ITR

Generated by Doxygen 1.8.13

Contents

Chapter 1

ITR

Source code for a real-time computer science course.

Repository organization

The exercices guide the development of functions and classes to manage the real-time execution of tasks on a Linux platform. In this repository we gathered them in a static library called `itr`. It is used by several example programs that can be compiled and run to demonstrate the lib features.

- `includes/` contains the header files of the library.
- `src/` contains the source files of both the library functions and classes, and the example programs.
- `doc/` contains the project documentation generated with Doxygen.
- `lib/` contains the static library
- `build/` contains the object files
- `bin/` contains the examples binaries

The `Makefile` allows oneself to compile the static library, the examples and the documentation.

Usage

```
make
```

Build the documentation

```
make doc
```

to generate HTML and Latex Doxygen documentation for the project in the `doc/` folder.

It will load the produced html page on your default browser (`doc/html/index.html`).

Documentation

- TD1
- TD2
- TD3
- TD4
- TD6

Chapter 2

1.

2.1 1.a

2.1.1 Summary

Implement utils functions to ease the manipulations of the POSIX `timespec` struct.

2.1.2 Contents

We first define functions to convert and convert back such a struct into a number (`double`) of milliseconds, easier to represent. We need special attention for the negative time case. Indeed the specification of the `timespec` struct imposes its nanosecond fields to be always positive.

We then define functions to add and substract two timespecs, and get the current time. We use them to override the + and − operator. Comparisons operators are overriden as well.

Finally, we define a function to make the current thread wait for a certain time. Note that the kernel scheduler will try to wait the specified time but can return sooner (case of an interruption, round-robin task scheduling etc.). So the functions returns the remaining time to wait if it was interrupted.

We tested these functions, especially the negative time mechanisms, in the `main.cpp` file.

2.1.3 Related files

- src/td1/a/main.cpp
- src/td1/a/time.cpp

2.2 1.b

2.2.1 Summary

Demonstrate the POSIX timer usage.

2.2.2 Contents

We initializes a `counter` integer to `0`. We declare it as `volatile` so the compiler won't try to deduce its value from the `main` following code (indeed the variable is never modified in it so it could chose to bypass the loops that come later on). We create a POSIX timer and configure it to run the `handler` function every half second. This function increments the `counter` by one.

Once the timer is set and started, a loop blocks the main thread until it reaches `15`. Then the timer is deleted and the program exits.

2.2.3 Related files

- src/td1/b/main.cpp

2.3 1.c

2.3.1 Summary

Demonstrate how to measure a function execution time.

2.3.2 Contents

We define a function to increments a variable until it reaches a defined value.

In the main function we get the current time and save it in a variable, execute the function, get the current time after it finished, compute the difference with the previous time and print it. This is considered as the function execution time.

2.3.3 Related files

- src/td1/c/main.cpp

2.4 1.d

2.4.1 Summary

Find a relation between a functions number of iterations and its execution time.

2.4.2 Contents

We use the previous function with a subtil change, we check at each iteration the value of a boolean value (passed to the function through a pointer). If it is false, the function returns immediatly. This boolean variable is defined as volatile to avoid the compiler optimizations. The loop function returns the number of iterations it actually made.

Our goal is now to obtain the affine parameters of the function describing the number of loop iterations depending on the time. These parameter are returns in the form of a struct (`Parameters`) by the function `calibrate`. This function get two samples (a tuple: (second, iterations number)) for two different times, 4 and 6 seconds. Then it computes a linear regression for these two points. This gives the parameters.

2.4.3 Related files

- src/td1/d/main.cpp

2.5 1.e

2.5.1 Summary

Improvements to the previous exercice.

2.5.2 Contents

We use the previous mechanism but get a lot more samples, and perform a least square fit linear regression to compute the affine parameters.

2.5.3 Related files

- src/td1/e/main.cpp

Chapter 3

2.

3.1 2.a

3.1.1 Summary

Demonstrate race conditions effects.

3.1.2 Contents

We create 10 threads and make them increment a counter value passed through a pointer. We can see that the final counter value does not match the expected value (it should be 10 times the iterations number). It is because there is a race condition between the threads since they write a variable at the same time. We can see that each run yields a different counter value.

3.1.3 Related files

- src/td2/a/main.cpp

3.2 2.b

3.2.1 Summary

Demonstrate the threads number effect.

3.2.2 Contents

We reuse the previous code but set different scheduling methods. Finally we try several threads and iterations numbers for the `SCHED_RR` scheduling. It gives us the following graph:

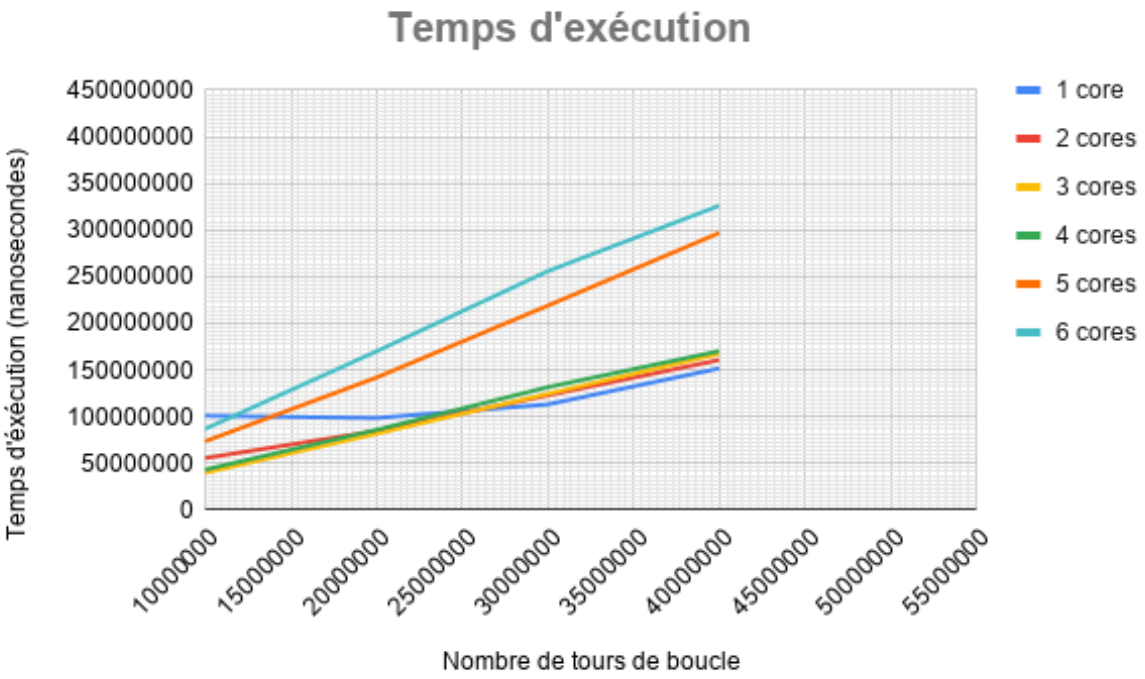


Figure 3.1 Temps d'exécution

We observe that for every thread number, the execution is almost linear depending in the iterations number. However, the curves for `5` and `6` threads are separated from the others. It is normal : the machine the program was runned on has `4` cores. With more threads than cores, the scheduler has to perform multi-threaded computation on a same core which takes more time.

3.2.3 Related files

- src/td2/b/main.cpp

3.3 2.c

3.3.1 Summary

Demonstrate the use of a POSIX mutex.

3.3.2 Contents

We reuse the counter incrementation program but define a POSIX mutex that protects the access to the incremented variable. The execution time is slightly longer for each run but the final value of the variable is always the same and matches the number of threads times the number of iterations performed in each thread.

3.3.3 Related files

- src/td2/c/main.cpp

Chapter 4

3.

4.1 3.a

4.1.1 Summary

Implement a Chrono class.

4.1.2 Contents

We define a Chrono class with utils methods to start, stop and get an elapsed time. We use the following convention, if the `stopTime` is `0` then it means the Chrono is running (since the `stopTime` is less than the `startTime`). The startTime is set to the current time at object instanciation but is updated at each `restart` call. The elapsed time returned by the `lap` method is the `stopTime` minus the `startTime` if the Chrono is stopped, and the current time minus the `startTime` if it is running.

The class is tested in `main.ccp` with a few test functions, asserting its laps values in different configurations.

4.1.3 Related files

- src/td3/a/main.cpp
- src/td3/a/Chrono.cpp

4.2 3.b

4.2.1 Summary

Implement a Timer class.

4.2.2 contents

We define a Timer class that embeds a POSIX timer. Obviously, the Timer constructor and destructor are public. It has two other public methods `start` and `stop`. They are accessible to the user since it is the ones used to control the Timer. However, the class is an abstract one. It means it cannot be constructed itself, but through child classes that implement its virtual and protected method `callback`. This method is the one executed when the timer expires. It is protected so only child classes can access it. However, the `start` methods needs to pass a pointer to this function to the POSIX timer struct. It is not possible if it is implemented by a child function. To overcome that, we define a private static method whose job is to call the virtual child implemeted method. The static method has a well-known address (which does not depend on the child implementation) that can be passed to the timer POSIX struct.

We define another abstract function that inherits from Timer : PeriodicTimer. Its definition just overrides the `start` method to configure the POSIX timer to execute the callback method at each period and only at expiration.

We tested this set of classes through the CountDown class that inherits from PeriodicTimer. It implements `callback` with a simple mecanism: it decrements an attribute value and print out its value. We create a class instance in `main.cpp` (it is not abstract) and call its `start` method to run it with a period of 100ms. We use our timespec_wait function to block the main thread a certain time (1s) and so wait for the CountDown object counter's value to reach 0.

4.2.3 Related files

- src/td3/b/main.cpp
- src/td3/b/CountDown.cpp
- src/td3/b/Timer.cpp
- src/td3/b/PeriodicTimer.cpp

4.3 3.c

4.3.1 Summary

Find a relation between a functions number of iterations and its execution time with an object oriented architecture.

4.3.2 Contents

We want to recode 1.d with classes.

The Looper class runs a loop and yields at will (through its method `getSample`) the number of iterations it has done. It can be stopped at will as well with `stopLoop`.

The Calibrator class inherits from PeriodicTimer. It initializes a Looper and get a certain amount of samples every period, then performing a linear regression (least square fit) over them to get the affine parameters of the function giving the number of iterations of the Looper function of the time. Once the Calibrator has been initialized, it can be passed as an argument to the CpuLoop constructor. This class inherits from Looper so it has the same behavior. But it is also time controllable : since the Looper was calibrated, we can compute the number of iterations for a certain time and execute them. We are sure that it will run the correct time with a certain accuracy.

4.3.3 Related files

- src/td3/c/main.cpp
- src/td3/c/CpuLoop.cpp
- src/td3/c/Looper.cpp

Chapter 5

4.

5.1 4.a

5.1.1 Summary

Implement a PosixThread and Thread classes.

5.1.2 Contents

We define a PosixThread class that embeds a POSIX thread management. The two constructors allow to instanciate an object from scratch or from an existing thread. Methods are defined to update or read the thread scheduling policy, start it, wait for its end, or wait for its end with a timeout.

The Thread class inherits from this first class. It uses the same mechanism as in the Timer class to be able to pass a static method pointer to the thread configuration POSIX tructure, that calls the virtual method `run` implemented by child classes. Methods enables the time measurement of the thread execution time.

We tested the Thread class in `main.ccp`. The Worker class inherits from it and run a loop that increments an attribute. We create several Workers objects (stored in a vector), start them and then wait for them to end. We print the final results and see that all their incremented attributes have been fully incremented to the iterations number.

5.1.3 Related files

- src/td4/a/main.cpp
- src/td4/a/PosixThread.cpp
- src/td4/a/Thread.cpp
- src/td4/a/Worker.cpp

5.2 4.b

5.2.1 Summary

Implement a Mutex class.

5.2.2 Contents

We implement a Mutex class that embeds a POSIX mutex. It is not designed to be controlled directly so all its methods that handle locking and unlocking are made private. Then we define the subclass Monitor that exposes public methods to wait (with or without a timeout) that a Mutex (whose reference is stored as an attribute) is free, and notify the watchers for these mutex. One interesting point is that since Monitor is a subclass of Mutex it has access to its private methods. Two other subclasses Lock and TryLock inherit from Monitor. At instanciation they lock or try to lock with a timeout a Mutex, and they unlock it within their destructor.

