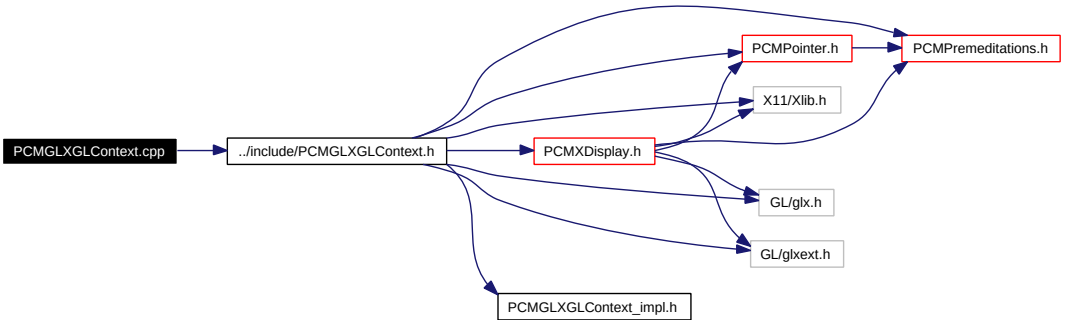
8.27 PCMException_impl.h File Reference

This graph shows which files directly or indirectly include this file:

8.28 PCMGLXGLContext.cpp File Reference

```
#include "../include/PCMGLXGLContext.h"
```

Include dependency graph for PCMGLXGLContext.cpp:



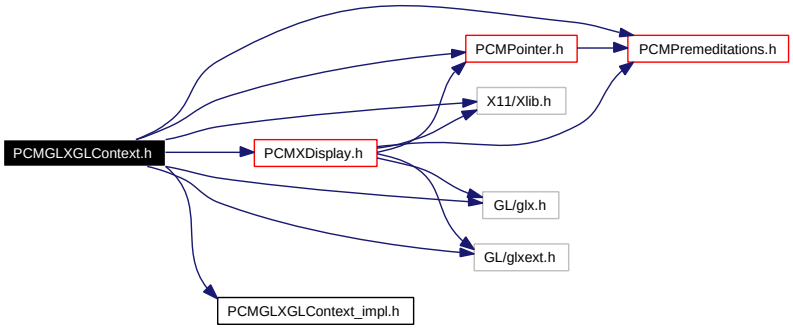
Namespaces

- namespace **ParCompMark**

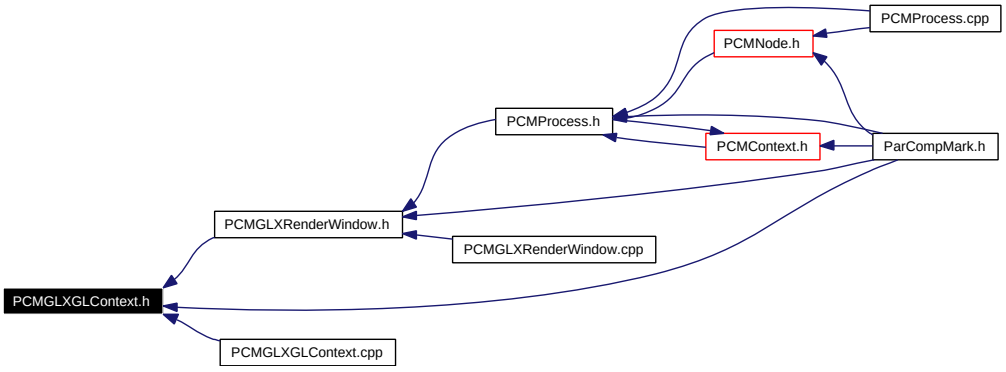
8.29 PCMGLXGLContext.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMPointer.h"
#include "PCMXDisplay.h"
#include <X11/Xlib.h>
#include <GL/glx.h>
#include <GL/glxext.h>
#include "PCMGLXGLContext_impl.h"
```

Include dependency graph for PCMGLXGLContext.h:



This graph shows which files directly or indirectly include this file:



Namespaces

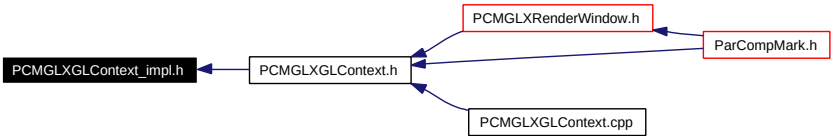
- namespace **ParCompMark**

Classes

- class **ParCompMark::GLXGLContext**

8.30 PCMGLXGLContext_impl.h File Reference

This graph shows which files directly or indirectly include this file:



8.31 PCMGLXRenderWindow.cpp File Reference

```
#include "../include/PCMGLXRenderWindow.h"
```

Include dependency graph for PCMGLXRenderWindow.cpp:

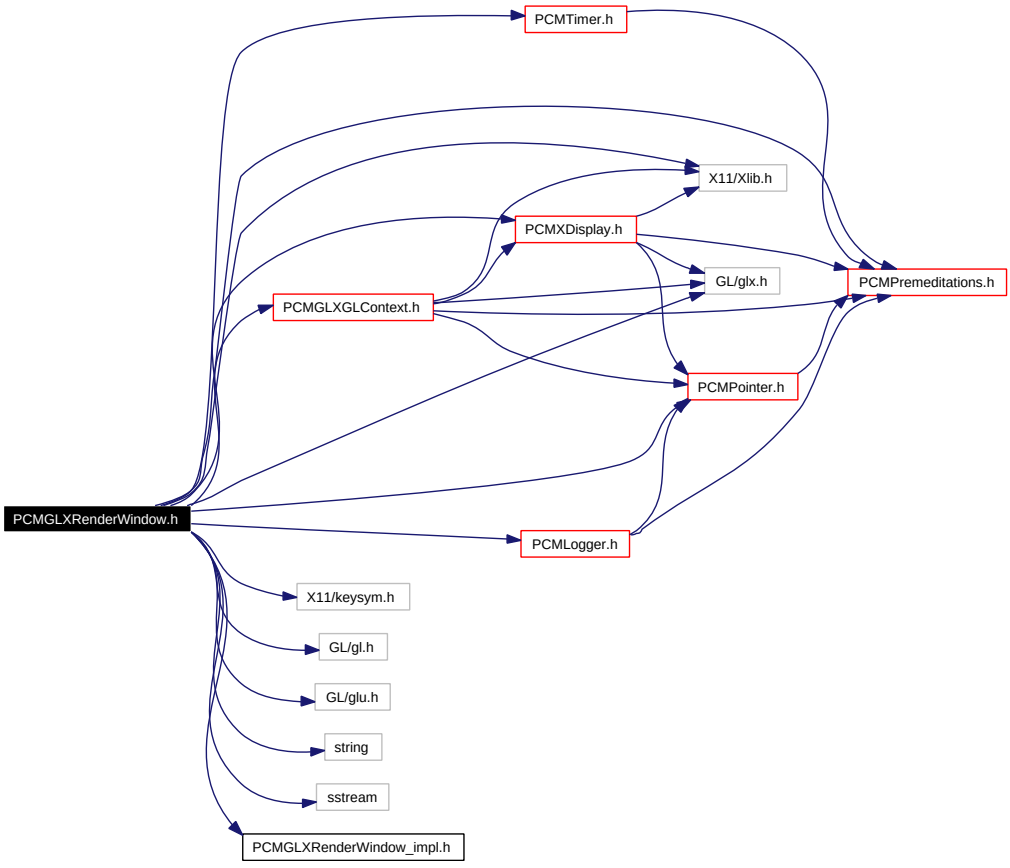
Namespaces

- namespace **ParCompMark**

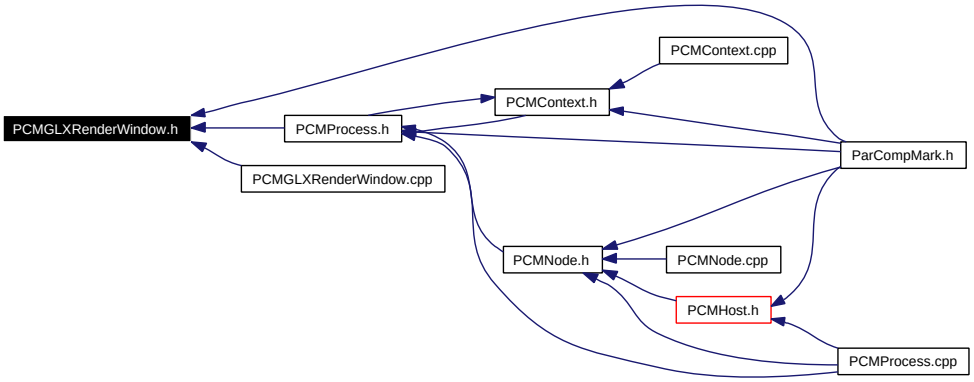
8.32 PCMGLXRenderWindow.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMPointer.h"
#include "PCMLogger.h"
#include "PCMGLXGLContext.h"
#include "PCMXDisplay.h"
#include "PCMTimer.h"
#include <X11/Xlib.h>
#include <X11/keysym.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glx.h>
#include <string>
#include <sstream>
#include "PCMGLXRenderWindow_impl.h"
```

Include dependency graph for PCMGLXRenderWindow.h:



This graph shows which files directly or indirectly include this file:



Namespaces

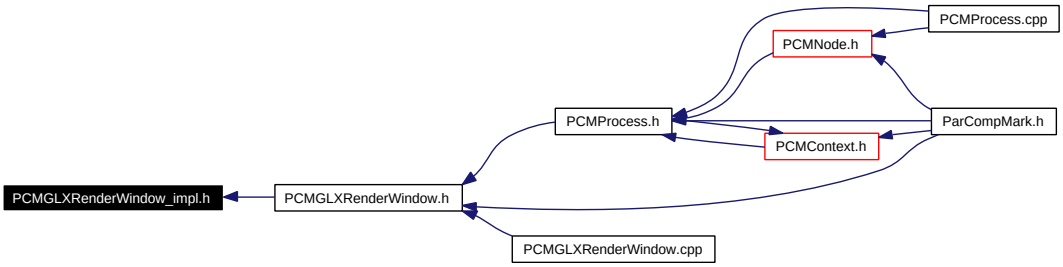
- namespace **ParCompMark**

Classes

- class **ParCompMark::GLXRenderWindow**
- struct **ParCompMark::GLXRenderWindow::WindowStatistics**

8.33 PCMGLXRenderWindow_impl.h File Reference

This graph shows which files directly or indirectly include this file:



8.34 PCMHandleClient.cpp File Reference

```
#include "../include/PCMHandleClient.h"
```

Include dependency graph for PCMHandleClient.cpp:

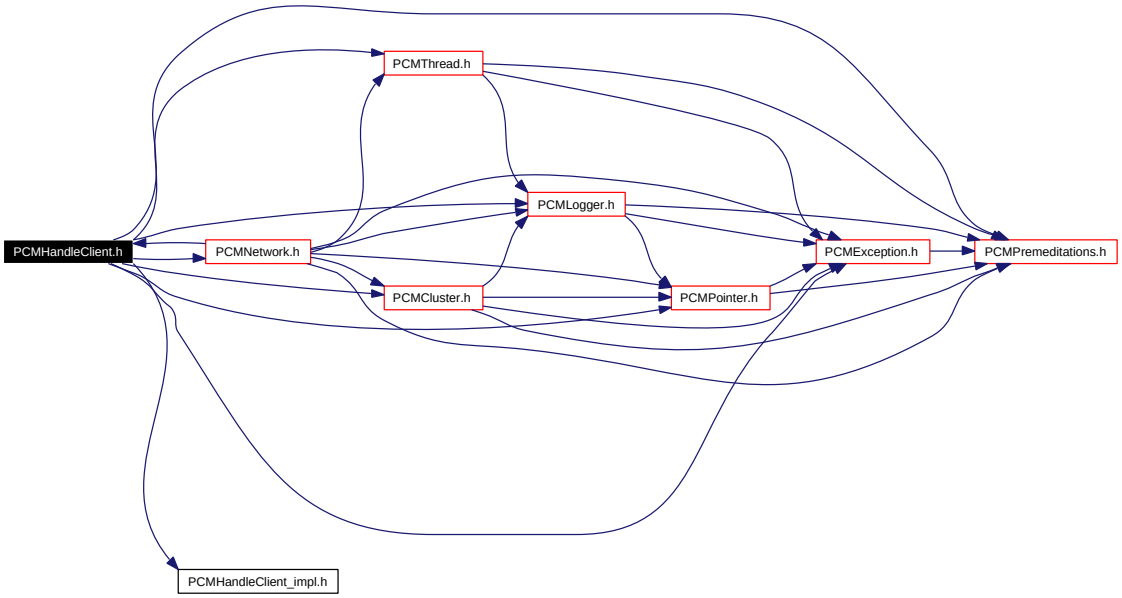
Namespaces

- namespace **ParCompMark**

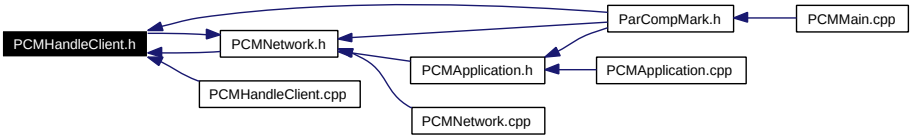
8.35 PCMHandleClient.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMThread.h"
#include "PCMLogger.h"
#include "PCMException.h"
#include "PCMCluster.h"
#include "PCMNetwork.h"
#include "PCMPointer.h"
#include "PCMHandleClient_impl.h"
```

Include dependency graph for PCMHandleClient.h:



This graph shows which files directly or indirectly include this file:



Namespaces

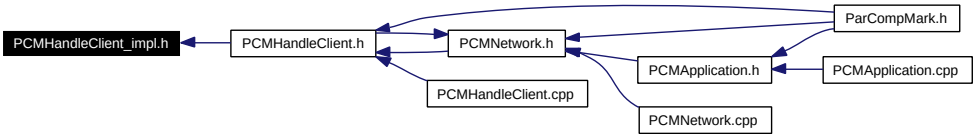
- namespace **ParCompMark**

Classes

- class **ParCompMark::HandleClient**

8.36 PCMHandleClient_impl.h File Reference

This graph shows which files directly or indirectly include this file:



8.37 PCMHost.cpp File Reference

