SFML

* [Main Page](http://docs.google.com/index.htm)
* [Modules](http://docs.google.com/modules.htm)
* [Classes](http://docs.google.com/annotated.htm)
* [Files](http://docs.google.com/files.htm)
* [File List](http://docs.google.com/files.htm)
* [include](http://docs.google.com/dir_f3190241575fd2bd132a392ae6942f4a.htm)
* [SFML](http://docs.google.com/dir_692f376662c82a26cfe4cfa3aceebe24.htm)
* [System](http://docs.google.com/dir_60c5c649f8df3b69a45a020d59f81335.htm)

Thread.inl

1

2 //

3 // SFML - Simple and Fast Multimedia Library

4 // Copyright (C) 2007-2012 Laurent Gomila (laurent.gom@gmail.com)

5 //

6 // This software is provided 'as-is', without any express or implied warranty.

7 // In no event will the authors be held liable for any damages arising from the use of this software.

8 //

9 // Permission is granted to anyone to use this software for any purpose,

10 // including commercial applications, and to alter it and redistribute it freely,

11 // subject to the following restrictions:

12 //

13 // 1. The origin of this software must not be misrepresented;

14 // you must not claim that you wrote the original software.

15 // If you use this software in a product, an acknowledgment

16 // in the product documentation would be appreciated but is not required.

17 //

18 // 2. Altered source versions must be plainly marked as such,

19 // and must not be misrepresented as being the original software.

20 //

21 // 3. This notice may not be removed or altered from any source distribution.

22 //

24

25 namespace priv

26 {

27 // Base class for abstract thread functions

28 struct ThreadFunc

29 {

30  virtual ~ThreadFunc() {}

31  virtual void run() = 0;

32 };

33

34 // Specialization using a functor (including free functions) with no argument

35 template <typename T>

36 struct ThreadFunctor : ThreadFunc

37 {

38  ThreadFunctor(T functor) : m\_functor(functor) {}

39  virtual void run() {m\_functor();}

40  T m\_functor;

41 };

42

43 // Specialization using a functor (including free functions) with one argument

44 template <typename F, typename A>

45 struct ThreadFunctorWithArg : ThreadFunc

46 {

47  ThreadFunctorWithArg(F function, A arg) : m\_function(function), m\_arg(arg) {}

48  virtual void run() {m\_function(m\_arg);}

49  F m\_function;

50  A m\_arg;

51 };

52

53 // Specialization using a member function

54 template <typename C>

55 struct ThreadMemberFunc : ThreadFunc

56 {

57  ThreadMemberFunc(void(C::\*function)(), C\* object) : m\_function(function), m\_object(object) {}

58  virtual void run() {(m\_object->\*m\_function)();}

59  void(C::\*m\_function)();

60  C\* m\_object;

61 };

62

63 } // namespace priv

64

65

67 template <typename F>

68 Thread::Thread(F functor) :

69 m\_impl (NULL),

70 m\_entryPoint(new priv::ThreadFunctor<F>(functor))

71 {

72 }

73

74

76 template <typename F, typename A>

77 [Thread::Thread](http://docs.google.com/classsf_1_1Thread.htm#a4cc65399bbb111cf8132537783b8e96c)(F function, A argument) :

78 m\_impl (NULL),

79 m\_entryPoint(new priv::ThreadFunctorWithArg<F, A>(function, argument))

80 {

81 }

82

83

85 template <typename C>

86 [Thread::Thread](http://docs.google.com/classsf_1_1Thread.htm#a4cc65399bbb111cf8132537783b8e96c)(void(C::\*function)(), C\* object) :

87 m\_impl (NULL),

88 m\_entryPoint(new priv::ThreadMemberFunc<C>(function, object))

89 {

90 }

Copyright � Laurent Gomila  ::  Documentation generated by [doxygen](http://www.doxygen.org/)  ::