An example of use is given in `main.cpp`. We create a pool of workers that inherit from Thread and increments a shared value protected by a Mutex. In the end, the variable value has a correct value 100% of the time : the mutex protected it from race conditions.

5.2.3 Related files

- src/td4/b/main.cpp
- src/td4/b/Mutex.cpp
- src/td4/b/Worker.cpp

5.3 4.c

5.3.1 Summary

Implement a Semaphore class.

5.3.2 Contents

We define a Semaphore class. It is composed of a token counter capped by a maximum value and protected with a Mutex. It exposes a `give` and `take` methods that increment or decrement this counter. If it reached `0` the `take` method will be blocking until a token is added to it with a call to `give`. An overload of `take` with a timeout exists. It returns a bool corresponsing to `true` if it achieved to get a token in time and `false` else. This is why we use `notify` at the end of the counter incrementation in `give`, to notify eventually waiting threads that a token was added.

We tested it in `main.cpp` by creating a Semaphore and two types of workers : Producer that gives tokens and Consumer that takes tokens.

We create a pool of each of these objects, start them in their own threads (they inherit from Thread) and check that after the all ended, the consumers consumed all the tokens. In order to do that, we try to take from the Semaphore with a timeout and check it returns `false`.

5.3.3 Related files

- src/td4/c/main.cpp
- src/td4/c/Semaphore.cpp
- src/td4/c/Consumer.cpp
- src/td4/a/Producer.cpp

5.4 4.d

5.4.1 Summary

Implement a FIFO class template.

5.4.2 Contents

We define a template that stores values in a standard library queue. The queue is protected with a Mutex to avoid race conditions. It exposes classic queue operations `push` and `pop` (the latter with a timeout overloading) over this queue.

We tested it in `main.cpp` with two type of workers: Consumer that pop values from a FIFO of int, Producer that push values to it. We check that in the end, the FIFO is empty, that is to say all values pushed by producers were consumed by consumers.

5.4.3 Related files

- src/td4/d/main.cpp
- src/td4/d/Consumer.cpp
- src/td4/d/Producer.cpp

Chapter 6

6.

6.1 Summary

Implement the ActiveObject paradigm.

6.2 Contents

We define an ActiveObject class that inherits from Thread and stores pointers to Request objects in a FIFO. The Request class is an abstract one that owns a Semaphore.

To demonstrate this mechanism we define ActiveCalc that inherits from ActiveObject. CrunchReq inherits from Request. ActiveCalc `async_crunch` creates and adds a CrunchReq to the FIFO and returns a pointer to it. We need to allocate the memory for it on the heap since it is shared by the ActiveObject thread and the main thread. In order to separate the request logic from the payload computation itself, we define a Calculator object whose reference is passed at every CrunchRequest instance. In our example, Calculator returns the square of a double value. The CrunchRequest `execute` method override uses the Calculator object to perform the computation.

Finally we create a Client class that inherits from Thread and send multiple requests to the ActiveCalc object. It returns the results trough a method that waits for all the client's request to be done, and return their results in a vector. In the end we print all the clients results and check that all their requests were well treated.

6.3 Related files

- src/td6/main.cpp
- src/td6/ActiveCalc.cpp
- src/td6/ActiveObject.cpp
- src/td6/Calculator.cpp
- src/td6/CrunchReq.cpp
- src/td6/Request.cpp
- src/td6/TerminalReq.cpp
- src/td6/Client.cpp

Chapter 7

Hierarchical Index

7.1 Class Hierarchy

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

Chapter 8

Class Index

8.1 Class List

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

Chapter 9

File Index

9.1 File List

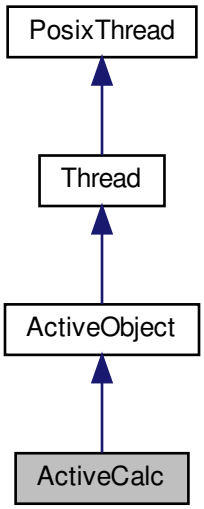
Here is a list of all documented files with brief descriptions:

Chapter 10

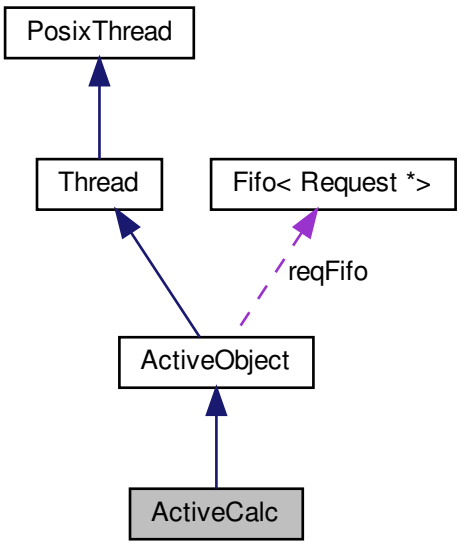
Class Documentation

10.1 ActiveCalc Class Reference

Inheritance diagram for ActiveCalc:



Collaboration diagram for ActiveCalc:



Public Member Functions

- **ActiveCalc** (Calculator &calc)
- CrunchReq * **async_crunch** (double param)
- TerminalReq * **stop** ()

Additional Inherited Members

10.1.1 Detailed Description

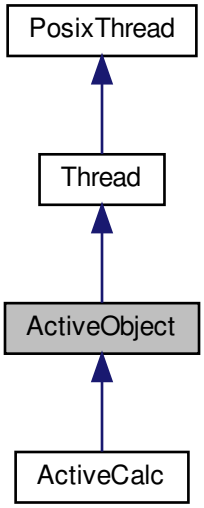
Definition at line 9 of file ActiveCalc.hpp.

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

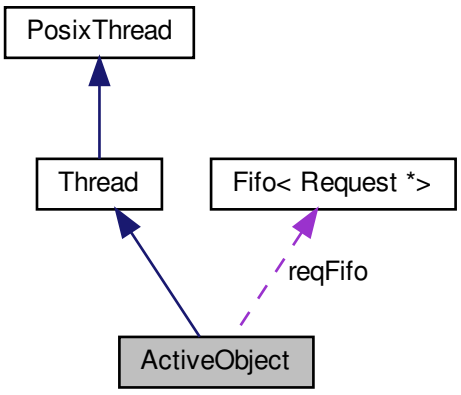
- src/td6/ActiveCalc.hpp
- src/td6/ActiveCalc.cpp

10.2 ActiveObject Class Reference

Inheritance diagram for ActiveObject:



Collaboration diagram for ActiveObject:



Protected Member Functions

- void **run** ()

Protected Attributes

- Fifo< Request * > **reqFifo**

Additional Inherited Members

10.2.1 Detailed Description

Definition at line 8 of file ActiveObject.hpp.

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

- includes/itr/ActiveObject.hpp
- src/td6/ActiveObject.cpp

10.3 Calculator Class Reference

Public Member Functions

- double **crunch** (double param)

10.3.1 Detailed Description

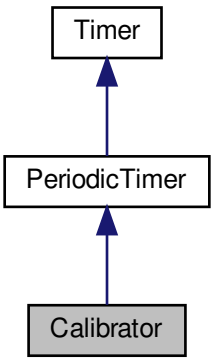
Definition at line 4 of file Calculator.hpp.

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

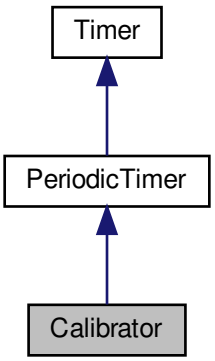
- src/td6/Calculator.hpp
- src/td6/Calculator.cpp

10.4 Calibrator Class Reference

Inheritance diagram for Calibrator:



Collaboration diagram for Calibrator:



Public Member Functions

- **Calibrator** (double samplingPeriod, unsigned int nSamples)
- double **nLoops** (double duration_ms)

Protected Member Functions

- void **callback** ()

Additional Inherited Members

10.4.1 Detailed Description

Definition at line 8 of file Calibrator.hpp.

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

- src/td3/c/Calibrator.hpp
- src/td3/c/Calibrator.cpp

10.5 Chrono Class Reference

Public Member Functions

- void **stop** ()
- void **restart** ()
- bool **isActive** ()
- double **startTime** ()
- double **stopTime** ()
- double **lap** ()

10.5.1 Detailed Description

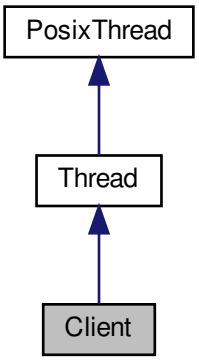
Definition at line 6 of file Chrono.hpp.

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

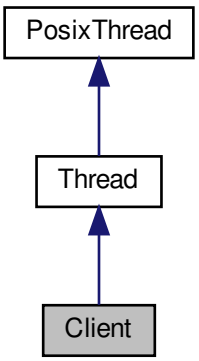
- includes/itr/Chrono.hpp
- src/td3/a/Chrono.cpp

10.6 Client Class Reference

Inheritance diagram for Client:



Collaboration diagram for Client:



Public Member Functions

- **Client** (unsigned int id, int minBound, int maxBound, ActiveCalc &calc)
- std::vector< int > **getResults** ()

Public Attributes

- unsigned int **id**

Protected Member Functions

- void **run** ()

Additional Inherited Members

10.6.1 Detailed Description

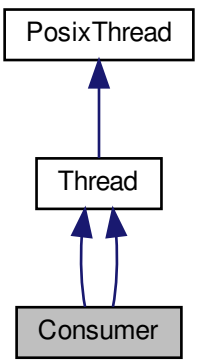
Definition at line 9 of file Client.hpp.

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

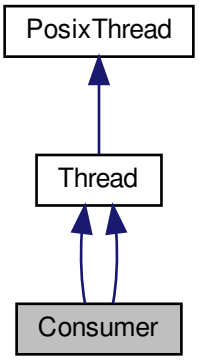
- src/td6/Client.hpp
- src/td6/Client.cpp

10.7 Consumer Class Reference

Inheritance diagram for Consumer:



Collaboration diagram for Consumer:



Public Member Functions

- **Consumer** (Semaphore &semaphore)
- **Consumer** (unsigned int id, int queries, Fifo< int > &fifo, Mutex &printMutex)

Protected Member Functions

- void **run** ()
- void **run** ()

Additional Inherited Members

10.7.1 Detailed Description

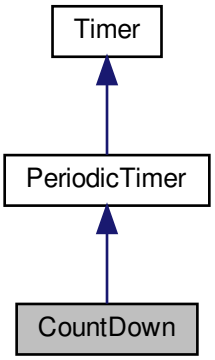
Definition at line 7 of file Consumer.hpp.

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

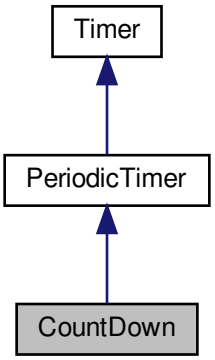
- src/td4/c/Consumer.hpp
- src/td4/c/Consumer.cpp

10.8 CountDown Class Reference

Inheritance diagram for CountDown:



Collaboration diagram for CountDown:



Public Member Functions

- **CountDown** (int n)
- void **callback** ()

Additional Inherited Members

10.8.1 Detailed Description

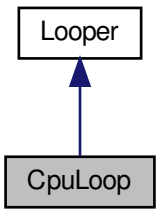
Definition at line 6 of file CountDown.hpp.

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

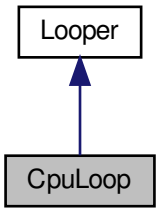
- src/td3/b/CountDown.hpp
- src/td3/b/CountDown.cpp

10.9 CpuLoop Class Reference

Inheritance diagram for CpuLoop:



Collaboration diagram for CpuLoop:



Public Member Functions

- **CpuLoop** (Calibrator &calibrator)
- void **runTime** (double duration_ms)

10.9.1 Detailed Description

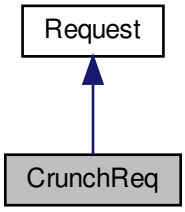
Definition at line 7 of file CpuLoop.hpp.