```
#include "../include/PCMHost.h"
```

Include dependency graph for PCMHost.cpp:

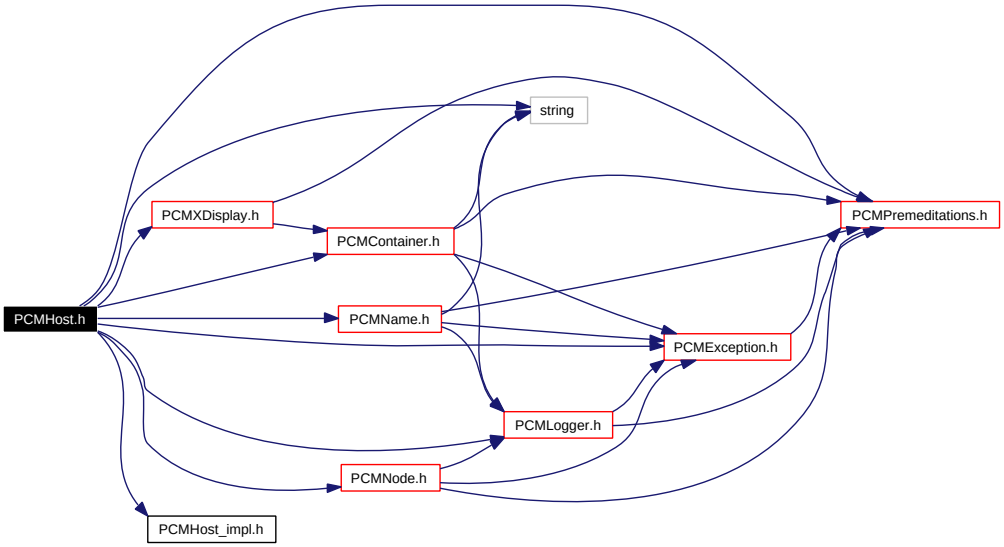
Namespaces

- namespace **ParCompMark**

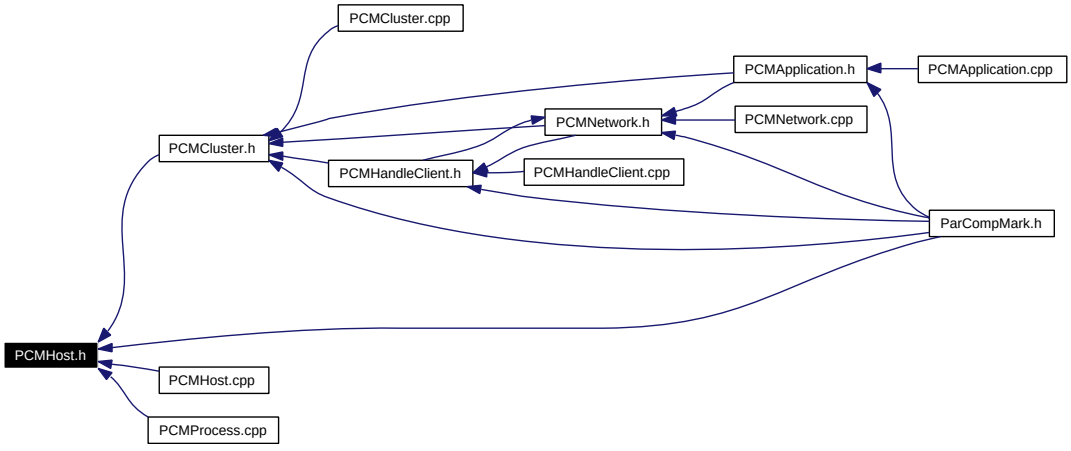
8.38 PCMHost.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMName.h"
#include "PCMLogger.h"
#include "PCMException.h"
#include "PCMContainer.h"
#include "PCMXDisplay.h"
#include "PCMNode.h"
#include <string>
#include "PCMHost_impl.h"
```

Include dependency graph for PCMHost.h:



This graph shows which files directly or indirectly include this file:



Namespaces

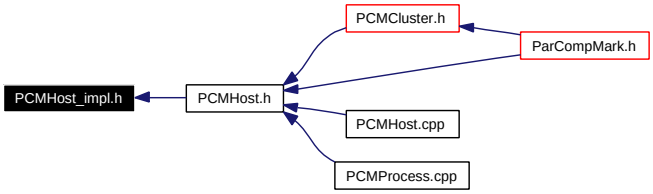
- namespace **ParCompMark**

Classes

- class **ParCompMark::Host**

8.39 PCMHost_impl.h File Reference

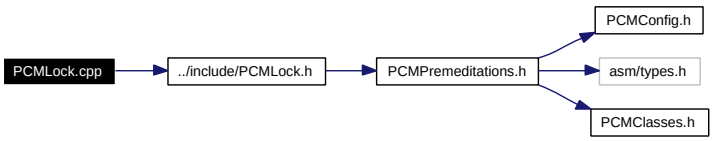
This graph shows which files directly or indirectly include this file:



8.40 PCMLock.cpp File Reference

```
#include "../include/PCMLock.h"
```

Include dependency graph for PCMLock.cpp:



Namespaces

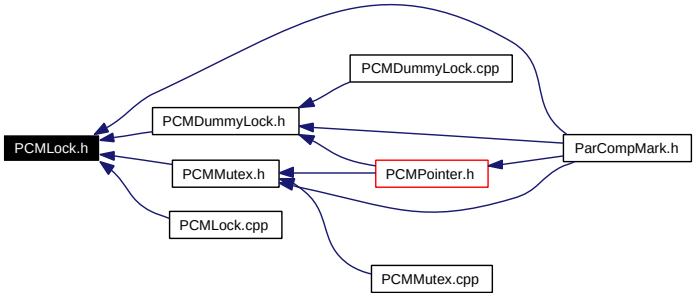
- namespace **ParCompMark**

8.41 PCMLock.h File Reference

```
#include "PCMPremeditations.h"
```

Include dependency graph for PCMLock.h:

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMark**

Classes

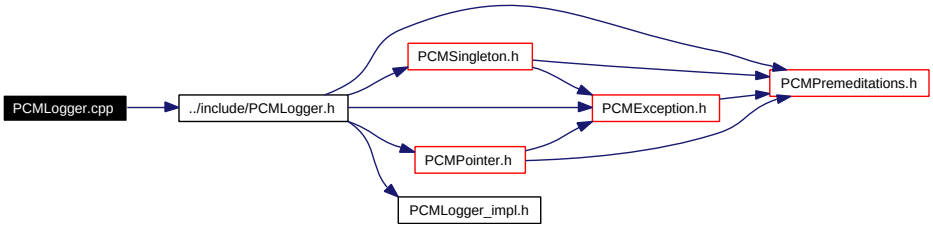
- class **ParCompMark::Lock**

8.42 PCMLock_impl.h File Reference

8.43 PCMLogger.cpp File Reference

```
#include "../include/PCMLogger.h"
```

Include dependency graph for PCMLogger.cpp:



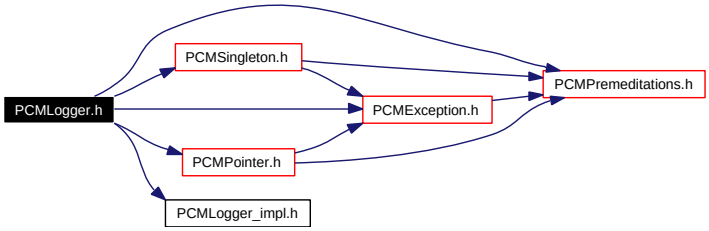
Namespaces

- namespace **ParCompMark**

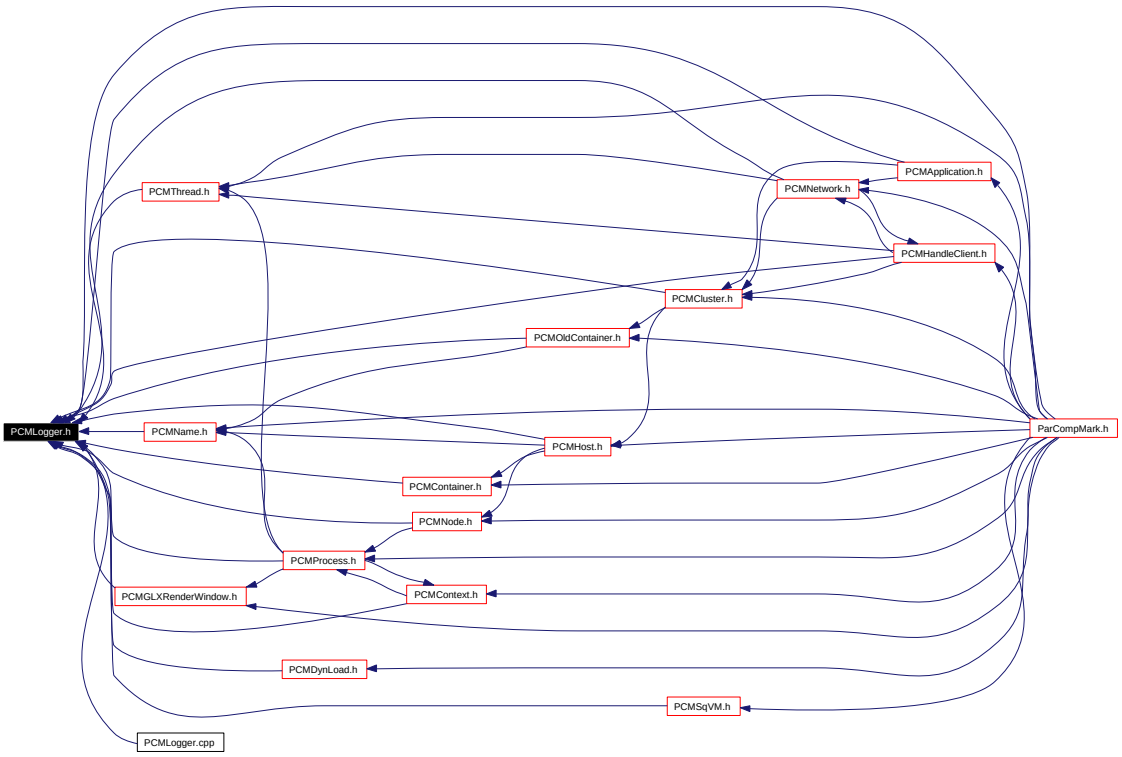
8.44 PCMLogger.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMSingleton.h"
#include "PCMPointer.h"
#include "PCMException.h"
#include "PCMLogger_impl.h"
```

Include dependency graph for PCMLogger.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMark**

Classes

- class **ParCompMark::Logger**

8.45 PCMLogger_impl.h File Reference

This graph shows which files directly or indirectly include this file:

8.46 PCMMain.cpp File Reference

```
#include "../include/ParCompMark.h"
```

Include dependency graph for PCMMain.cpp:

Functions

- int **main** (const int argc, const char **argv)

8.46.1 Function Documentation

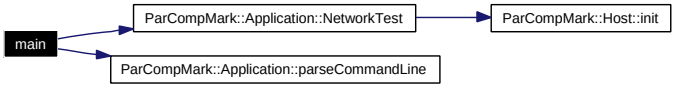
8.46.1.1 int main (const int *argc*, const char ** *argv*)

Program entry point

Definition at line 33 of file PCMMain.cpp.

References ParCompMark::Application::NetworkTest(), and ParCompMark::Application::parseCommandLine().

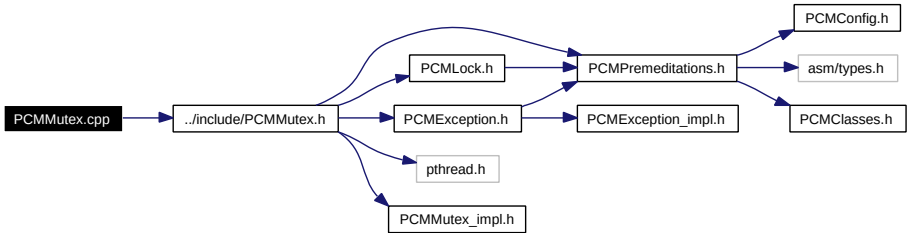
Here is the call graph for this function:



8.47 PCMMutex.cpp File Reference

```
#include "../include/PCMMutex.h"
```

Include dependency graph for PCMMutex.cpp:



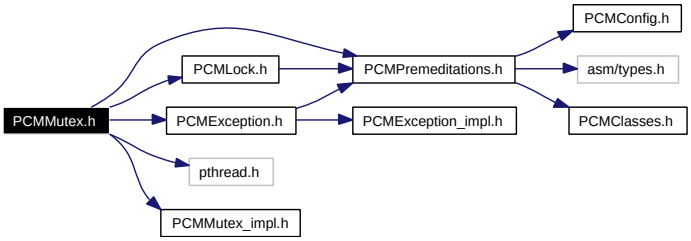
Namespaces

- namespace **ParCompMark**

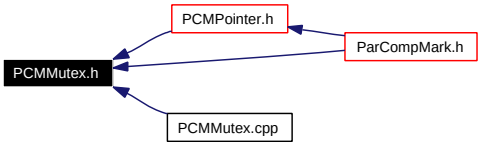
8.48 PCMMutex.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMLock.h"
#include "PCMException.h"
#include <pthread.h>
#include "PCMMutex_impl.h"
```

Include dependency graph for PCMMutex.h:



This graph shows which files directly or indirectly include this file:



Namespaces

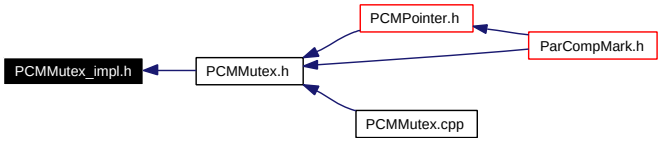
- namespace **ParCompMark**

Classes

- class **ParCompMark::Mutex**

8.49 PCMMutex_impl.h File Reference

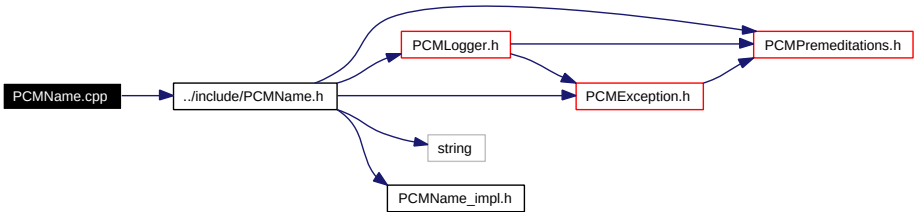
This graph shows which files directly or indirectly include this file:



8.50 PCMName.cpp File Reference

```
#include "../include/PCMName.h"
```

Include dependency graph for PCMName.cpp:



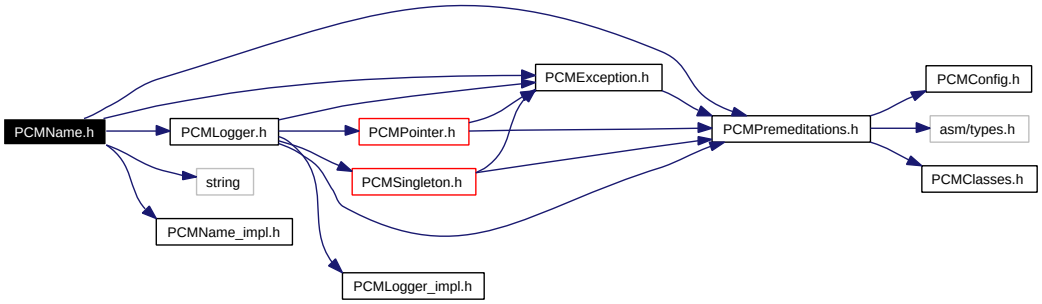
Namespaces

- namespace **ParCompMark**

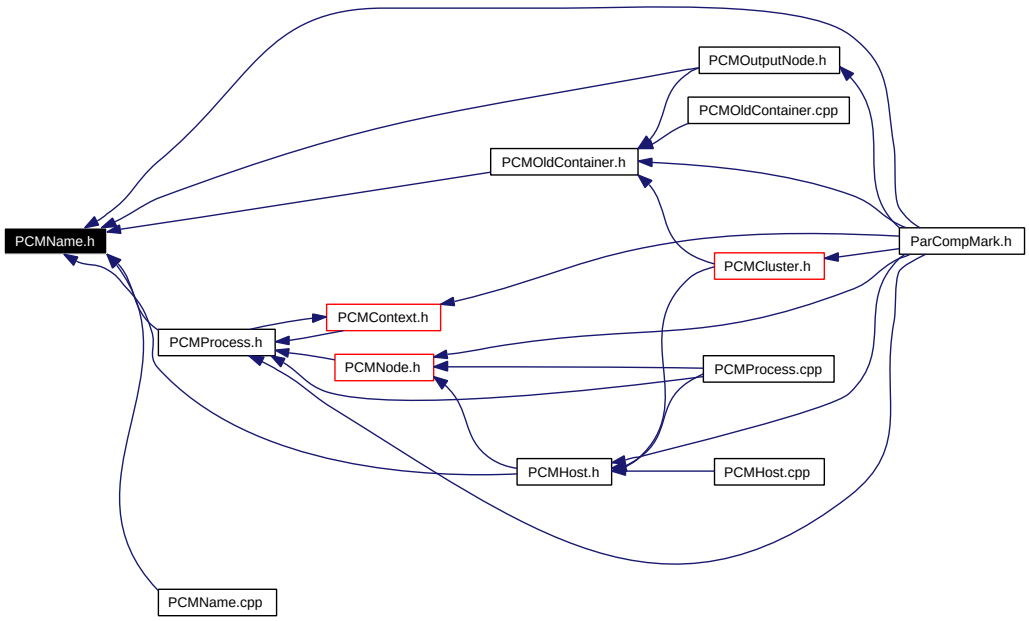
8.51 PCMName.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMLogger.h"
#include "PCMException.h"
#include <string>
#include "PCMName_impl.h"
```

Include dependency graph for PCMName.h:



This graph shows which files directly or indirectly include this file:



Namespaces

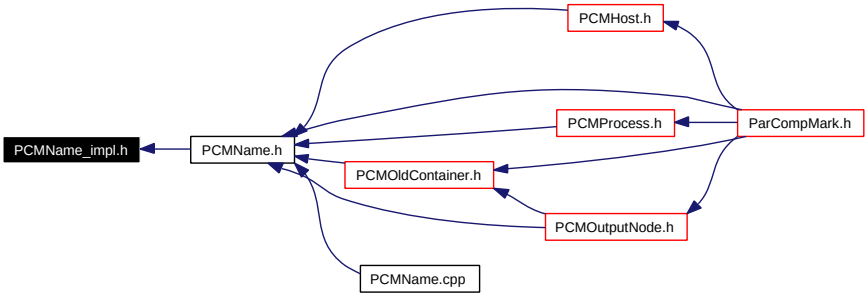
- namespace **ParCompMark**

Classes

- class **ParCompMark::Name**

8.52 PCMName_impl.h File Reference

This graph shows which files directly or indirectly include this file:



8.53 PCMNetwork.cpp File Reference

```
#include "../include/PCMNetwork.h"
```

Include dependency graph for PCMNetwork.cpp:

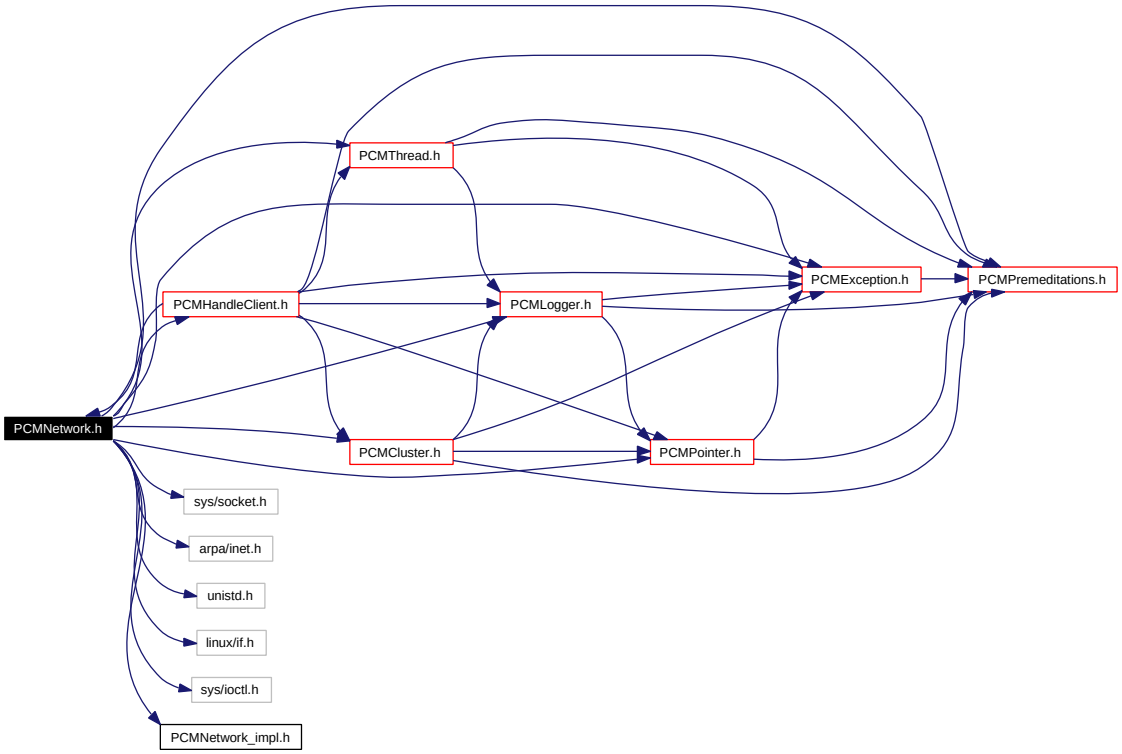
Namespaces

- namespace **ParCompMark**

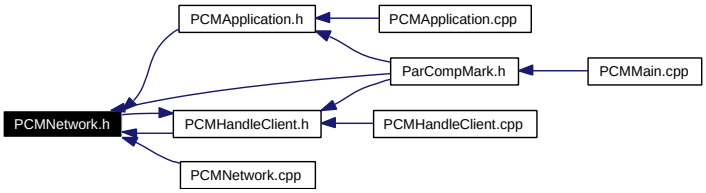
8.54 PCMNetwork.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMThread.h"
#include "PCMPointer.h"
#include "PCMException.h"
#include "PCMLogger.h"
#include "PCMHandleClient.h"
#include "PCMCluster.h"
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <linux/if.h>
#include <sys/ioctl.h>
#include "PCMNetwork_impl.h"
```

Include dependency graph for PCMNetwork.h:



This graph shows which files directly or indirectly include this file:



Namespaces

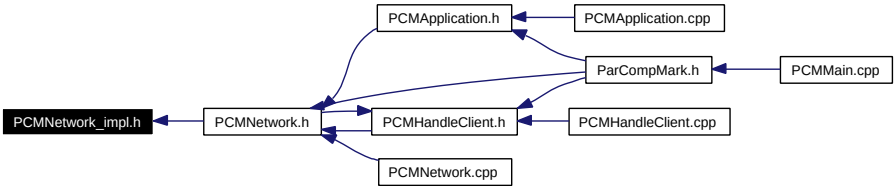
- namespace **ParCompMark**

Classes

- class **ParCompMark::Network**

8.55 PCMNetwork_impl.h File Reference

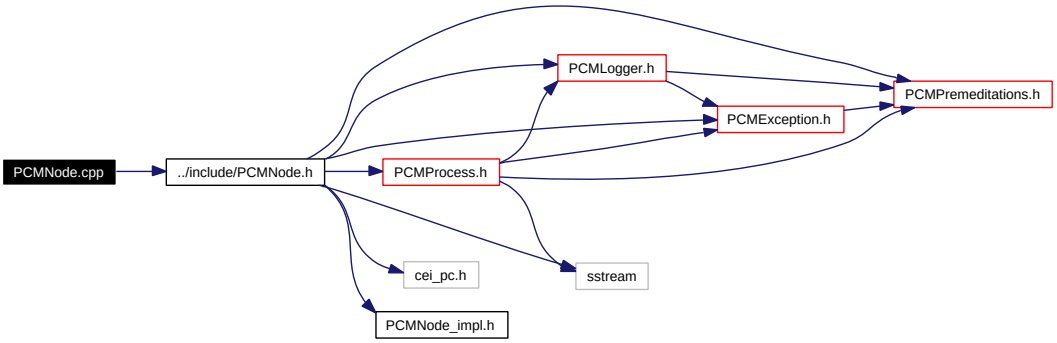
This graph shows which files directly or indirectly include this file:



8.56 PCMNode.cpp File Reference

```
#include "../include/PCMNode.h"
```

Include dependency graph for PCMNode.cpp:



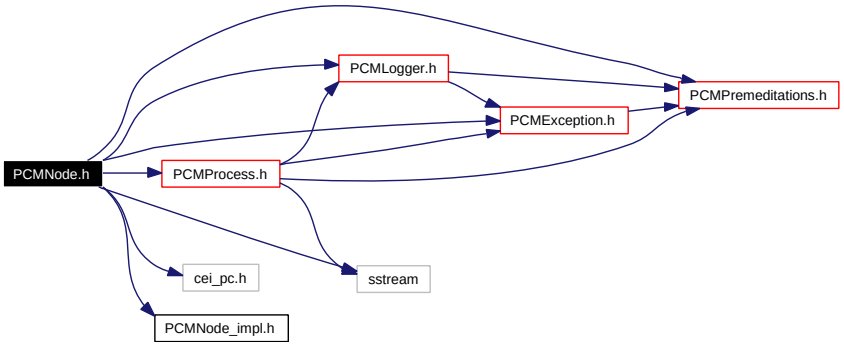
Namespaces

- namespace **ParCompMark**

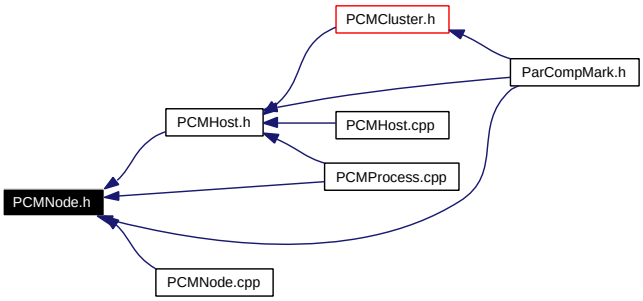
8.57 PCMNode.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMException.h"
#include "PCMLogger.h"
#include "PCMProcess.h"
#include <cei_pc.h>
#include <sstream>
#include "PCMNode_impl.h"
```

Include dependency graph for PCMNode.h:



This graph shows which files directly or indirectly include this file:



Namespaces

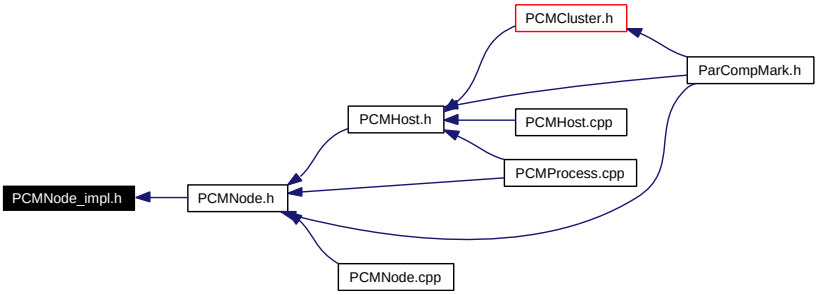
- namespace **ParCompMark**

Classes

- class **ParCompMark::Node**

8.58 PCMNode_impl.h File Reference

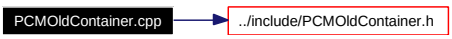
This graph shows which files directly or indirectly include this file:



8.59 PCMOldContainer.cpp File Reference

```
#include "../include/PCMOldContainer.h"
```

Include dependency graph for PCMOldContainer.cpp:



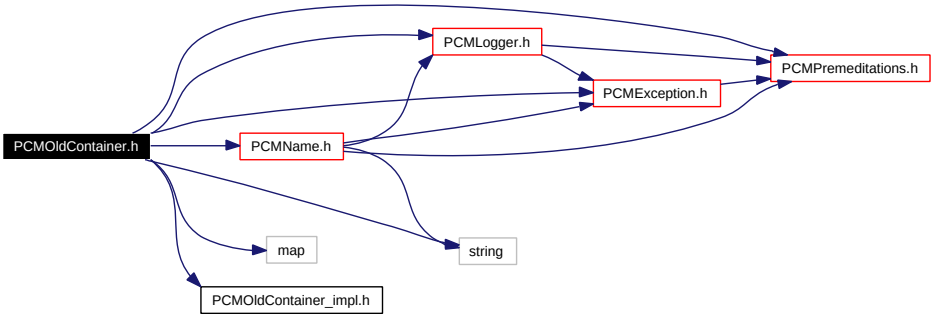
Namespaces

- namespace **ParCompMark**

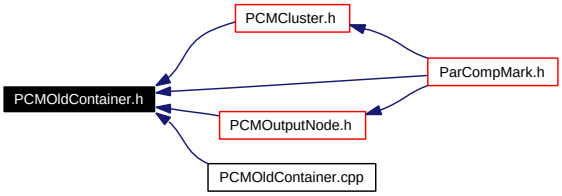
8.60 PCMOldContainer.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMLogger.h"
#include "PCMException.h"
#include "PCMName.h"
#include <string>
#include <map>
#include "PCMOldContainer_impl.h"
```