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

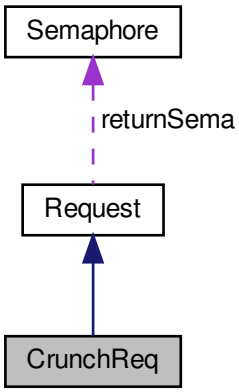
- src/td3/c/CpuLoop.hpp
- src/td3/c/CpuLoop.cpp

10.10 CrunchReq Class Reference

Inheritance diagram for CrunchReq:



Collaboration diagram for CrunchReq:



Public Member Functions

- **CrunchReq** (double param, Calculator &calc)
- void **execute** ()
- double **waitReturn** ()
- bool **shouldTerminate** ()

Additional Inherited Members

10.10.1 Detailed Description

Definition at line 7 of file CrunchReq.hpp.

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

- src/td6/CrunchReq.hpp
- src/td6/CrunchReq.cpp

10.11 Data Struct Reference

Public Attributes

- std::vector< Sample > **samples**
- unsigned int **nSamples**
- volatile bool **pStop**
- volatile unsigned int **iLoop**
- double **startTime**
- volatile unsigned int **nLoops**
- volatile double * **pCounter**
- bool **protec**
- pthread_mutex_t **mutex**

10.11.1 Detailed Description

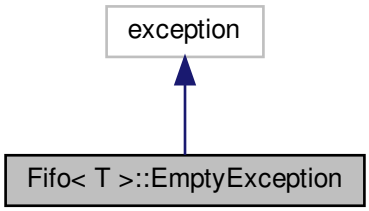
Definition at line 20 of file main.cpp.

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

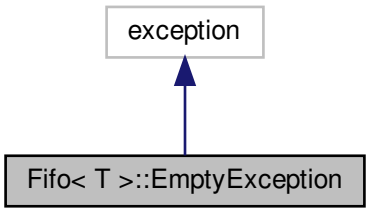
- src/td1/e/main.cpp

10.12 Fifo< T >::EmptyException Class Reference

Inheritance diagram for Fifo< T >::EmptyException:



Collaboration diagram for Fifo< T >::EmptyException:



10.12.1 Detailed Description

template<typename T>
class Fifo< T >::EmptyException

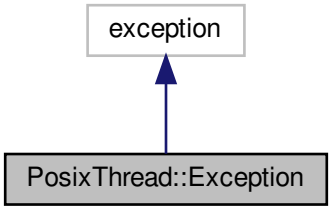
Definition at line 25 of file Fifo.hpp.

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

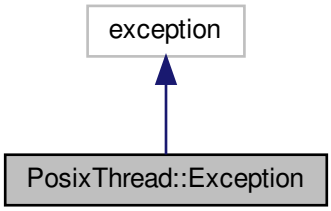
- includes/itr/Fifo.hpp

10.13 PosixThread::Exception Class Reference

Inheritance diagram for PosixThread::Exception:



Collaboration diagram for PosixThread::Exception:



10.13.1 Detailed Description

Definition at line 30 of file PosixThread.hpp.

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

- includes/itr/PosixThread.hpp

10.14 Fifo< T > Class Template Reference

Classes

- class EmptyException

Public Member Functions

- void **push** (T element)
- T **pop** ()
- T **pop** (double timeout_ms)

10.14.1 Detailed Description

template<typename T>
class Fifo< T >

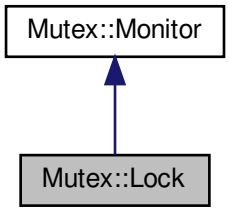
Definition at line 9 of file Fifo.hpp.

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

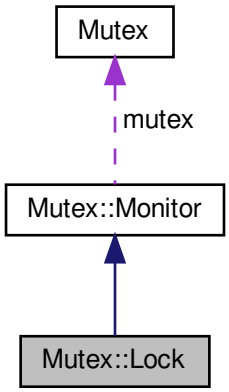
- includes/itr/Fifo.hpp

10.15 Mutex::Lock Class Reference

Inheritance diagram for Mutex::Lock:



Collaboration diagram for Mutex::Lock:



Public Member Functions

- **Lock** (Mutex &mutex)
- **Lock** (Mutex &mutex, double timeout_ms)

Additional Inherited Members

10.15.1 Detailed Description

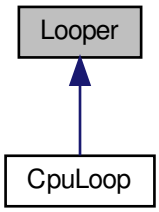
Definition at line 53 of file Mutex.hpp.

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

- includes/itr/Mutex.hpp
- src/td4/b/Mutex.cpp

10.16 Looper Class Reference

Inheritance diagram for Looper:



Public Member Functions

- double **runLoop** (double nLoops=DBL_MAX)
- double **getSample** ()
- double **stopLoop** ()

10.16.1 Detailed Description

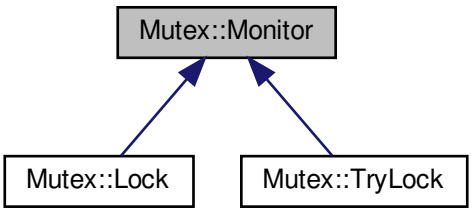
Definition at line 6 of file Looper.hpp.

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

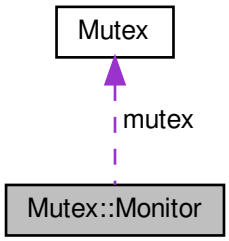
- src/td3/c/Looper.hpp
- src/td3/c/Looper.cpp

10.17 Mutex::Monitor Class Reference

Inheritance diagram for Mutex::Monitor:



Collaboration diagram for Mutex::Monitor:



Classes

- class TimeoutException

Public Member Functions

- void **wait** ()
- bool **wait** (double timeout_ms)
- void **notify** ()
- void **notifyAll** ()

Protected Member Functions

- **Monitor** (Mutex &mutex)

Protected Attributes

- Mutex & **mutex**

10.17.1 Detailed Description

Definition at line 27 of file Mutex.hpp.

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

- includes/itr/Mutex.hpp
- src/td4/b/Mutex.cpp

10.18 Mutex Class Reference

Classes

- class Lock
- class Monitor
- class TryLock

10.18.1 Detailed Description

Definition at line 7 of file Mutex.hpp.

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

- includes/itr/Mutex.hpp
- src/td4/b/Mutex.cpp

10.19 Parameters Struct Reference

Public Attributes

- double **a**
- double **b**

10.19.1 Detailed Description

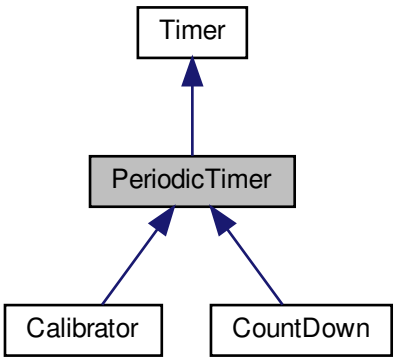
Definition at line 80 of file main.cpp.

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

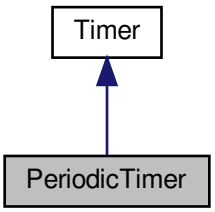
- src/td1/d/main.cpp

10.20 PeriodicTimer Class Reference

Inheritance diagram for PeriodicTimer:



Collaboration diagram for PeriodicTimer:



Public Member Functions

- void **start** (double duration_ms)

Additional Inherited Members

10.20.1 Detailed Description

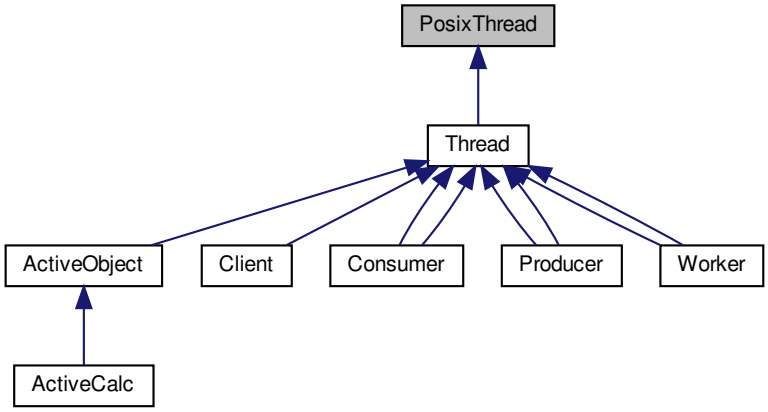
Definition at line 6 of file PeriodicTimer.hpp.

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

- includes/itr/PeriodicTimer.hpp
- src/td3/b/PeriodicTimer.cpp

10.21 PosixThread Class Reference

Inheritance diagram for PosixThread:



Classes

- class Exception

Public Member Functions

- **PosixThread** (pthread_t posixId)
- void **start** (void ∗(∗threadFunc)(void ∗), void ∗threadArg)
- void **join** ()
- bool **join** (double timeout_ms)
- bool **setScheduling** (int schedPolicy, int priority)
- bool **getScheduling** (int ∗p_schedPolicy, int ∗p_priority)

Protected Attributes

- bool **isActive**

10.21.1 Detailed Description

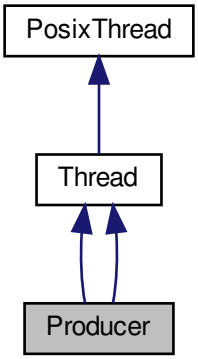
Definition at line 7 of file PosixThread.hpp.

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

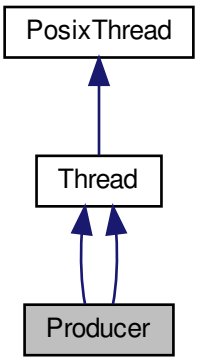
- includes/itr/PosixThread.hpp
- src/td4/a/PosixThread.cpp

10.22 Producer Class Reference

Inheritance diagram for Producer:



Collaboration diagram for Producer:



Public Member Functions

- **Producer** (Semaphore &semaphore)
- **Producer** (unsigned int id, int upperBound, Fifo< int > &fifo, Mutex &printMutex)

Protected Member Functions

- void **run** ()
- void **run** ()

Additional Inherited Members

10.22.1 Detailed Description

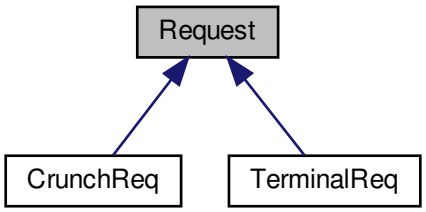
Definition at line 7 of file Producer.hpp.

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

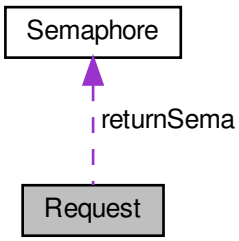
- src/td4/c/Producer.hpp
- src/td4/c/Producer.cpp

10.23 Request Class Reference

Inheritance diagram for Request:



Collaboration diagram for Request:



Public Member Functions

- virtual void **execute** ()=0
- virtual bool **shouldTerminate** ()=0
- void **waitReturn** ()

Protected Attributes

- Semaphore **returnSema**

10.23.1 Detailed Description

Definition at line 6 of file Request.hpp.

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

- includes/itr/Request.hpp
- src/td6/Request.cpp

10.24 Sample Struct Reference

Public Attributes

- int **seconds**
- unsigned int **loopsNumber**
- double **time_ms**

10.24.1 Detailed Description

Definition at line 31 of file main.cpp.

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

- src/td1/d/main.cpp

10.25 Semaphore Class Reference

Public Member Functions

- **Semaphore** (unsigned int initCount=0, unsigned int maxCount=UINT_MAX)
- void **give** ()
- void **take** ()
- bool **take** (double timeout_ms)

10.25.1 Detailed Description

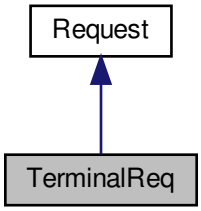
Definition at line 7 of file Semaphore.hpp.

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

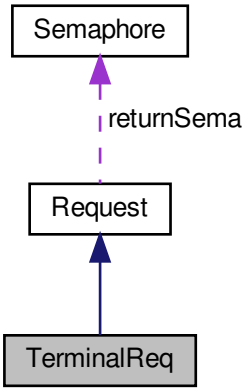
- includes/itr/Semaphore.hpp
- src/td4/c/Semaphore.cpp

10.26 TerminalReq Class Reference

Inheritance diagram for TerminalReq:



Collaboration diagram for TerminalReq:



Public Member Functions

- void **execute** ()
- bool **shouldTerminate** ()

Additional Inherited Members

10.26.1 Detailed Description

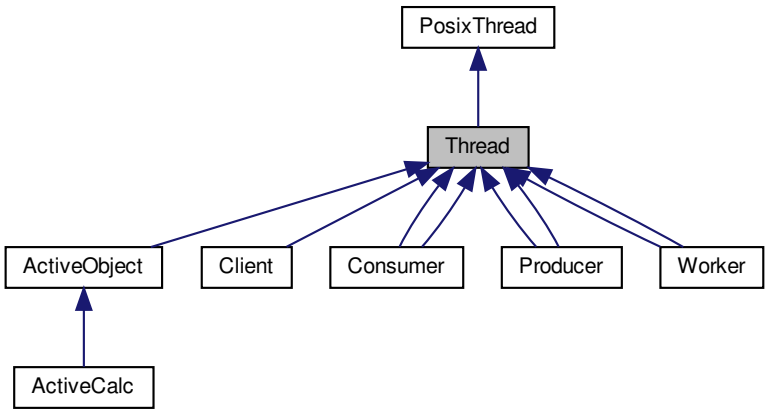
Definition at line 6 of file TerminalReq.hpp.

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

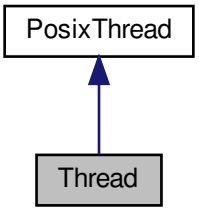
- src/td6/TerminalReq.hpp
- src/td6/TerminalReq.cpp

10.27 Thread Class Reference

Inheritance diagram for Thread:



Collaboration diagram for Thread:



Public Member Functions

- bool **start** ()
- void **sleep_ms** (double delay_ms)
- double **startTime_ms** ()
- double **stopTime_ms** ()
- double **execTime_ms** ()

Protected Member Functions

- virtual void **run** ()=0

Additional Inherited Members

10.27.1 Detailed Description

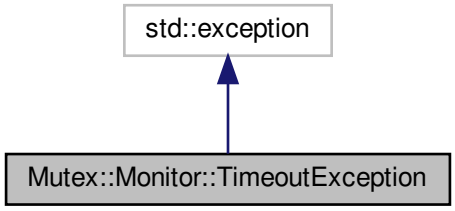
Definition at line 6 of file Thread.hpp.

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

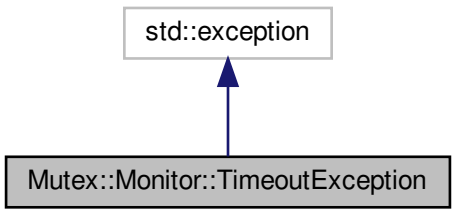
- includes/itr/Thread.hpp
- src/td4/a/Thread.cpp

10.28 Mutex::Monitor::TimeoutException Class Reference

Inheritance diagram for Mutex::Monitor::TimeoutException:



Collaboration diagram for Mutex::Monitor::TimeoutException:



10.28.1 Detailed Description

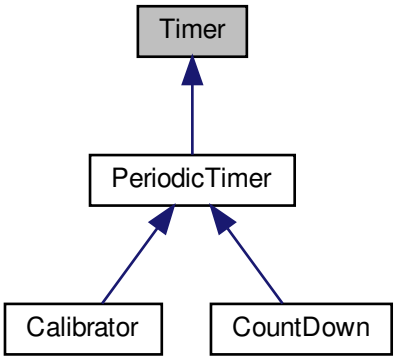
Definition at line 45 of file Mutex.hpp.

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

- includes/itr/Mutex.hpp

10.29 Timer Class Reference

Inheritance diagram for Timer:



Public Member Functions

- void **start** (double duration_ms)
- void **stop** ()

Protected Member Functions

- virtual void **callback** ()=0

Protected Attributes

- timer_t **tid**

10.29.1 Detailed Description

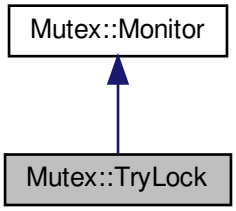
Definition at line 7 of file Timer.hpp.

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

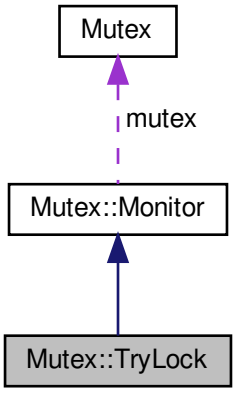
- includes/itr/Timer.hpp
- src/td3/b/Timer.cpp

10.30 Mutex::TryLock Class Reference

Inheritance diagram for Mutex::TryLock:



Collaboration diagram for Mutex::TryLock:



Public Member Functions

- **TryLock** (Mutex &mutex)

Additional Inherited Members

10.30.1 Detailed Description

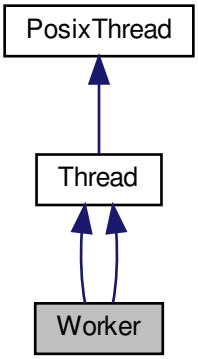
Definition at line 61 of file Mutex.hpp.

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

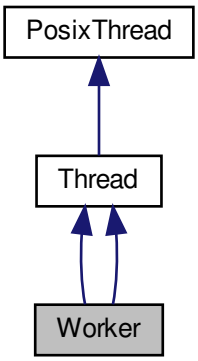
- includes/itr/Mutex.hpp
- src/td4/b/Mutex.cpp

10.31 Worker Class Reference

Inheritance diagram for Worker:



Collaboration diagram for Worker:



Public Member Functions

- **Worker** (unsigned int loops, volatile int ∗counter)
- **Worker** (unsigned int loops, volatile int ∗counter, Mutex &mutex)

Protected Member Functions

- void **run** ()
- void **run** ()

Additional Inherited Members

10.31.1 Detailed Description

Definition at line 6 of file Worker.hpp.

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

- src/td4/a/Worker.hpp
- src/td4/a/Worker.cpp

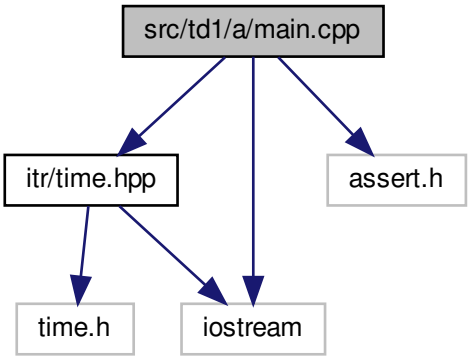
Chapter 11

File Documentation

11.1 src/td1/a/main.cpp File Reference

```
#include "itr/time.hpp"
#include <assert.h>
#include <iostream>
```

Include dependency graph for main.cpp:



Functions

- void **test_timespec_to_ms** ()
- void **test_timespec_from_ms** ()
- void **test_timespec_now** ()
- void **test_timespec_negate** ()
- void **test_timespec_add** ()
- void **test_timespec_substract** ()
- int **main** ()

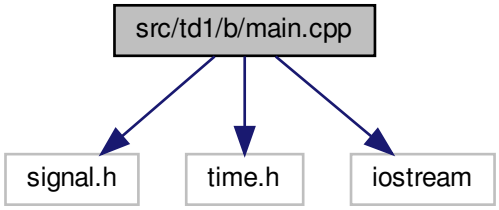
11.1.1 Detailed Description

- **TD:** 1.a
- **Example:** lib's time utils functions tests

11.2 src/td1/b/main.cpp File Reference

```
#include <signal.h>
#include <time.h>
#include <iostream>
```

Include dependency graph for main.cpp:



Functions

- void **handler** (int, siginfo_t *si, void *)
- int **main** ()

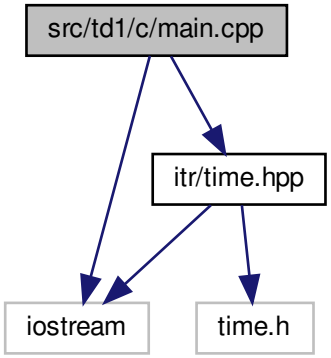
11.2.1 Detailed Description

- **TD:** 1.b
- **Example:** POSIX timer usage

11.3 src/td1/c/main.cpp File Reference

```
#include <iostream>
#include "itr/time.hpp"
```

Include dependency graph for main.cpp:



Functions

- void **incr** (unsigned int nLoops, double ∗pCounter)
- int **main** (int argc, char ∗argv[ ])

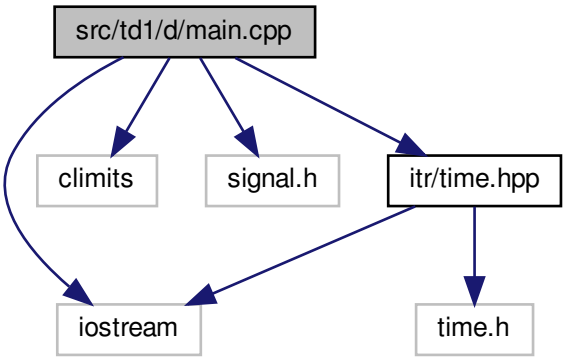
11.3.1 Detailed Description

- **TD:** 1.d
- **Example:** function simple execution time measurement

11.4 src/td1/d/main.cpp File Reference

```
#include <iostream>
#include <climits>
#include <signal.h>
#include "itr/time.hpp"
```

Include dependency graph for main.cpp:



Classes

- struct Sample
- struct Parameters

Functions

- void **handler** (int, siginfo_t *si, void *)
- unsigned int **incr** (unsigned int nLoops, double *pCounter, volatile bool *pStop)
- Sample **run** (int seconds)
- Parameters **calibrate** ()
- int **main** ()

11.4.1 Detailed Description

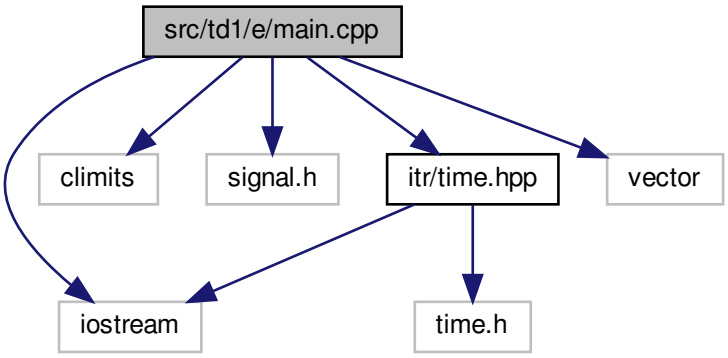
- **TD:** 1.d
- **Example:** function simple execution time calibration

11.5 src/td1/e/main.cpp File Reference

```
#include <iostream>
#include <climits>
#include <signal.h>
#include "itr/time.hpp"
```

```
#include <vector>
```

Include dependency graph for main.cpp:



Classes

- struct Sample
- struct Data
- struct Parameters

Functions

- void **handler** (int, siginfo_t *si, void *)
- unsigned int **incr** (unsigned int nLoops, double *pCounter, Data *data)
- std::vector< Sample > **run** (double samplingPeriod_ms, unsigned int samplesNumber)
- Parameters **calibrate** ()
- int **main** ()

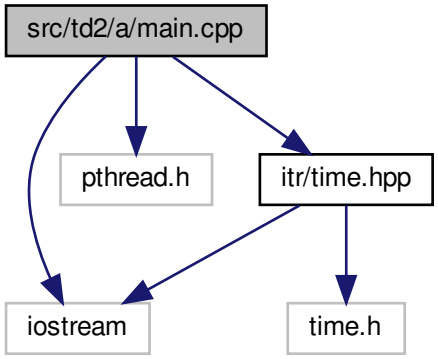
11.5.1 Detailed Description

- **TD:** 1.e
- **Example:** more accurate function execution time calibration

11.6 src/td2/a/main.cpp File Reference

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

```
#include "itr/time.hpp"
```
Include dependency graph for main.cpp:



Classes

- struct Data

Functions

- void **incr** (unsigned int nLoops, volatile double *pCounter)
- void * **call_incr** (void *v_data)
- int **main** (int argc, char *argv[])

11.6.1 Detailed Description

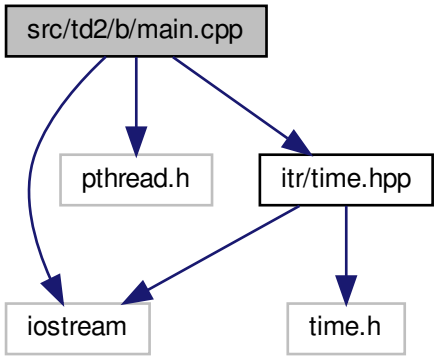
- **TD:** 2.a
- **Example:** POSIX thread usage

11.7 src/td2/b/main.cpp File Reference

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

```
#include "itr/time.hpp"
```

Include dependency graph for main.cpp:



Classes

- struct Data

Functions

- void **incr** (unsigned int nLoops, volatile double *pCounter)
- void * **call_incr** (void *v_data)
- int **main** (int argc, char *argv[])

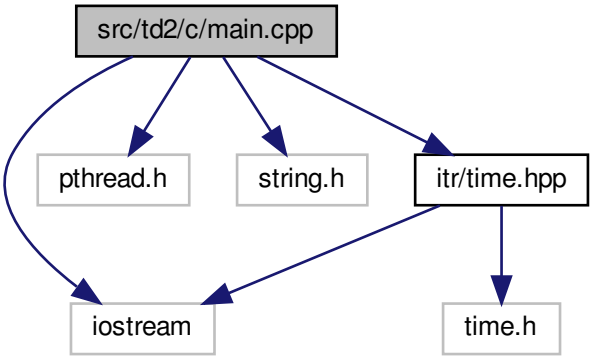
11.7.1 Detailed Description

- **TD:** 2.b
- **Example:** multi-theaded computation execution time measurement

11.8 src/td2/c/main.cpp File Reference

```
#include <iostream>
#include <pthread.h>
#include <string.h>
```

```
#include "itr/time.hpp"
```
Include dependency graph for main.cpp:



Classes

- struct Data

Functions

- void **incr** (unsigned int nLoops, volatile double *pCounter)
- void * **call_incr** (void *v_data)
- int **main** (int argc, char *argv[])

11.8.1 Detailed Description

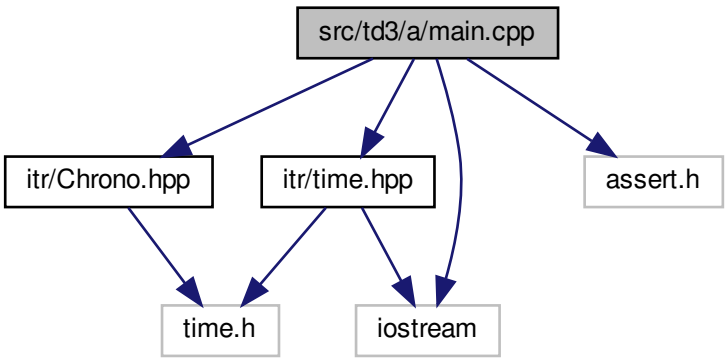
- **TD:** 2.c
- **Example:** POSIX mutex usage

11.9 src/td3/a/main.cpp File Reference

```
#include "itr/Chrono.hpp"
#include "itr/time.hpp"
#include <assert.h>
```

```
#include <iostream>
```

Include dependency graph for main.cpp:



Functions

- void **test_chrono_init** ()
- void **test_chrono_wait** ()
- int **main** ()

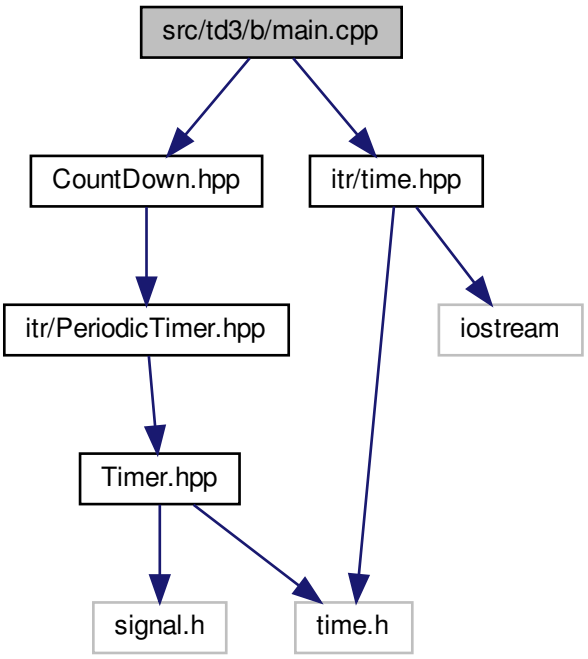
11.9.1 Detailed Description

- **TD:** 3.a
- **Example:** lib's Chrono tests

11.10 src/td3/b/main.cpp File Reference

```
#include "CountDown.hpp"
#include "itr/time.hpp"
```

Include dependency graph for main.cpp:



Functions

- int **main** ()

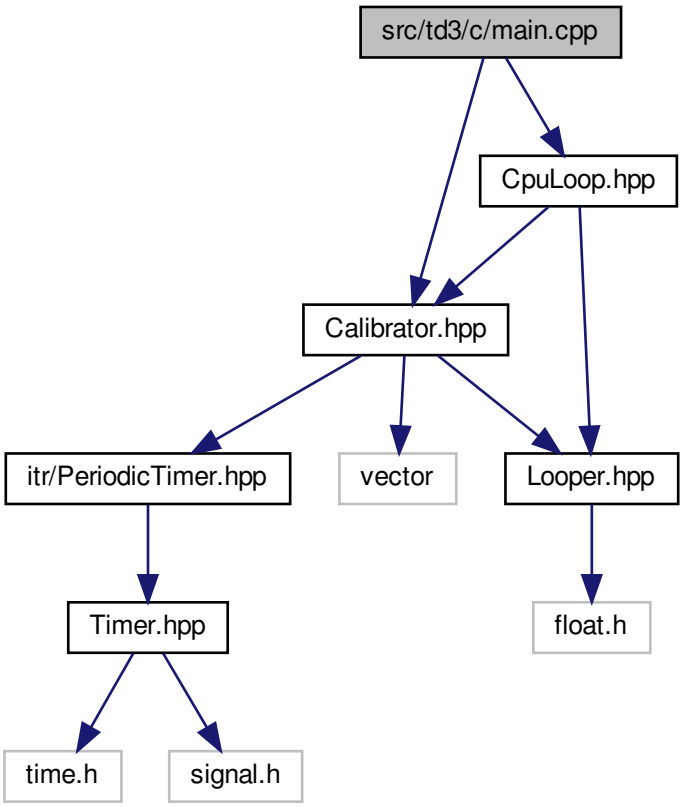
11.10.1 Detailed Description

- **TD:** 3.b
- **Example:** lib's PeriodicTimer usage (through CountDown).

11.11 src/td3/c/main.cpp File Reference

```
#include "CpuLoop.hpp"
#include "Calibrator.hpp"
```

Include dependency graph for main.cpp:



Functions

- int **main** ()

11.11.1 Detailed Description

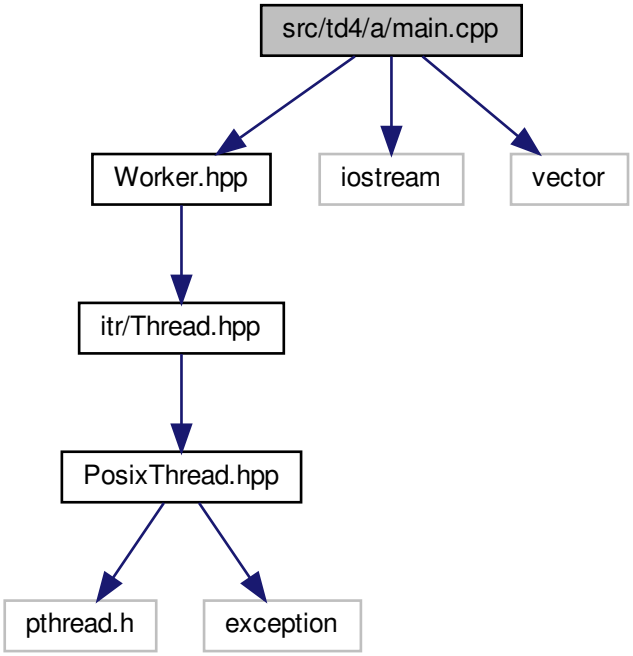
- **TD:** 3.c
- **Example:** function execution time calibration using lib's Timer

11.12 src/td4/a/main.cpp File Reference

```
#include "Worker.hpp"
#include <iostream>
```

```
#include <vector>
```

Include dependency graph for main.cpp:



Functions

- int **main** ()

11.12.1 Detailed Description

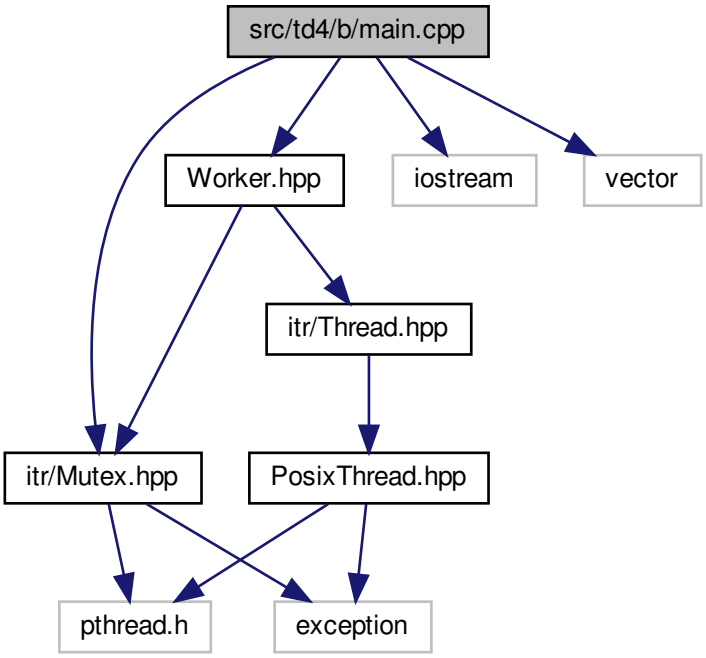
- **TD:** 4.a
- **Example:** lib's Thread usage (through Worker)

11.13 src/td4/b/main.cpp File Reference

```
#include "itr/Mutex.hpp"
#include "Worker.hpp"
#include <iostream>
```

```
#include <vector>
```

Include dependency graph for main.cpp:



Functions

- int **main** ()

11.13.1 Detailed Description

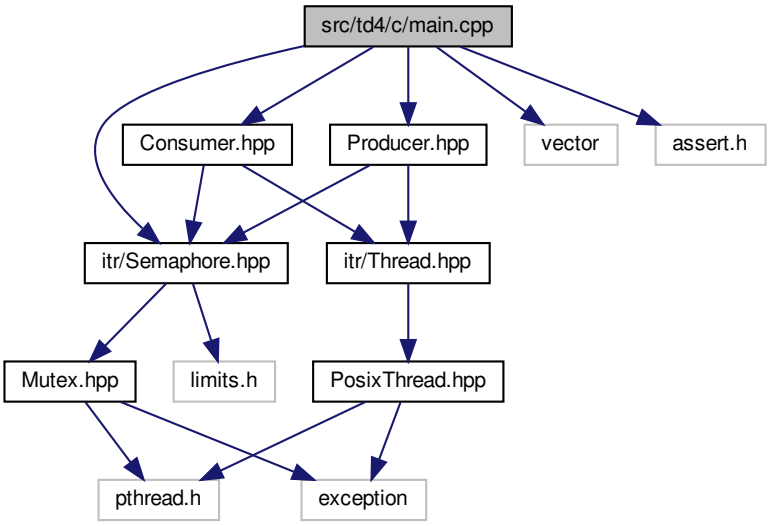
- **TD:** 4.b
- **Example:** lib's Mutex usage (through Worker)