Include dependency graph for PCMOldContainer.h:



This graph shows which files directly or indirectly include this file:



Namespaces

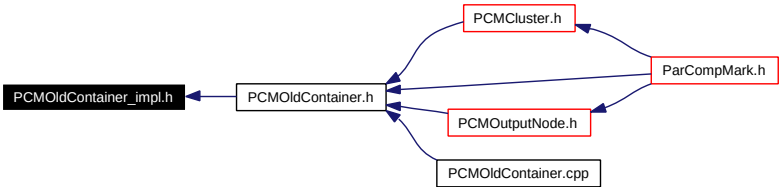
- namespace **ParCompMark**

Classes

- class **ParCompMark::OldContainer**

8.61 PCMOldContainer_impl.h File Reference

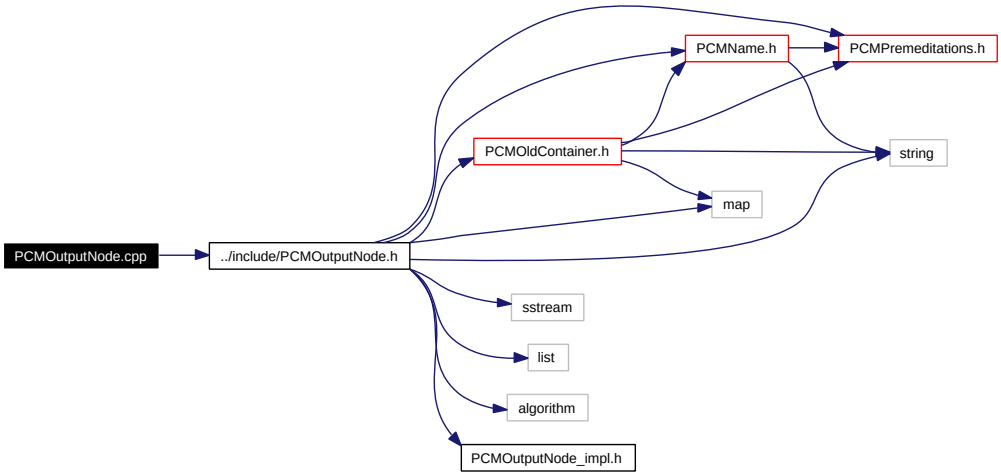
This graph shows which files directly or indirectly include this file:



8.62 PCMOutputNode.cpp File Reference

```
#include "../include/PCMOutputNode.h"
```

Include dependency graph for PCMOutputNode.cpp:



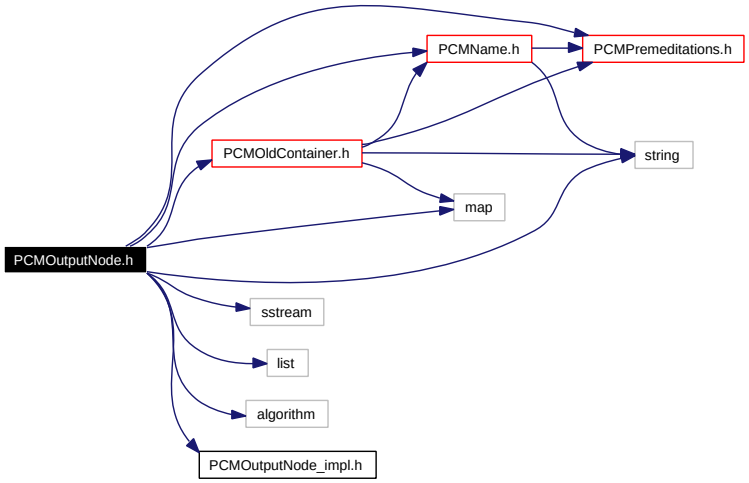
Namespaces

- namespace **ParCompMark**

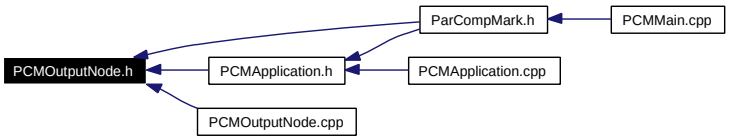
8.63 PCMOutputNode.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMName.h"
#include "PCMOldContainer.h"
#include <string>
#include <sstream>
#include <list>
#include <map>
#include <algorithm>
#include "PCMOutputNode_impl.h"
```

Include dependency graph for PCMOutputNode.h:



This graph shows which files directly or indirectly include this file:



Namespaces

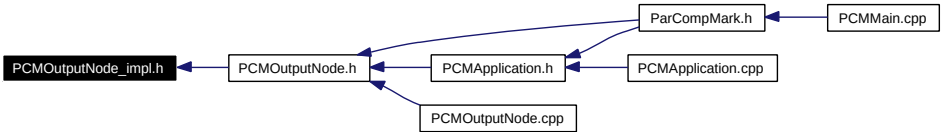
- namespace **ParCompMark**

Classes

- class **ParCompMark::OutputNode**

8.64 PCMOutputNode_impl.h File Reference

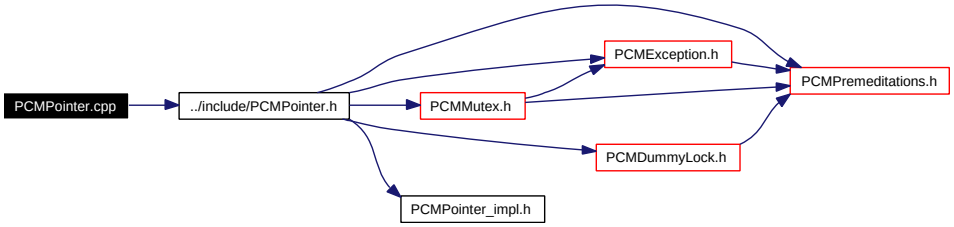
This graph shows which files directly or indirectly include this file:



8.65 PCMPointer.cpp File Reference

```
#include "../include/PCMPointer.h"
```

Include dependency graph for PCMPointer.cpp:



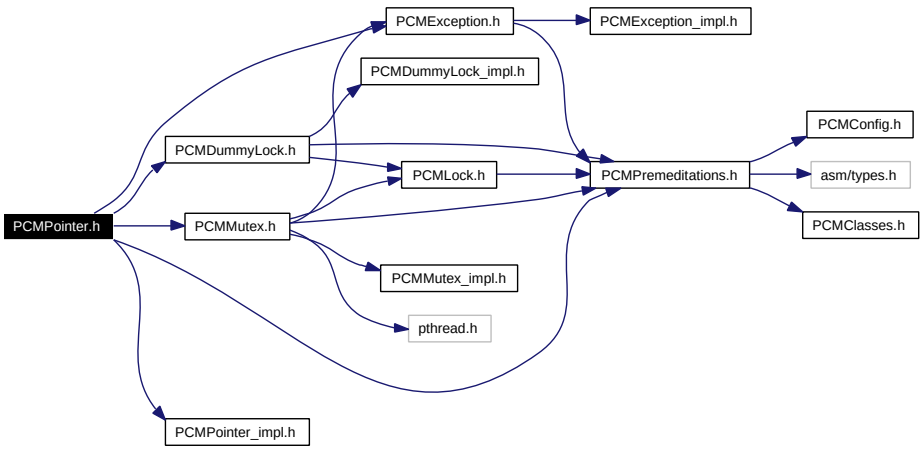
Namespaces

- namespace **ParCompMark**

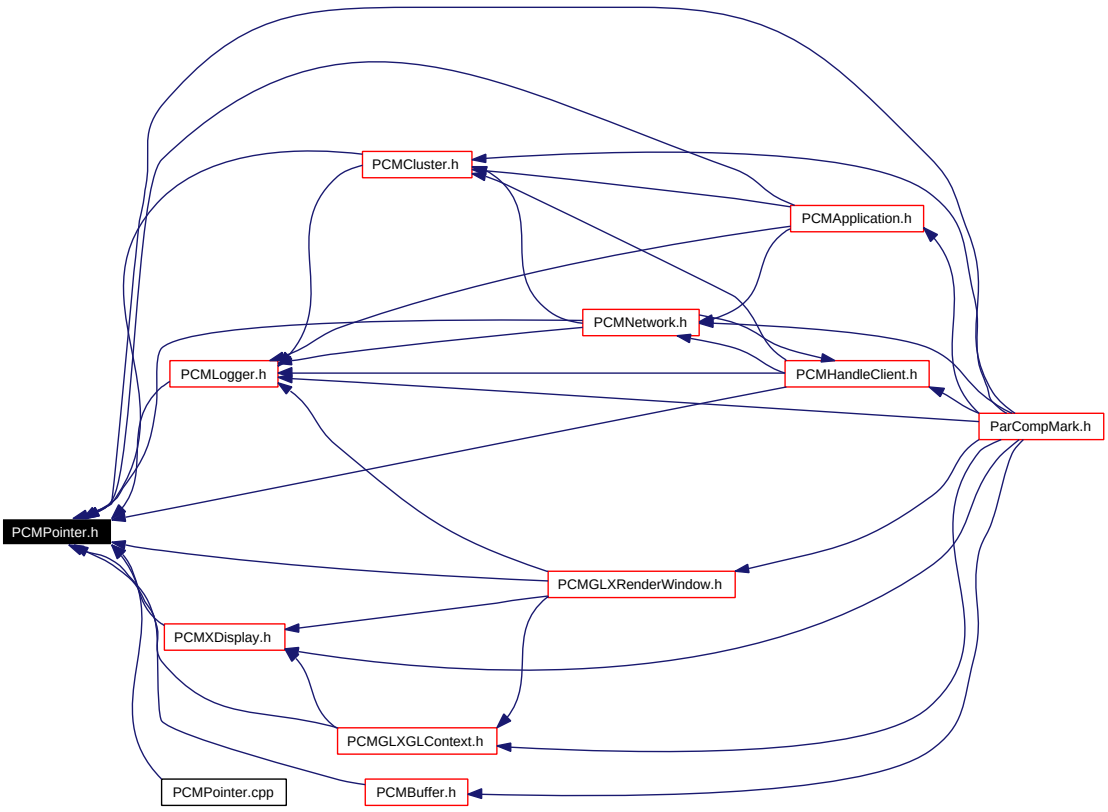
8.66 PCMPointer.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMException.h"
#include "PCMDummyLock.h"
#include "PCMMutex.h"
#include "PCMPointer_impl.h"
```

Include dependency graph for PCMPointer.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMark**

Classes

- class **ParCompMark::Pointer< T, Lock >**
- struct **ParCompMark::Pointer< T, Lock >::Meta**

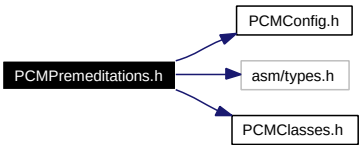
8.67 PCMPointer_impl.h File Reference

This graph shows which files directly or indirectly include this file:

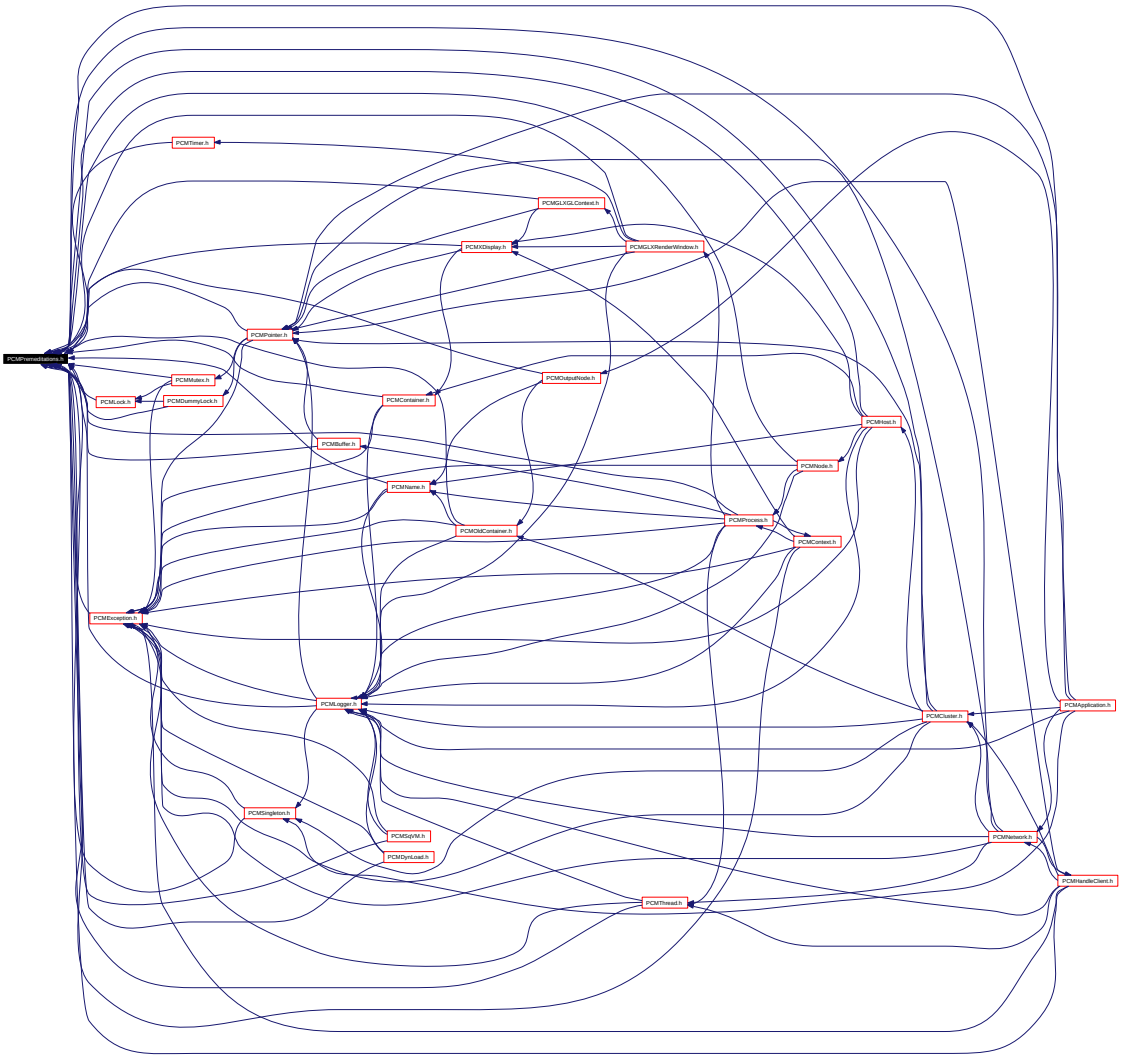
8.68 PCMPremeditations.h File Reference

```
#include "PCMConfig.h"
#include <asm/types.h>
#include "PCMClasses.h"
```

Include dependency graph for PCMPremeditations.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMark**

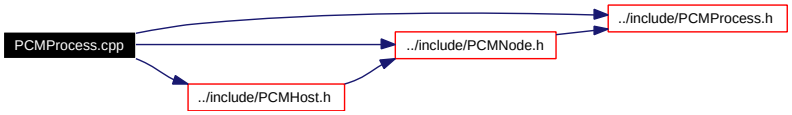
Typedefs

- typedef __u8 **ParCompMark::u8**
- typedef __s8 **ParCompMark::s8**
- typedef __u16 **ParCompMark::u16**
- typedef __s16 **ParCompMark::s16**
- typedef __u32 **ParCompMark::u32**
- typedef __s32 **ParCompMark::s32**
- typedef __u64 **ParCompMark::u64**
- typedef __s64 **ParCompMark::s64**
- typedef double **ParCompMark::Real**

8.69 PCMProcess.cpp File Reference

```
#include "../include/PCMProcess.h"
#include "../include/PCMNode.h"
#include "../include/PCMHost.h"
```

Include dependency graph for PCMProcess.cpp:



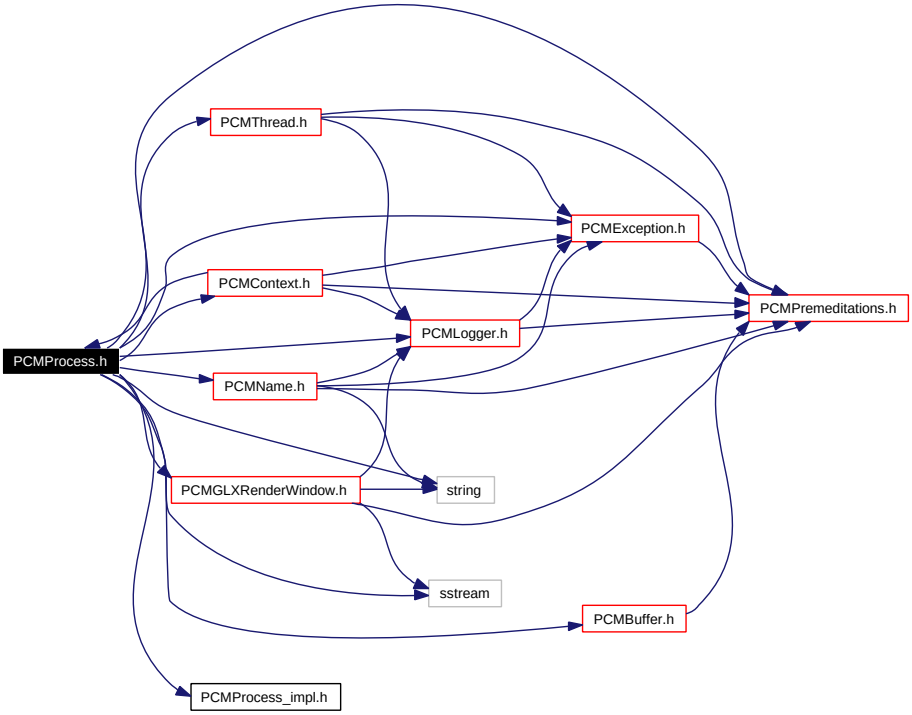
Namespaces

- namespace **ParCompMark**

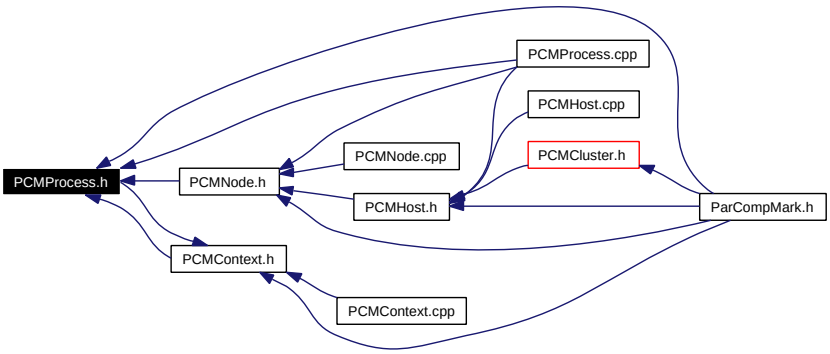
8.70 PCMProcess.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMThread.h"
#include "PCMException.h"
#include "PCMLogger.h"
#include "PCMContext.h"
#include "PCMName.h"
#include "PCMBuffer.h"
#include "PCMGLXRenderWindow.h"
#include <string>
#include <sstream>
#include "PCMProcess_impl.h"
```

Include dependency graph for PCMProcess.h:



This graph shows which files directly or indirectly include this file:



Namespaces

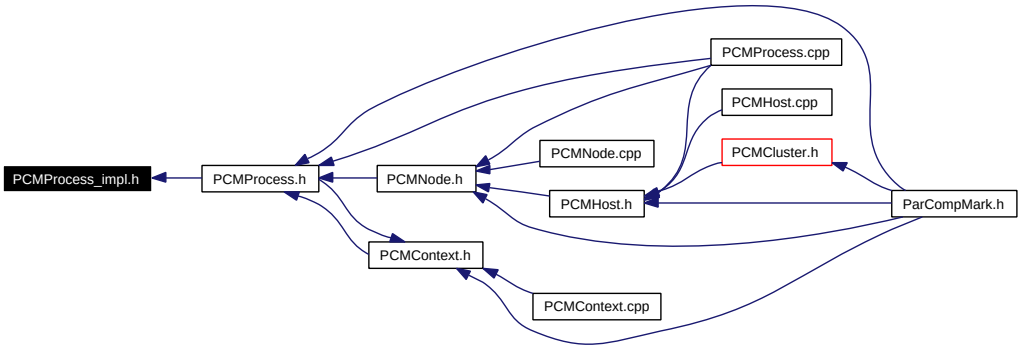
- namespace **ParCompMark**

Classes

- class **ParCompMark::Process**

8.71 PCMProcess_impl.h File Reference

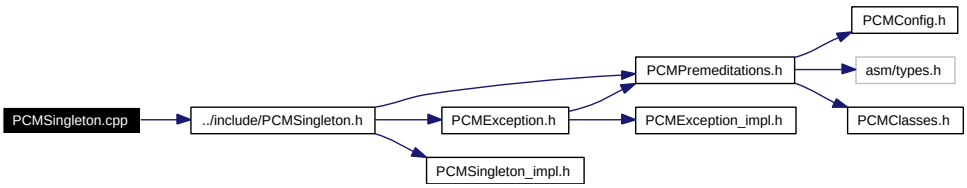
This graph shows which files directly or indirectly include this file:



8.72 PCMSingleton.cpp File Reference

```
#include "../include/PCMSingleton.h"
```

Include dependency graph for PCMSingleton.cpp:



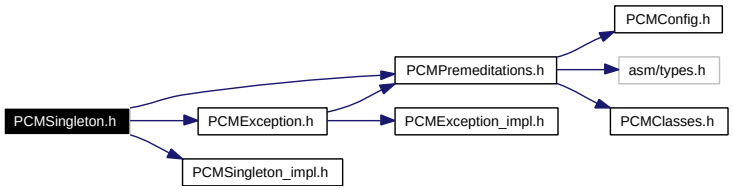
Namespaces

- namespace **ParCompMark**

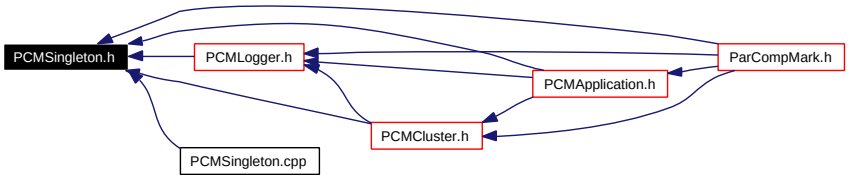
8.73 PCMSingleton.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMException.h"
#include "PCMSingleton_impl.h"
```

Include dependency graph for PCMSingleton.h:



This graph shows which files directly or indirectly include this file:



Namespaces

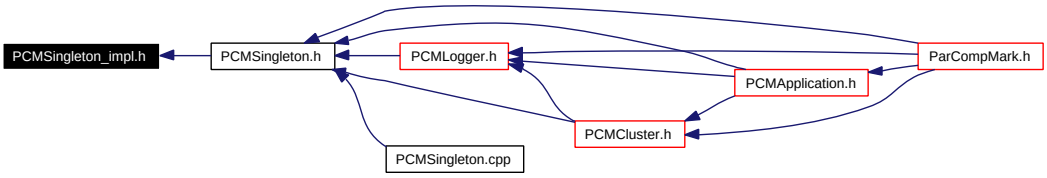
- namespace **ParCompMark**

Classes

- class **ParCompMark::Singleton**< **T** >

8.74 PCMSingleton_impl.h File Reference

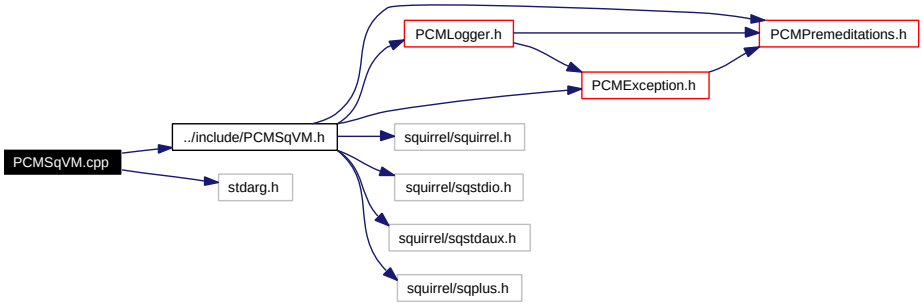
This graph shows which files directly or indirectly include this file:



8.75 PCMSqVM.cpp File Reference

```
#include "../include/PCMSqVM.h"
#include <stdarg.h>
```

Include dependency graph for PCMSqVM.cpp:



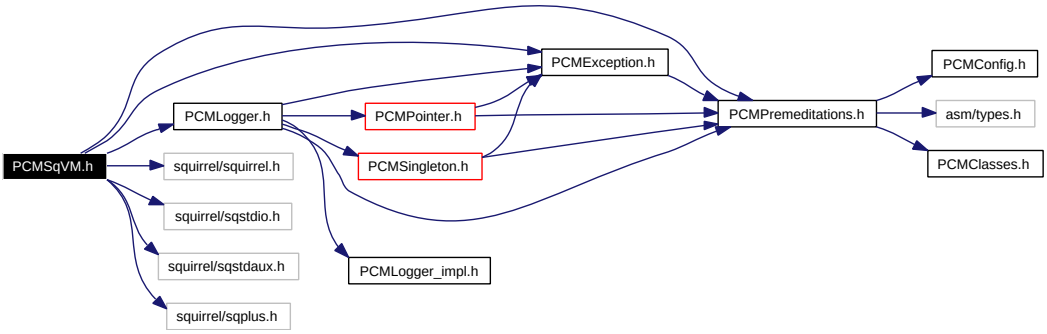
Namespaces

- namespace **ParCompMark**

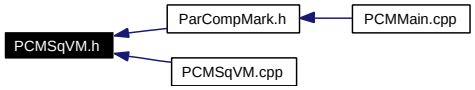
8.76 PCMSqVM.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMLogger.h"
#include "PCMException.h"
#include <squirrel/squirrel.h>
#include <squirrel/sqstdio.h>
#include <squirrel/sqstdaux.h>
#include <squirrel/sqplus.h>
```

Include dependency graph for PCMSqVM.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMark**

Classes

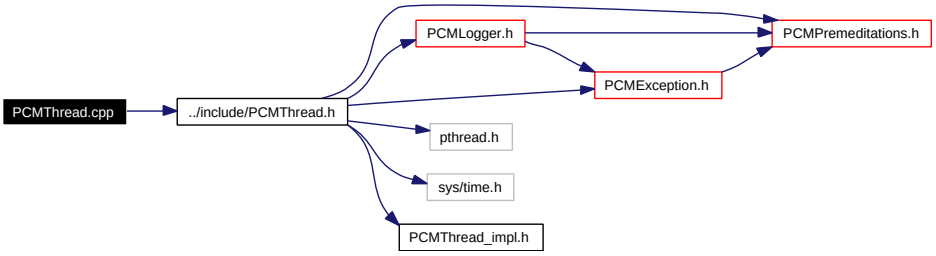
- class **ParCompMark::SqVM**

8.77 PCMSqVM_impl.h File Reference

8.78 PCMThread.cpp File Reference