11.14 src/td4/c/main.cpp File Reference

```
#include "itr/Semaphore.hpp"
#include "Consumer.hpp"
#include "Producer.hpp"
#include <vector>
```

```
#include <assert.h>
```

Include dependency graph for main.cpp:



Functions

- int **main** ()

11.14.1 Detailed Description

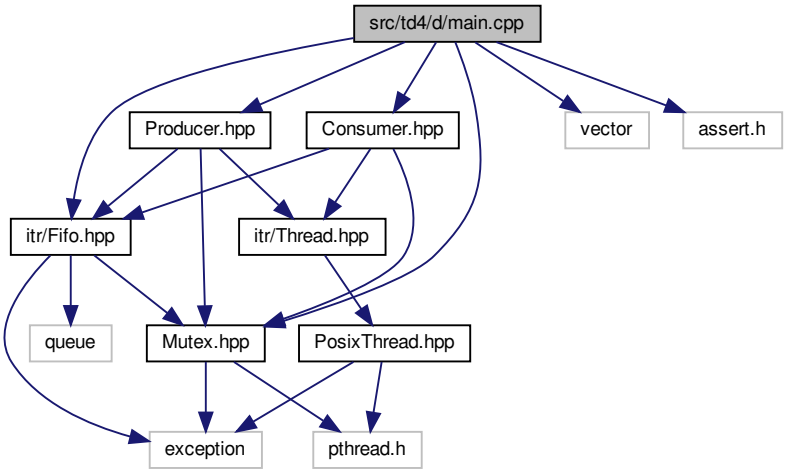
- **TD:** 4.c
- **Example:** lib's Semaphore usage (through Consumer and Producer)

11.15 src/td4/d/main.cpp File Reference

```
#include "itr/Fifo.hpp"
#include "itr/Mutex.hpp"
#include "Producer.hpp"
#include "Consumer.hpp"
#include <vector>
```

```
#include <assert.h>
```

Include dependency graph for main.cpp:



Functions

- int **main** ()

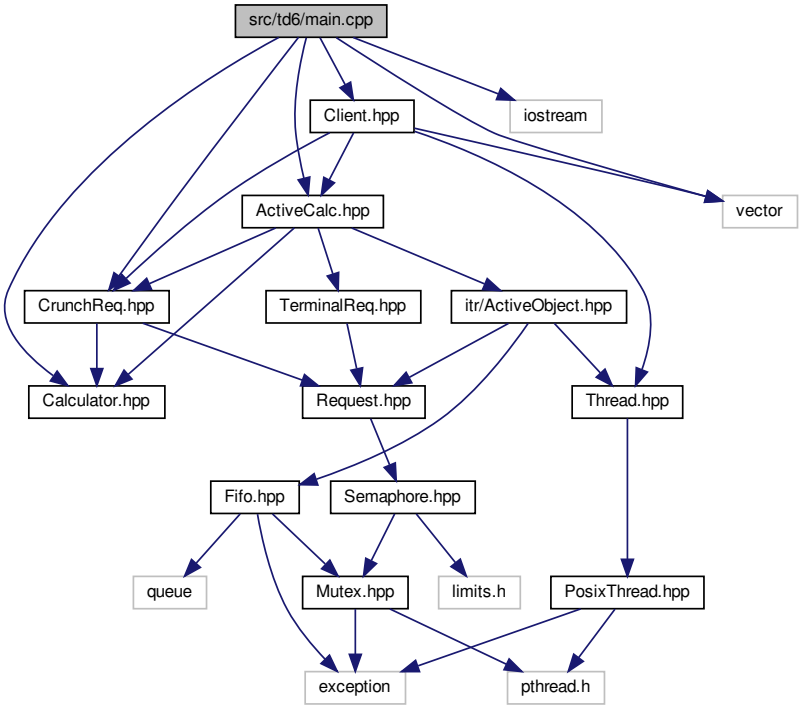
11.15.1 Detailed Description

- **TD:** 4.d
- **Example:** lib's Fifo usage (through Consumer and Producer)

11.16 src/td6/main.cpp File Reference

```
#include "Calculator.hpp"
#include "ActiveCalc.hpp"
#include "CrunchReq.hpp"
#include "Client.hpp"
#include <iostream>
#include <vector>
```

Include dependency graph for main.cpp:



Functions

- int **main** ()

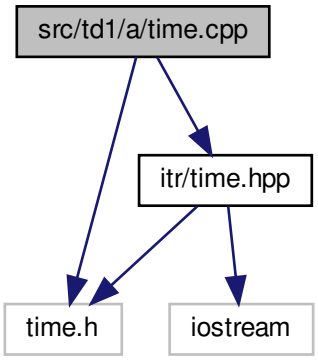
11.16.1 Detailed Description

- **TD:** 6.
- **Example:** lib's ActiveObject usage (through ActiveCalc, Calculator, CrunchReq, TerminalReq and Client)

11.17 src/td1/a/time.cpp File Reference

```
#include "itr/time.hpp"
#include <time.h>
```

Include dependency graph for time.cpp:



Functions

- double timespec_to_ms (const timespec &time_ts)

  This is a test.
- timespec **timespec_from_ms** (double time_ms)
- timespec **timespec_now** ()
- timespec **timespec_negate** (const timespec &time_ts)
- timespec **timespec_add** (const timespec &time1_ts, const timespec &time2_ts)
- timespec **timespec_substract** (const timespec &time1_ts, const timespec &time2_ts)
- timespec **timespec_wait** (const timespec &delay_ts)
- timespec **operator-** (const timespec &time_ts)
- timespec **operator+** (const timespec &time1_ts, const timespec &time2_ts)
- timespec **operator-** (const timespec &time1_ts, const timespec &time2_ts)
- timespec & **operator+=** (timespec &time_ts, const timespec &delay_ts)
- timespec & **operator-=** (timespec &time_ts, const timespec &delay_ts)
- bool **operator==** (const timespec &time1_ts, const timespec &time2_ts)
- bool **operator!=** (const timespec &time1_ts, const timespec &time2_ts)
- bool **operator**< (const timespec &time1_ts, const timespec &time2_ts)
- bool **operator**> (const timespec &time1_ts, const timespec &time2_ts)
- std::ostream & **operator**<< (std::ostream &os, const timespec &time_ts)

11.17.1 Detailed Description

- **TD:** 1.a
- **Lib implementation:** time utils functions

11.17.2 Function Documentation

11.17.2.1 timespec_to_ms()

```
double timespec_to_ms (
            const timespec & time_ts )
```

This is a test.

Parameters

time_ts	is a timespec struct

Returns

it returns the absolute time in milliseconds as a double

Definition at line 11 of file time.cpp.

11.18 src/td1/doc.h File Reference

11.19 src/td2/doc.h File Reference

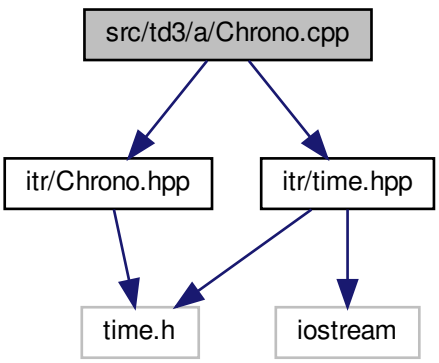
11.20 src/td3/doc.h File Reference

11.21 src/td4/doc.h File Reference

11.22 src/td3/a/Chrono.cpp File Reference

```
#include "itr/Chrono.hpp"
#include "itr/time.hpp"
```

Include dependency graph for Chrono.cpp:



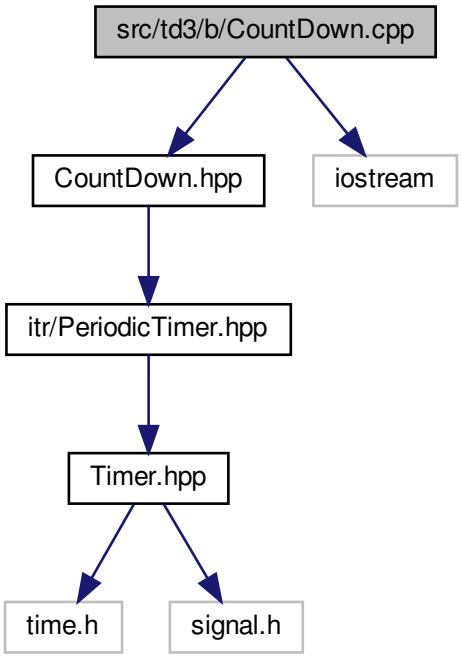
11.22.1 Detailed Description

- **TD:** 3.a
- **Lib implementation:** Chrono

11.23 src/td3/b/CountDown.cpp File Reference

```
#include "CountDown.hpp"
#include <iostream>
```

Include dependency graph for CountDown.cpp:



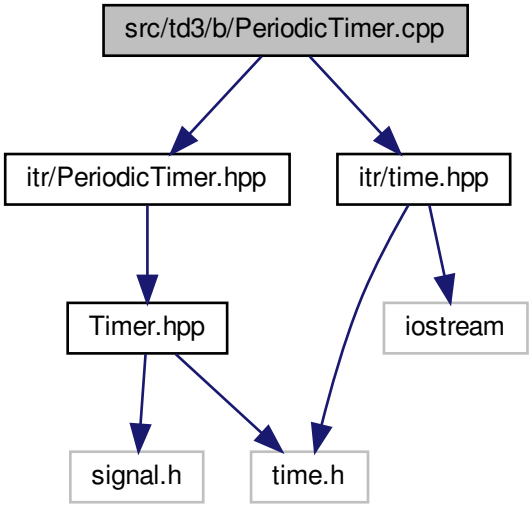
11.23.1 Detailed Description

- **TD:** 3.b
- **Example class:** CountDown

11.24 src/td3/b/PeriodicTimer.cpp File Reference

```
#include "itr/PeriodicTimer.hpp"
#include "itr/time.hpp"
```

Include dependency graph for PeriodicTimer.cpp:



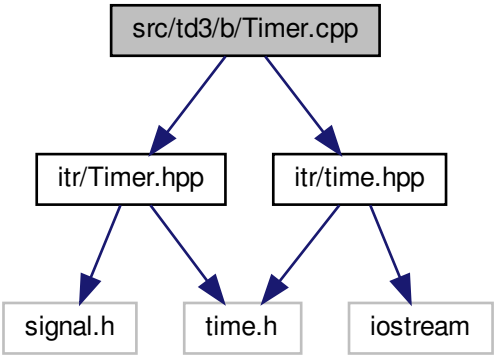
11.24.1 Detailed Description

- **TD:** 3.b
- **Lib implementation:** PeriodicTimer

11.25 src/td3/b/Timer.cpp File Reference

```
#include "itr/Timer.hpp"
#include "itr/time.hpp"
```

Include dependency graph for Timer.cpp:



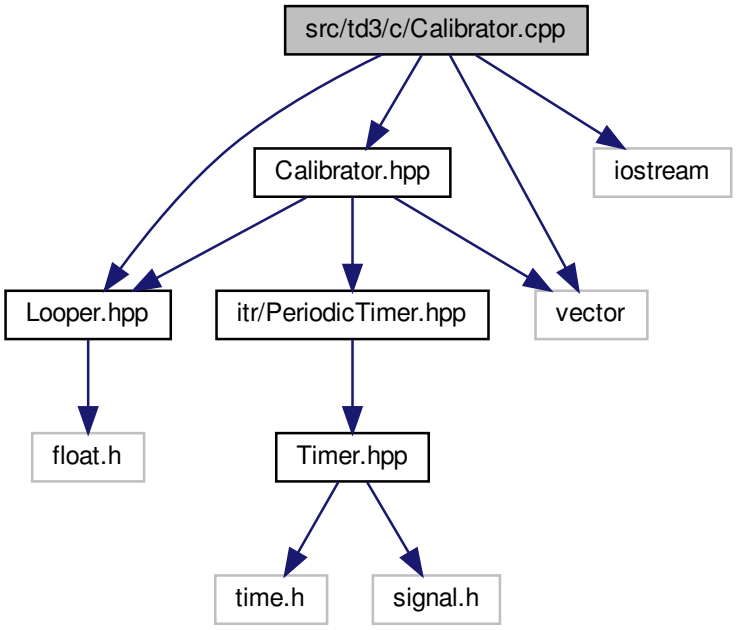
11.25.1 Detailed Description

- **TD:** 3.b
- **Lib implementation:** Timer

11.26 src/td3/c/Calibrator.cpp File Reference

```
#include "Calibrator.hpp"
#include "Looper.hpp"
#include <vector>
#include <iostream>
```