```
#include "../include/PCMThread.h"
```

Include dependency graph for PCMThread.cpp:



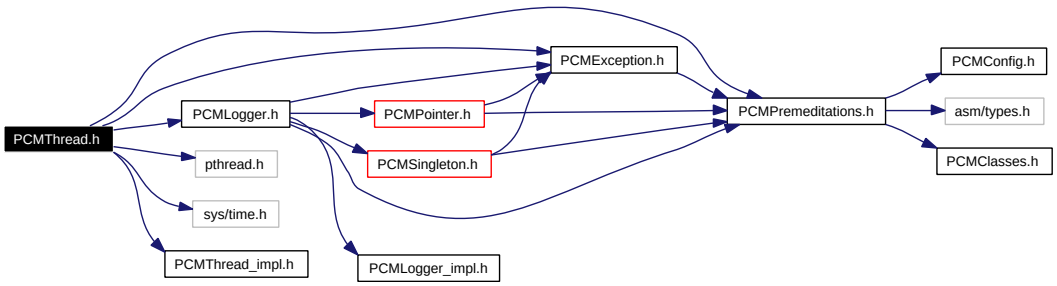
Namespaces

- namespace **ParCompMark**

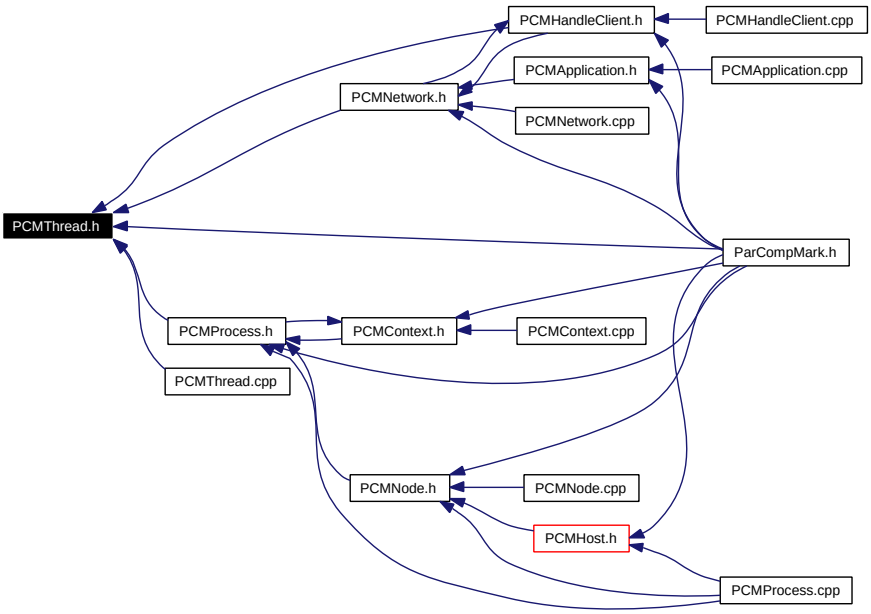
8.79 PCMThread.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMLogger.h"
#include "PCMException.h"
#include <pthread.h>
#include <sys/time.h>
#include "PCMThread_impl.h"
```

Include dependency graph for PCMThread.h:



This graph shows which files directly or indirectly include this file:



Namespaces

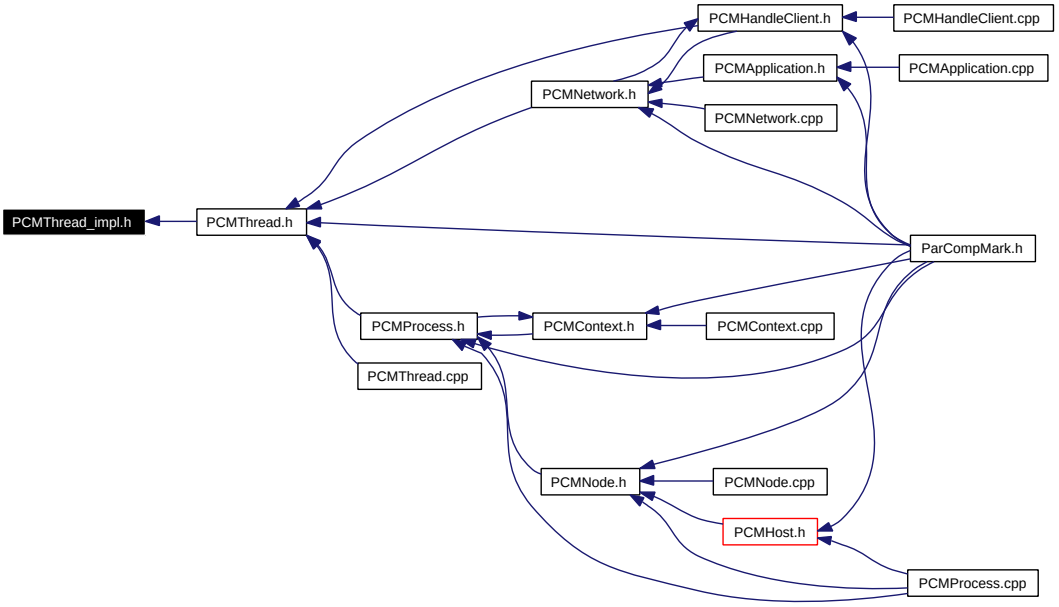
- namespace **ParCompMark**

Classes

- class **ParCompMark::Thread**

8.80 PCMThread_impl.h File Reference

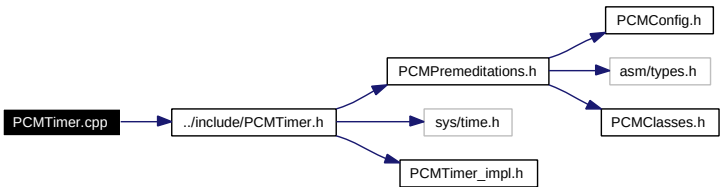
This graph shows which files directly or indirectly include this file:



8.81 PCMTimer.cpp File Reference

```
#include "../include/PCMTimer.h"
```

Include dependency graph for PCMTimer.cpp:



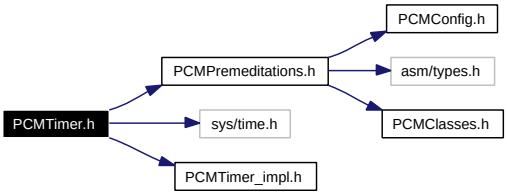
Namespaces

- namespace **ParCompMark**

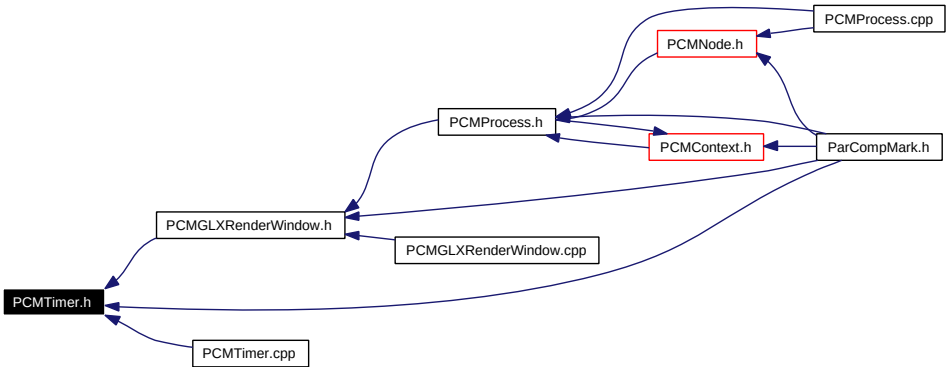
8.82 PCMTimer.h File Reference

```
#include "PCMPremeditations.h"
#include <sys/time.h>
#include "PCMTimer_impl.h"
```

Include dependency graph for PCMTimer.h:



This graph shows which files directly or indirectly include this file:



Namespaces

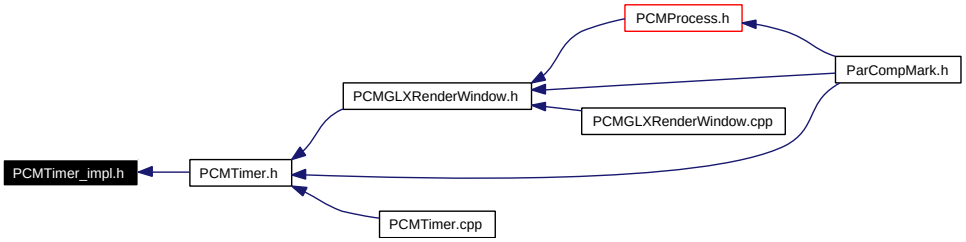
- namespace **ParCompMark**

Classes

- class **ParCompMark::Timer**

8.83 PCMTimer_impl.h File Reference

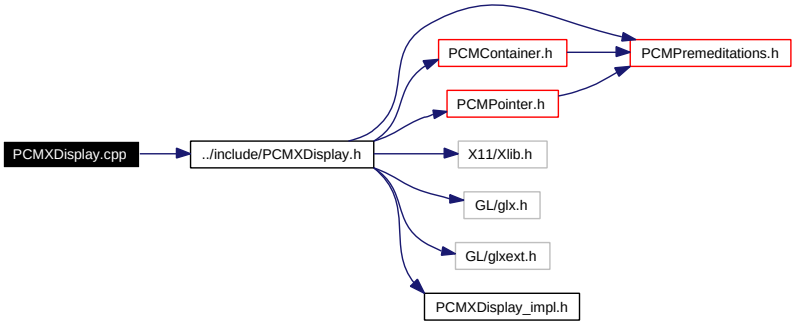
This graph shows which files directly or indirectly include this file:



8.84 PCMXDisplay.cpp File Reference

```
#include "../include/PCMXDisplay.h"
```

Include dependency graph for PCMXDisplay.cpp:



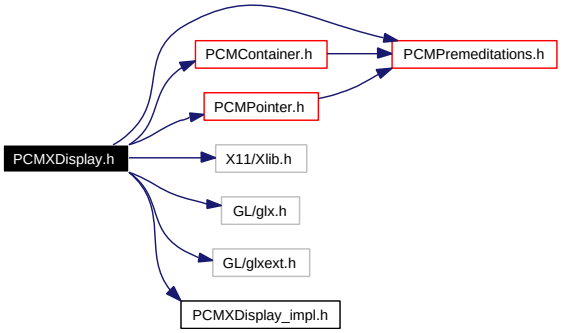
Namespaces

- namespace **ParCompMark**

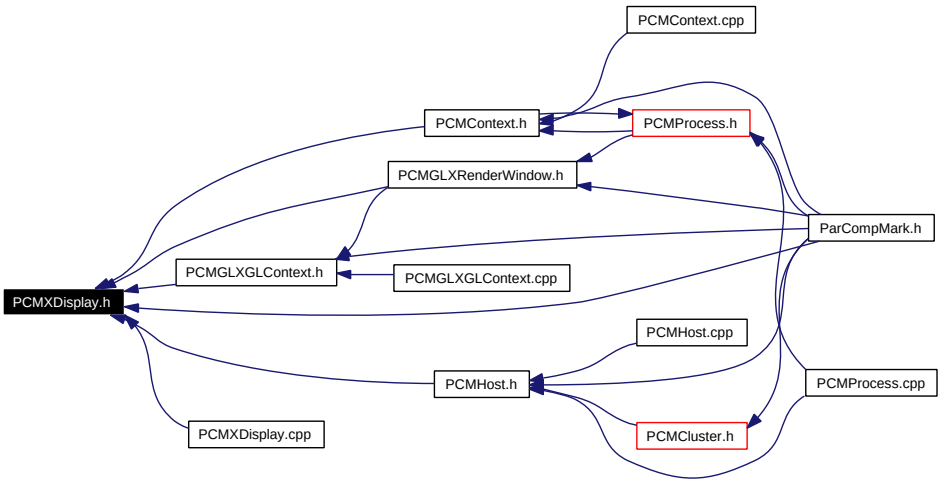
8.85 PCMXDisplay.h File Reference

```
#include "PCMPremeditations.h"
#include "PCMContainer.h"
#include "PCMPointer.h"
#include <X11/Xlib.h>
#include <GL/glx.h>
#include <GL/glxext.h>
#include "PCMXDisplay_impl.h"
```

Include dependency graph for PCMXDisplay.h:



This graph shows which files directly or indirectly include this file:



Namespaces

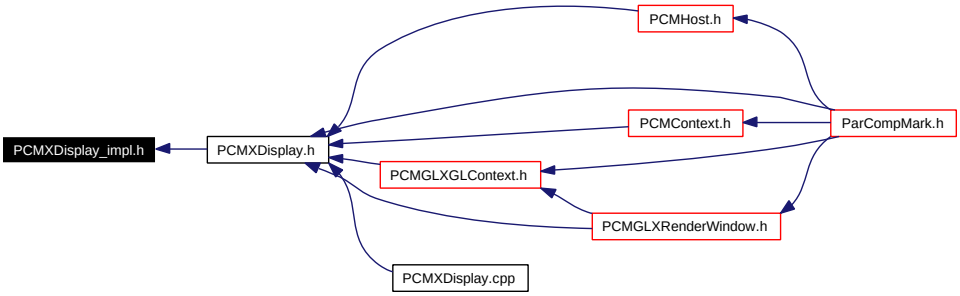
- namespace **ParCompMark**

Classes

- class **ParCompMark::XDisplay**
- struct **ParCompMark::XDisplay::VisualAttribs**

8.86 PCMXDisplay_impl.h File Reference

This graph shows which files directly or indirectly include this file:



8.87 TestApplication.cpp File Reference

```
#include "../include/TestApplication.h"
```

Include dependency graph for TestApplication.cpp:

Namespaces

- namespace **ParCompMarkTest**

Functions

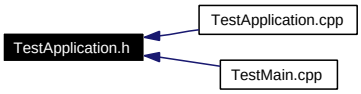
- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestApplication)

8.88 TestApplication.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMApplication.h"
```

Include dependency graph for TestApplication.h:

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestApplication**

Defines

- #define **PARCOMPMARK_TEST**

8.88.1 Define Documentation

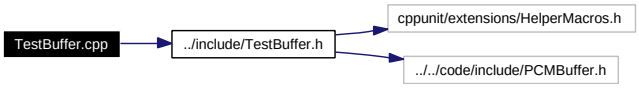
8.88.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestApplication.h.

8.89 TestBuffer.cpp File Reference

```
#include "../include/TestBuffer.h"
```

Include dependency graph for TestBuffer.cpp:



Namespaces

- namespace **ParCompMarkTest**

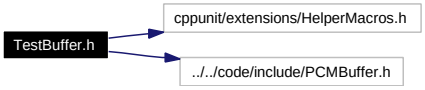
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestBuffer)

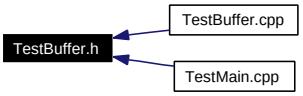
8.90 TestBuffer.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMBuffer.h"
```

Include dependency graph for TestBuffer.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestBuffer**

Defines

- #define **PARCOMPMARK_TEST**

8.90.1 Define Documentation

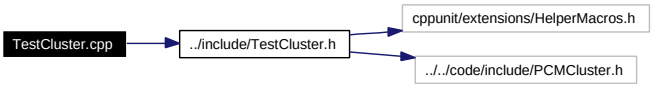
8.90.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestBuffer.h.