Include dependency graph for Calibrator.cpp:



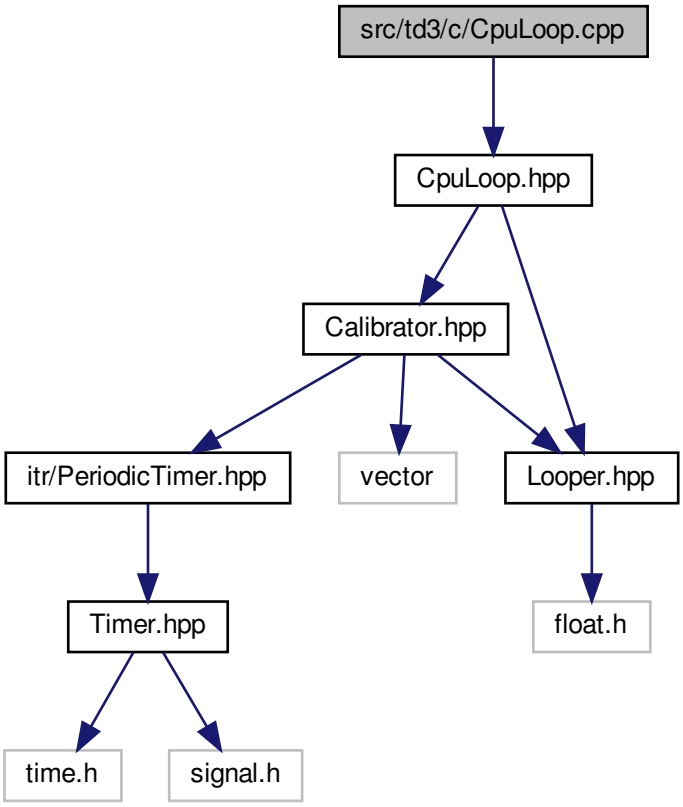
11.26.1 Detailed Description

- **TD:** 3.c
- **Example class:** Calibrator

11.27 src/td3/c/CpuLoop.cpp File Reference

```
#include "CpuLoop.hpp"
```

Include dependency graph for CpuLoop.cpp:



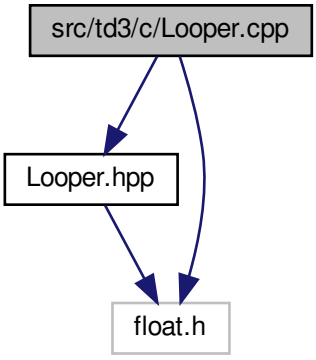
11.27.1 Detailed Description

- **TD:** 3.c
- **Example class:** CpuLoop

11.28 src/td3/c/Looper.cpp File Reference

```
#include "Looper.hpp"
#include <float.h>
```

Include dependency graph for Looper.cpp:



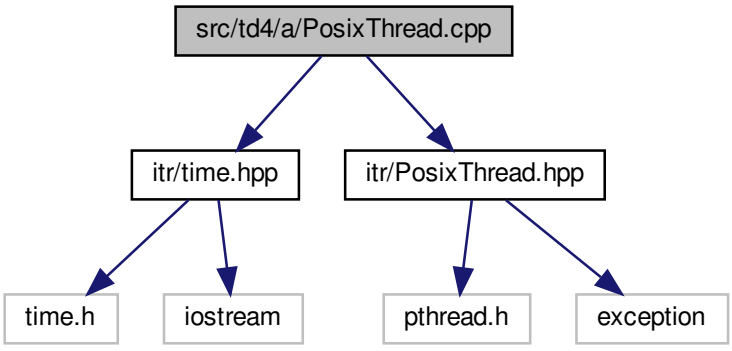
11.28.1 Detailed Description

- **TD:** 3.c
- **Example class:** Looper

11.29 src/td4/a/PosixThread.cpp File Reference

```
#include "itr/time.hpp"
#include "itr/PosixThread.hpp"
```

Include dependency graph for PosixThread.cpp:



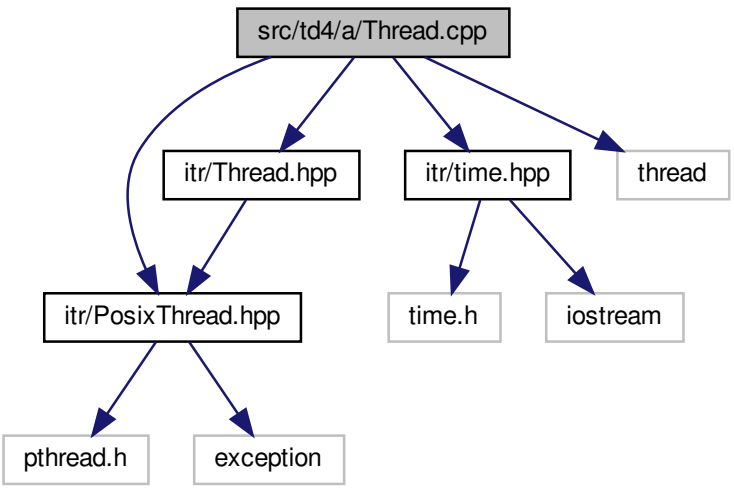
11.29.1 Detailed Description

- **TD:** 4.a
- **Lib implementation:** PosixThread

11.30 src/td4/a/Thread.cpp File Reference

```
#include "itr/PosixThread.hpp"
#include "itr/Thread.hpp"
#include "itr/time.hpp"
#include <thread>
```

Include dependency graph for Thread.cpp:



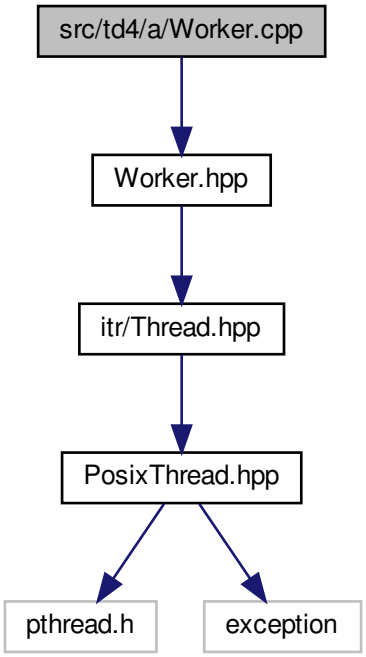
11.30.1 Detailed Description

- **TD:** 4.a
- **Lib implementation:** Thread

11.31 src/td4/a/Worker.cpp File Reference

```
#include "Worker.hpp"
```

Include dependency graph for Worker.cpp:



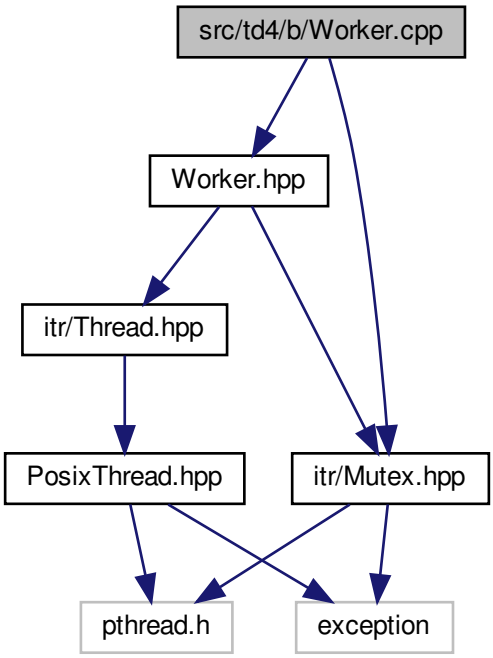
11.31.1 Detailed Description

- **TD:** 4.a
- **Example class:** Worker

11.32 src/td4/b/Worker.cpp File Reference

```
#include "Worker.hpp"
#include "itr/Mutex.hpp"
```

Include dependency graph for Worker.cpp:



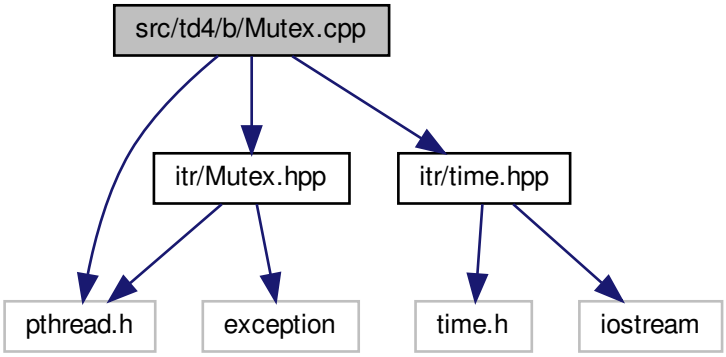
11.32.1 Detailed Description

- **TD:** 4.b
- **Example class:** Worker

11.33 src/td4/b/Mutex.cpp File Reference

```
#include "itr/Mutex.hpp"
#include <pthread.h>
#include "itr/time.hpp"
```

Include dependency graph for Mutex.cpp:



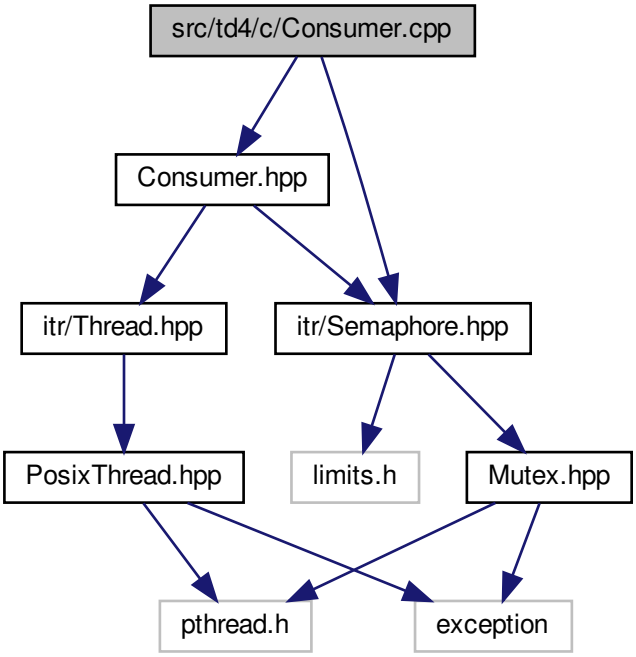
11.33.1 Detailed Description

- **TD:** 4.b
- **Lib implementation:** Mutex

11.34 src/td4/c/Consumer.cpp File Reference

```
#include "Consumer.hpp"
#include "itr/Semaphore.hpp"
```

Include dependency graph for Consumer.cpp:



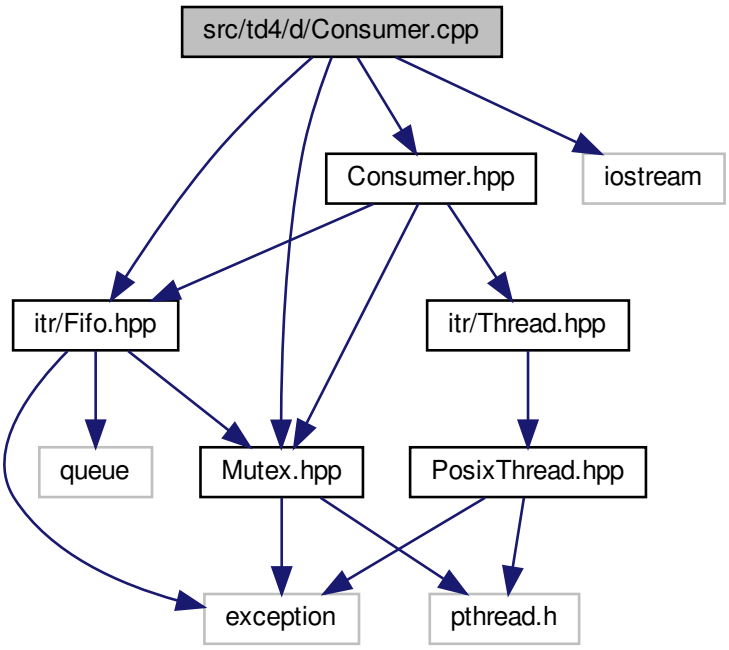
11.34.1 Detailed Description

- **TD:** 4.c
- **Example class:** Consumer

11.35 src/td4/d/Consumer.cpp File Reference

```
#include "itr/Fifo.hpp"
#include "itr/Mutex.hpp"
#include "Consumer.hpp"
#include <iostream>
```