8.91 TestCluster.cpp File Reference

```
#include "../include/TestCluster.h"
```

Include dependency graph for TestCluster.cpp:



Namespaces

- namespace **ParCompMarkTest**

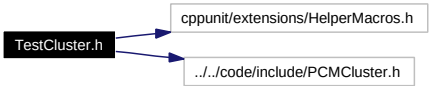
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestCluster)

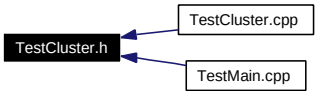
8.92 TestCluster.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMCluster.h"
```

Include dependency graph for TestCluster.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestCluster**

Defines

- #define **PARCOMPMARK_TEST**

8.92.1 Define Documentation

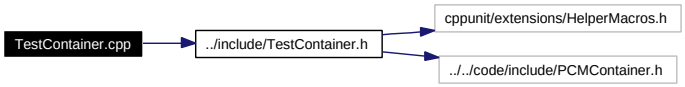
8.92.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestCluster.h.

8.93 TestContainer.cpp File Reference

```
#include "../include/TestContainer.h"
```

Include dependency graph for TestContainer.cpp:



Namespaces

- namespace **ParCompMarkTest**

Classes

- struct **ParCompMarkTest::MyClass**

Typedefs

- typedef **Container**< MyClass, **Mutex** > **ParCompMarkTest::MyContainer**

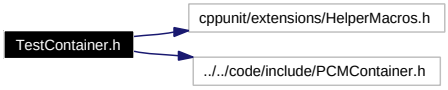
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestContainer)

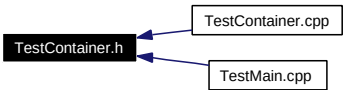
8.94 TestContainer.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMContainer.h"
```

Include dependency graph for TestContainer.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestContainer**

Defines

- #define **PARCOMPMARK_TEST**

8.94.1 Define Documentation

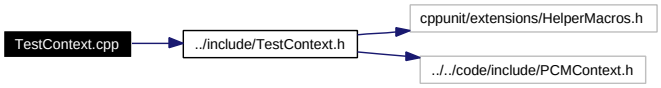
8.94.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestContainer.h.

8.95 TestContext.cpp File Reference

```
#include "../include/TestContext.h"
```

Include dependency graph for TestContext.cpp:



Namespaces

- namespace **ParCompMarkTest**

Functions

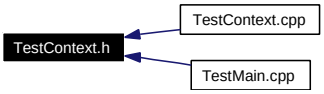
- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestContext)

8.96 TestContext.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMContext.h"
```

Include dependency graph for TestContext.h:

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestContext**

Defines

- #define **PARCOMPMARK_TEST**

8.96.1 Define Documentation

8.96.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestContext.h.

8.97 TestDummyLock.cpp File Reference

```
#include "../include/TestDummyLock.h"
```

Include dependency graph for TestDummyLock.cpp:

Namespaces

- namespace **ParCompMarkTest**

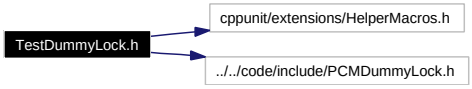
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestDummyLock)

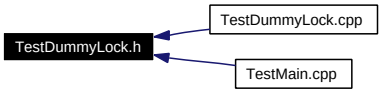
8.98 TestDummyLock.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMDummyLock.h"
```

Include dependency graph for TestDummyLock.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestDummyLock**

Defines

- #define **PARCOMPMARK_TEST**

8.98.1 Define Documentation

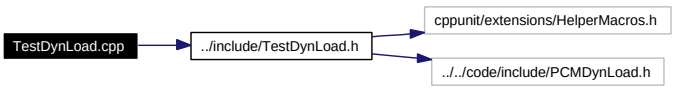
8.98.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestDummyLock.h.

8.99 TestDynLoad.cpp File Reference

```
#include "../include/TestDynLoad.h"
```

Include dependency graph for TestDynLoad.cpp:



Namespaces

- namespace **ParCompMarkTest**

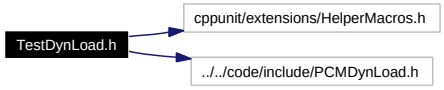
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestDynLoad)

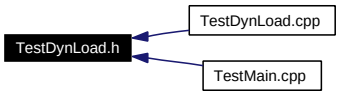
8.100 TestDynLoad.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMDynLoad.h"
```

Include dependency graph for TestDynLoad.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestDynLoad**

Defines

- #define **PARCOMPMARK_TEST**

8.100.1 Define Documentation

8.100.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestDynLoad.h.

8.101 TestException.cpp File Reference

```
#include "../include/TestException.h"
```

Include dependency graph for TestException.cpp:

Namespaces

- namespace **ParCompMarkTest**

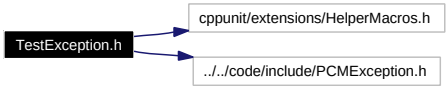
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestException)

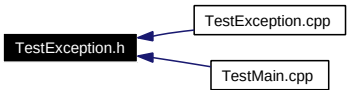
8.102 TestException.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMException.h"
```

Include dependency graph for TestException.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestException**

Defines

- #define **PARCOMPMARK_TEST**

8.102.1 Define Documentation

8.102.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestException.h.

8.103 TestGLXGLContext.cpp File Reference

```
#include "../include/TestGLXGLContext.h"
```

Include dependency graph for TestGLXGLContext.cpp:

Namespaces

- namespace **ParCompMarkTest**

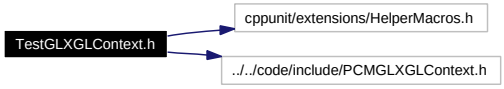
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestGLXGLContext)

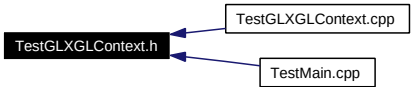
8.104 TestGLXGLContext.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMGLXGLContext.h"
```

Include dependency graph for TestGLXGLContext.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestGLXGLContext**

Defines

- #define **PARCOMPMARK_TEST**

8.104.1 Define Documentation

8.104.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestGLXGLContext.h.

8.105 TestGLXRenderWindow.cpp File Reference

```
#include "../include/TestGLXRenderWindow.h"
```

Include dependency graph for TestGLXRenderWindow.cpp:

Namespaces

- namespace **ParCompMarkTest**

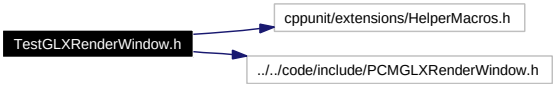
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestGLXRenderWindow)

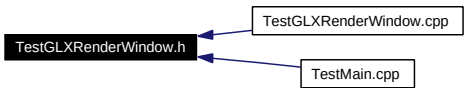
8.106 TestGLXRenderWindow.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMGLXRenderWindow.h"
```

Include dependency graph for TestGLXRenderWindow.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestGLXRenderWindow**

Defines

- #define **PARCOMPMARK_TEST**

8.106.1 Define Documentation

8.106.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestGLXRenderWindow.h.

8.107 TestHandleClient.cpp File Reference

```
#include "../include/TestHandleClient.h"
```

Include dependency graph for TestHandleClient.cpp:

Namespaces

- namespace **ParCompMarkTest**

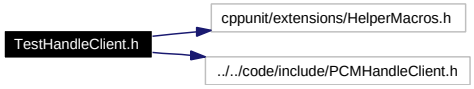
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestHandleClient)

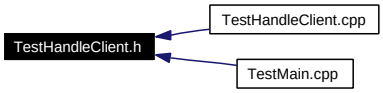
8.108 TestHandleClient.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMHandleClient.h"
```

Include dependency graph for TestHandleClient.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestHandleClient**

Defines

- #define **PARCOMPMARK_TEST**

8.108.1 Define Documentation

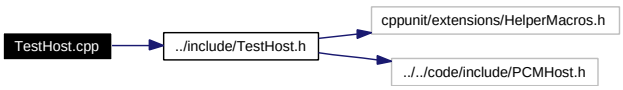
8.108.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestHandleClient.h.

8.109 TestHost.cpp File Reference

```
#include "../include/TestHost.h"
```

Include dependency graph for TestHost.cpp:



Namespaces

- namespace **ParCompMarkTest**

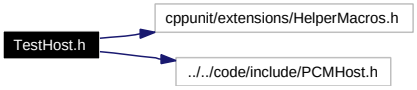
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestHost)

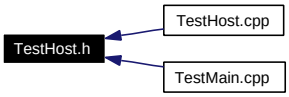
8.110 TestHost.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMHost.h"
```

Include dependency graph for TestHost.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestHost**

Defines

- #define **PARCOMPMARK_TEST**

8.110.1 Define Documentation

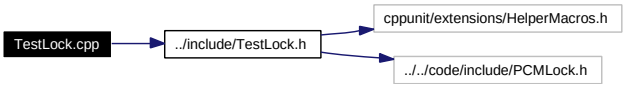
8.110.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestHost.h.

8.111 TestLock.cpp File Reference

```
#include "../include/TestLock.h"
```

Include dependency graph for TestLock.cpp:



Namespaces

- namespace **ParCompMarkTest**

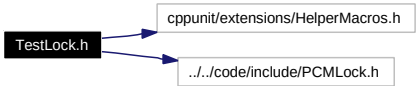
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestLock)

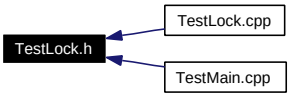
8.112 TestLock.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMLock.h"
```

Include dependency graph for TestLock.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestLock**

Defines

- #define **PARCOMPMARK_TEST**

8.112.1 Define Documentation

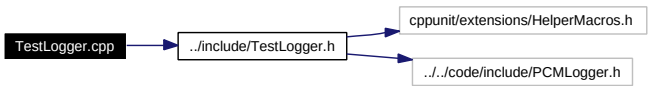
8.112.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestLock.h.

8.113 TestLogger.cpp File Reference

```
#include "../include/TestLogger.h"
```

Include dependency graph for TestLogger.cpp:



Namespaces

- namespace **ParCompMarkTest**

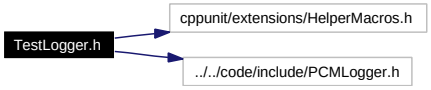
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestLogger)

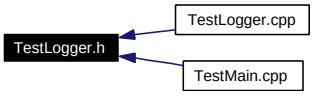
8.114 TestLogger.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMLogger.h"
```

Include dependency graph for TestLogger.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestLogger**

Defines

- #define **PARCOMPMARK_TEST**

8.114.1 Define Documentation

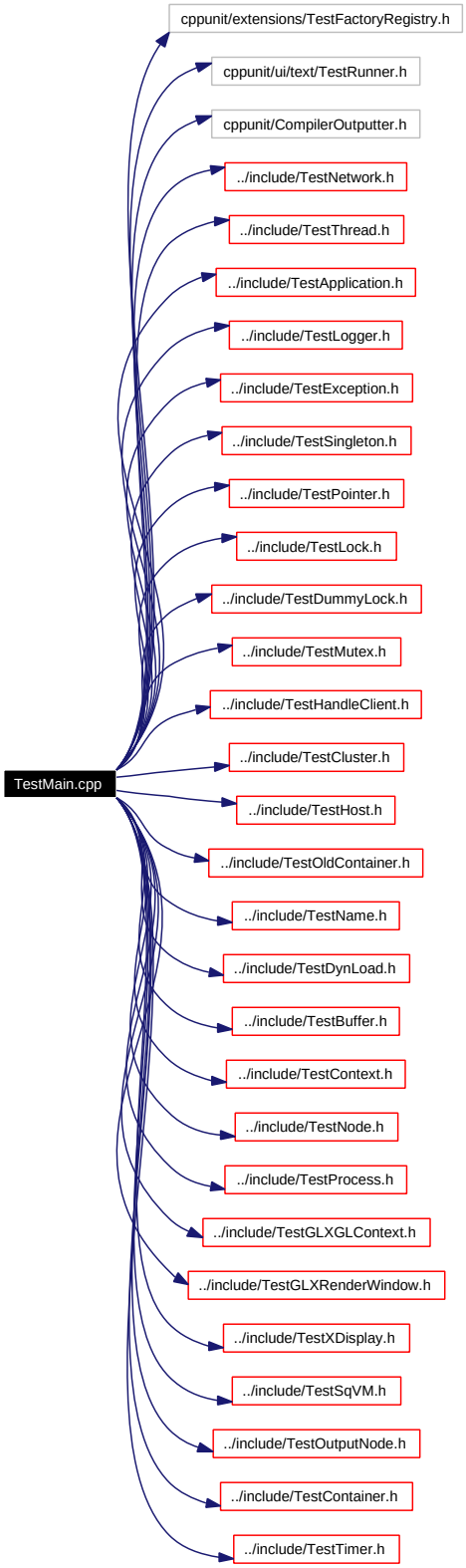
8.114.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestLogger.h.

8.115 TestMain.cpp File Reference

```
#include <cppunit/extensions/TestFactoryRegistry.h>
#include <cppunit/ui/text/TestRunner.h>
#include <cppunit/CompilerOutputter.h>
#include "../include/TestNetwork.h"
#include "../include/TestThread.h"
#include "../include/TestApplication.h"
#include "../include/TestLogger.h"
#include "../include/TestException.h"
#include "../include/TestSingleton.h"
#include "../include/TestPointer.h"
#include "../include/TestLock.h"
#include "../include/TestDummyLock.h"
#include "../include/TestMutex.h"
#include "../include/TestHandleClient.h"
#include "../include/TestCluster.h"
#include "../include/TestHost.h"
#include "../include/TestOldContainer.h"
#include "../include/TestName.h"
#include "../include/TestDynLoad.h"
#include "../include/TestBuffer.h"
#include "../include/TestContext.h"
#include "../include/TestNode.h"
#include "../include/TestProcess.h"
#include "../include/TestGLXGLContext.h"
#include "../include/TestGLXRenderWindow.h"
#include "../include/TestXDisplay.h"
#include "../include/TestSqVM.h"
#include "../include/TestOutputNode.h"
#include "../include/TestContainer.h"
#include "../include/TestTimer.h"
```

Include dependency graph for TestMain.cpp:


cppunit/extensions/TestFactoryRegistry.h
cppunit/ui/text/TestRunner.h
cppunit/CompilerOutputter.h
../include/TestNetwork.h
../include/TestThread.h
../include/TestApplication.h
../include/TestLogger.h
../include/TestException.h
../include/TestSingleton.h
../include/TestPointer.h
../include/TestLock.h
../include/TestDummyLock.h
../include/TestMutex.h
../include/TestHandleClient.h
../include/TestCluster.h
TestMain.cpp
../include/TestHost.h
../include/TestOldContainer.h
../include/TestName.h
../include/TestDynLoad.h
../include/TestBuffer.h
../include/TestContext.h
../include/TestNode.h
../include/TestProcess.h
../include/TestGLXGLContext.h
../include/TestGLXRenderWindow.h
../include/TestXDisplay.h
../include/TestSqVM.h
../include/TestOutputNode.h
../include/TestContainer.h
../include/TestTimer.h

Functions

- int **textRunner** ()
- int **main** (int argc, char ∗∗argv)

8.115.1 Function Documentation

8.115.1.1 int main (int *argc*, char ∗∗ *argv*)

Remarks:

Runs CppUnit tests.

Tested classes:

- **ParCompMark::Network**(p. 134)
- **ParCompMark::Thread**(p. 264)
- **ParCompMark::Application**(p. 21)
- **ParCompMark::Logger**(p. 118)
- **ParCompMark::Exception**(p. 73)
- **ParCompMark::Singleton**(p. 194)
- **ParCompMark::Pointer**(p. 170)
- **ParCompMark::Lock**(p. 116)
- **ParCompMark::DummyLock**(p. 67)
- **ParCompMark::Mutex**(p. 127)
- **ParCompMark::HandleClient**(p. 107)
- **ParCompMark::Cluster**(p. 46)
- **ParCompMark::Host**(p. 112)
- **ParCompMark::OldContainer**(p. 154)
- **ParCompMark::Name**(p. 131)
- **ParCompMark::DynLoad**(p. 69)
- **ParCompMark::Buffer**(p. 39)
- **ParCompMark::Context**(p. 53)
- **ParCompMark::Node**(p. 150)
- **ParCompMark::Process**(p. 182)
- **ParCompMark::GLXGLContext**(p. 79)
- **ParCompMark::GLXRenderWindow**(p. 85)
- **ParCompMark::XDisplay**(p. 276)
- **ParCompMark::SqVM**(p. 196)
- **ParCompMark::OutputNode**(p. 159)
- **ParCompMark::Container**(p. 49)
- **ParCompMark::Timer**(p. 275)

Definition at line 173 of file TestMain.cpp.

References textRunner().

Here is the call graph for this function:

8.115.1.2 int textRunner ()

Test runner with textual user interface

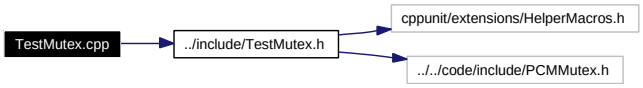
Definition at line 92 of file TestMain.cpp.

Referenced by main().

8.116 TestMutex.cpp File Reference

```
#include "../include/TestMutex.h"
```

Include dependency graph for TestMutex.cpp:



Namespaces

- namespace **ParCompMarkTest**

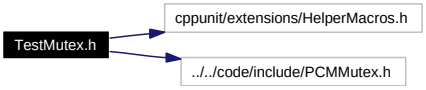
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestMutex)

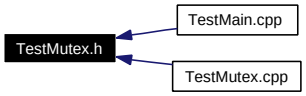
8.117 TestMutex.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMMutex.h"
```

Include dependency graph for TestMutex.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestMutex**

Defines

- #define **PARCOMPMARK_TEST**

8.117.1 Define Documentation

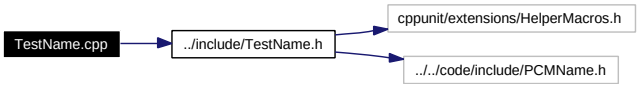
8.117.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestMutex.h.

8.118 TestName.cpp File Reference

```
#include "../include/TestName.h"
```

Include dependency graph for TestName.cpp:



Namespaces

- namespace **ParCompMarkTest**

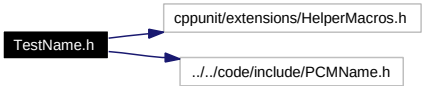
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestName)

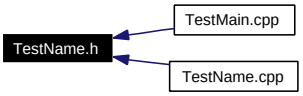
8.119 TestName.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMName.h"
```

Include dependency graph for TestName.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestName**

Defines

- #define **PARCOMPMARK_TEST**

8.119.1 Define Documentation

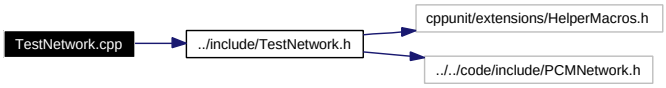
8.119.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestName.h.

8.120 TestNetwork.cpp File Reference

```
#include "../include/TestNetwork.h"
```

Include dependency graph for TestNetwork.cpp:



Namespaces

- namespace **ParCompMarkTest**

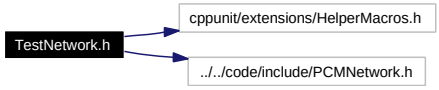
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestNetwork)

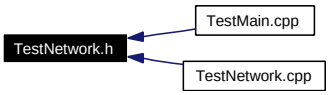
8.121 TestNetwork.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMNetwork.h"
```

Include dependency graph for TestNetwork.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestNetwork**

Defines

- #define **PARCOMPMARK_TEST**

8.121.1 Define Documentation

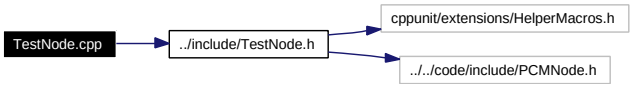
8.121.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestNetwork.h.

8.122 TestNode.cpp File Reference

```
#include "../include/TestNode.h"
```

Include dependency graph for TestNode.cpp:



Namespaces

- namespace **ParCompMarkTest**

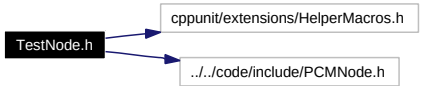
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestNode)

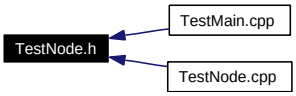
8.123 TestNode.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMNode.h"
```

Include dependency graph for TestNode.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestNode**

Defines

- #define **PARCOMPMARK_TEST**

8.123.1 Define Documentation

8.123.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestNode.h.

8.124 TestOldContainer.cpp File Reference

```
#include "../include/TestOldContainer.h"
```

Include dependency graph for TestOldContainer.cpp:

Namespaces

- namespace **ParCompMarkTest**

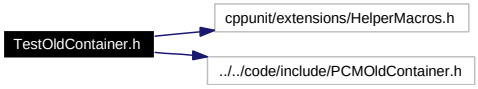
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestOldContainer)

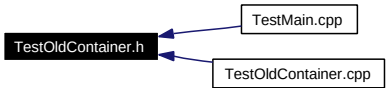
8.125 TestOldContainer.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMOldContainer.h"
```

Include dependency graph for TestOldContainer.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestOldContainer**

Defines

- #define **PARCOMPMARK_TEST**

8.125.1 Define Documentation

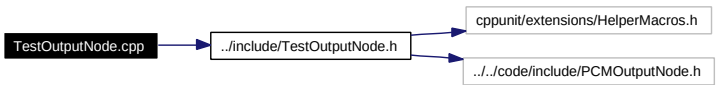
8.125.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestOldContainer.h.

8.126 TestOutputNode.cpp File Reference

```
#include "../include/TestOutputNode.h"
```

Include dependency graph for TestOutputNode.cpp:



Namespaces

- namespace **ParCompMarkTest**

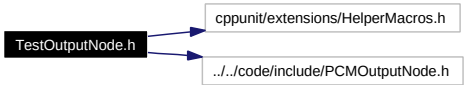
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestOutputNode)

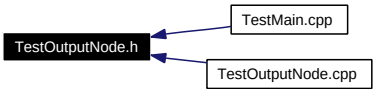
8.127 TestOutputNode.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMOutputNode.h"
```

Include dependency graph for TestOutputNode.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestOutputNode**

Defines

- #define **PARCOMPMARK_TEST**

8.127.1 Define Documentation

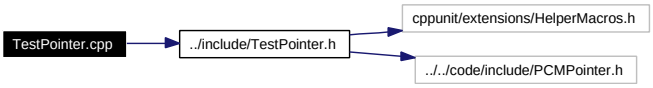
8.127.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestOutputNode.h.

8.128 TestPointer.cpp File Reference

```
#include "../include/TestPointer.h"
```

Include dependency graph for TestPointer.cpp:



Namespaces

- namespace **ParCompMarkTest**

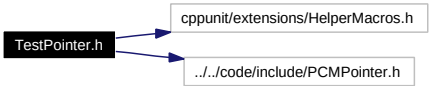
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestPointer)

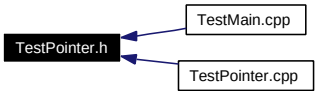
8.129 TestPointer.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMPointer.h"
```

Include dependency graph for TestPointer.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestPointer**

Defines

- #define **PARCOMPMARK_TEST**

8.129.1 Define Documentation

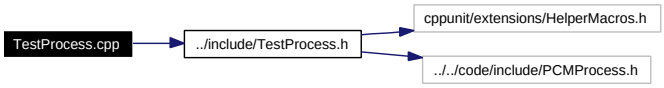
8.129.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestPointer.h.

8.130 TestProcess.cpp File Reference

```
#include "../include/TestProcess.h"
```

Include dependency graph for TestProcess.cpp:



Namespaces

- namespace **ParCompMarkTest**

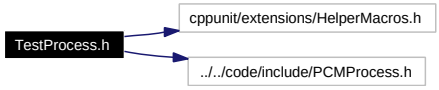
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestProcess)

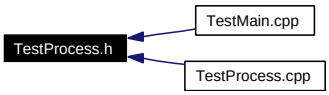
8.131 TestProcess.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMProcess.h"
```

Include dependency graph for TestProcess.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestProcess**

Defines

- #define **PARCOMPMARK_TEST**

8.131.1 Define Documentation

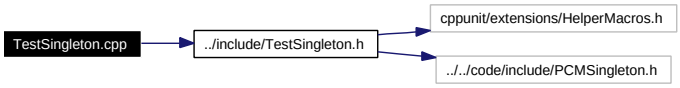
8.131.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestProcess.h.

8.132 TestSingleton.cpp File Reference

```
#include "../include/TestSingleton.h"
```

Include dependency graph for TestSingleton.cpp:



Namespaces

- namespace **ParCompMarkTest**

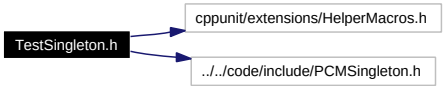
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestSingleton)

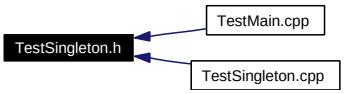
8.133 TestSingleton.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMSingleton.h"
```

Include dependency graph for TestSingleton.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestSingleton**

Defines

- #define **PARCOMPMARK_TEST**

8.133.1 Define Documentation

8.133.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestSingleton.h.

8.134 TestSqVM.cpp File Reference

```
#include "../include/TestSqVM.h"
```

Include dependency graph for TestSqVM.cpp:

Namespaces

- namespace **ParCompMarkTest**

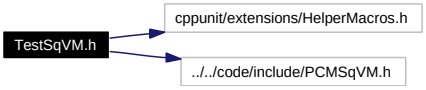
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestSqVM)

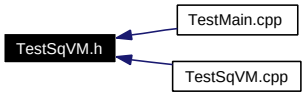
8.135 TestSqVM.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMSqVM.h"
```

Include dependency graph for TestSqVM.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestSqVM**

Defines

- #define **PARCOMPMARK_TEST**

8.135.1 Define Documentation

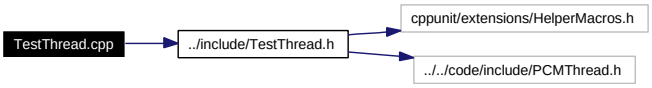
8.135.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestSqVM.h.

8.136 TestThread.cpp File Reference

```
#include "../include/TestThread.h"
```

Include dependency graph for TestThread.cpp:



Namespaces

- namespace **ParCompMarkTest**

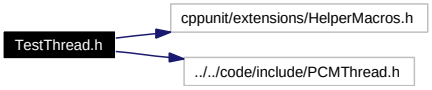
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestThread)

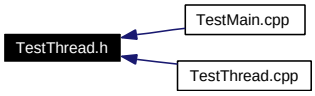
8.137 TestThread.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMThread.h"
```

Include dependency graph for TestThread.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestThread**

Defines

- #define **PARCOMPMARK_TEST**

8.137.1 Define Documentation

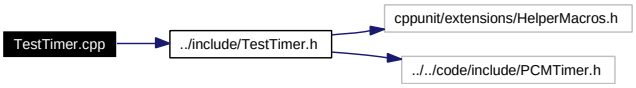
8.137.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestThread.h.

8.138 TestTimer.cpp File Reference

```
#include "../include/TestTimer.h"
```

Include dependency graph for TestTimer.cpp:



Namespaces

- namespace **ParCompMarkTest**

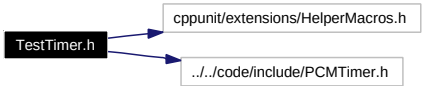
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestTimer)

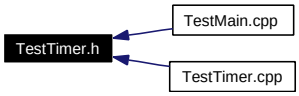
8.139 TestTimer.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMTimer.h"
```

Include dependency graph for TestTimer.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestTimer**

Defines

- #define **PARCOMPMARK_TEST**

8.139.1 Define Documentation

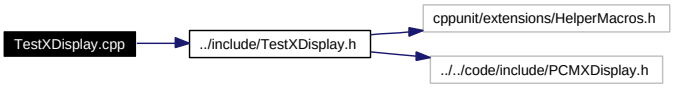
8.139.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestTimer.h.

8.140 TestXDisplay.cpp File Reference

```
#include "../include/TestXDisplay.h"
```

Include dependency graph for TestXDisplay.cpp:



Namespaces

- namespace **ParCompMarkTest**

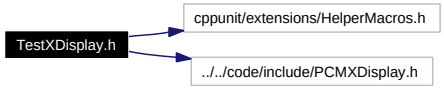
Functions

- **ParCompMarkTest::CPPUNIT_TEST_SUITE_REGISTRATION** (TestXDisplay)

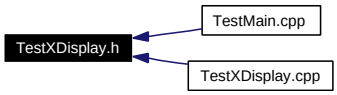
8.141 TestXDisplay.h File Reference

```
#include <cppunit/extensions/HelperMacros.h>
#include "../../code/include/PCMXDisplay.h"
```

Include dependency graph for TestXDisplay.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **ParCompMarkTest**

Classes

- class **ParCompMarkTest::TestXDisplay**

Defines

- #define **PARCOMPMARK_TEST**

8.141.1 Define Documentation

8.141.1.1 #define PARCOMPMARK_TEST

Definition at line 36 of file TestXDisplay.h.

Index