Include dependency graph for Consumer.cpp:



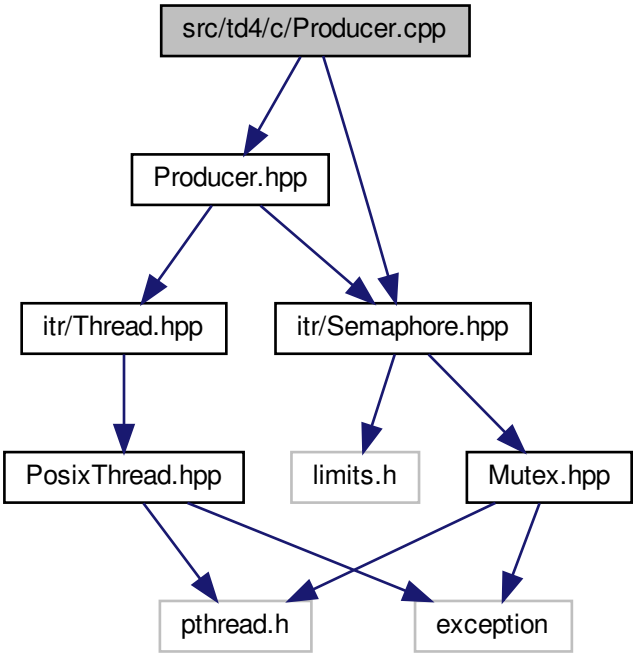
11.35.1 Detailed Description

- **TD:** 4.d
- **Example class:** Consumer

11.36 src/td4/c/Producer.cpp File Reference

```
#include "Producer.hpp"
#include "itr/Semaphore.hpp"
```

Include dependency graph for Producer.cpp:



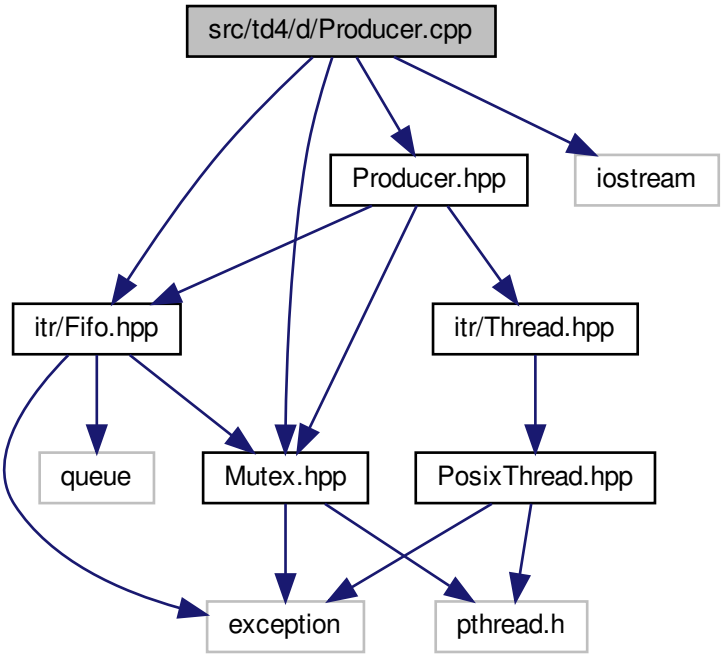
11.36.1 Detailed Description

- **TD:** 4.c
- **Example class:** Producer

11.37 src/td4/d/Producer.cpp File Reference

```
#include "itr/Fifo.hpp"
#include "itr/Mutex.hpp"
#include "Producer.hpp"
#include <iostream>
```

Include dependency graph for Producer.cpp:



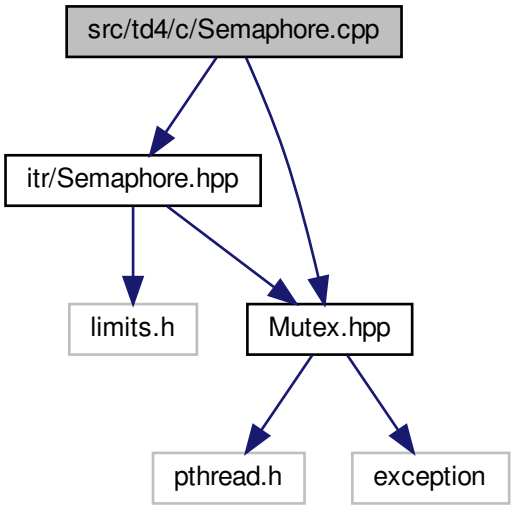
11.37.1 Detailed Description

- **TD:** 4.d
- **Example class:** Producer

11.38 src/td4/c/Semaphore.cpp File Reference

```
#include "itr/Semaphore.hpp"
#include "itr/Mutex.hpp"
```

Include dependency graph for Semaphore.cpp:



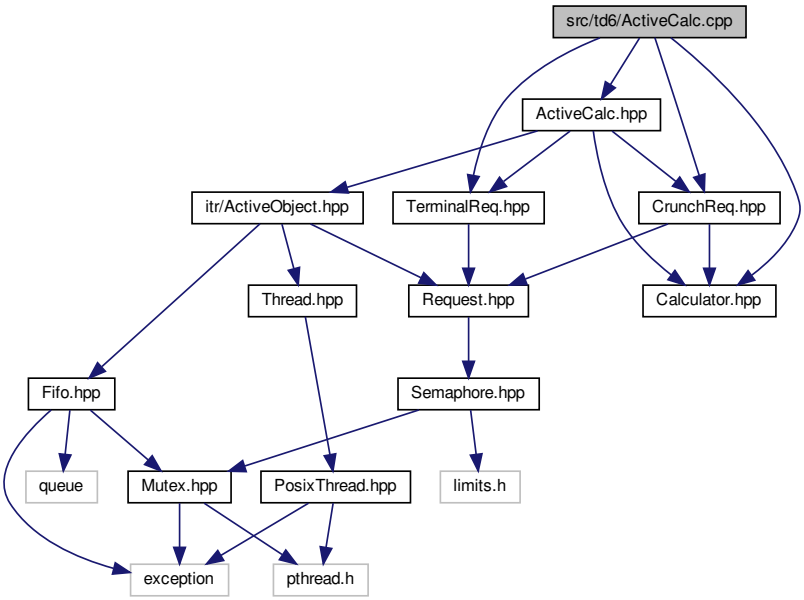
11.38.1 Detailed Description

- **TD:** 4.c
- **Lib implementation:** Semaphore

11.39 src/td6/ActiveCalc.cpp File Reference

```
#include "ActiveCalc.hpp"
#include "CrunchReq.hpp"
#include "TerminalReq.hpp"
#include "Calculator.hpp"
```

Include dependency graph for ActiveCalc.cpp:



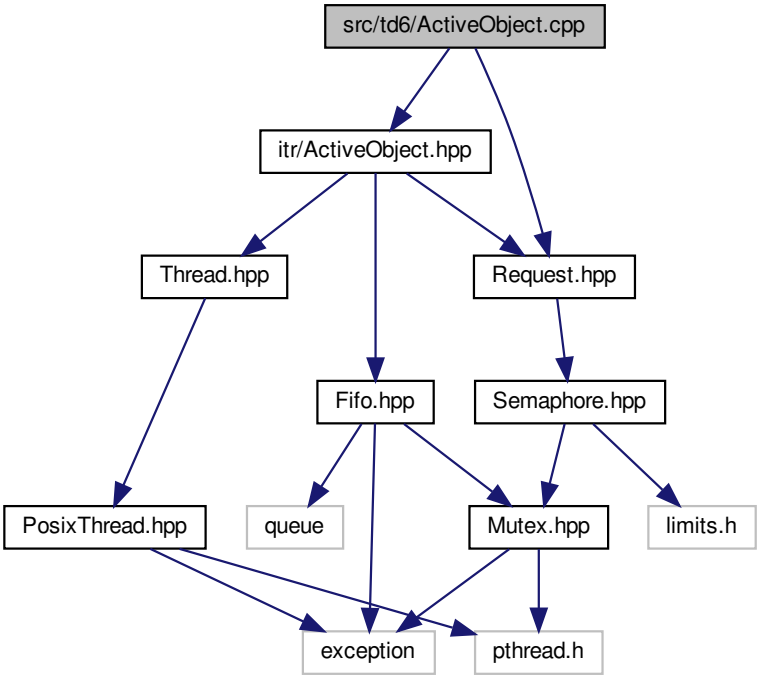
11.39.1 Detailed Description

- **TD:** 6.
- **Example class:** ActiveCalc

11.40 src/td6/ActiveObject.cpp File Reference

```
#include "itr/ActiveObject.hpp"
#include "itr/Request.hpp"
```

Include dependency graph for ActiveObject.cpp:



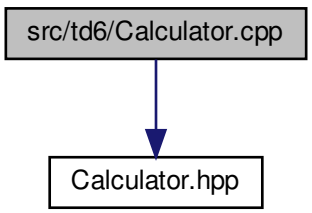
11.40.1 Detailed Description

- **TD:** 6.
- **Lib implementation:** ActiveObject

11.41 src/td6/Calculator.cpp File Reference

```
#include "Calculator.hpp"
```

Include dependency graph for Calculator.cpp:



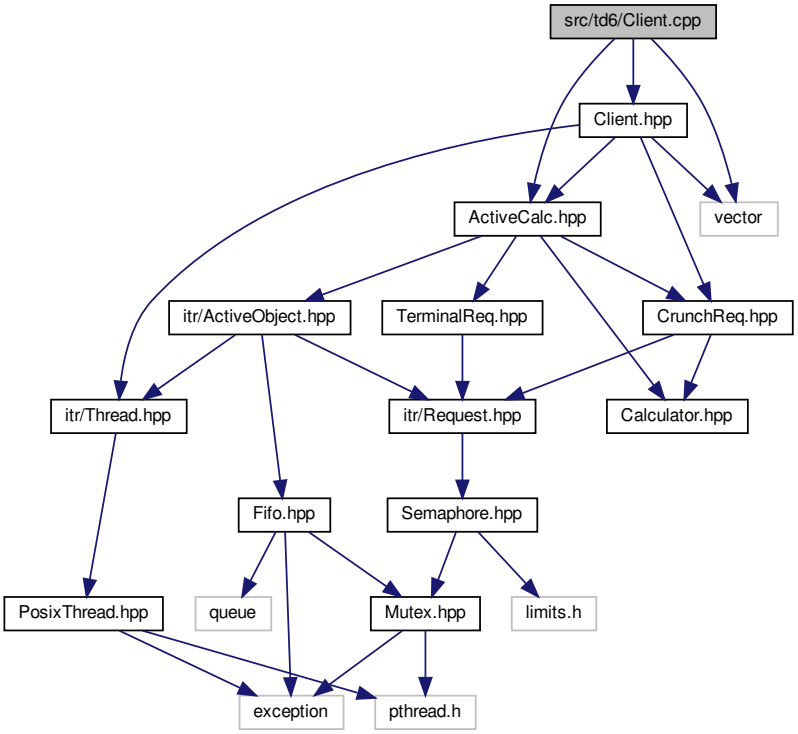
11.41.1 Detailed Description

- **TD:** 6.
- **Example class:** Calculator

11.42 src/td6/Client.cpp File Reference

```
#include "Client.hpp"
#include "ActiveCalc.hpp"
#include <vector>
```

Include dependency graph for Client.cpp:



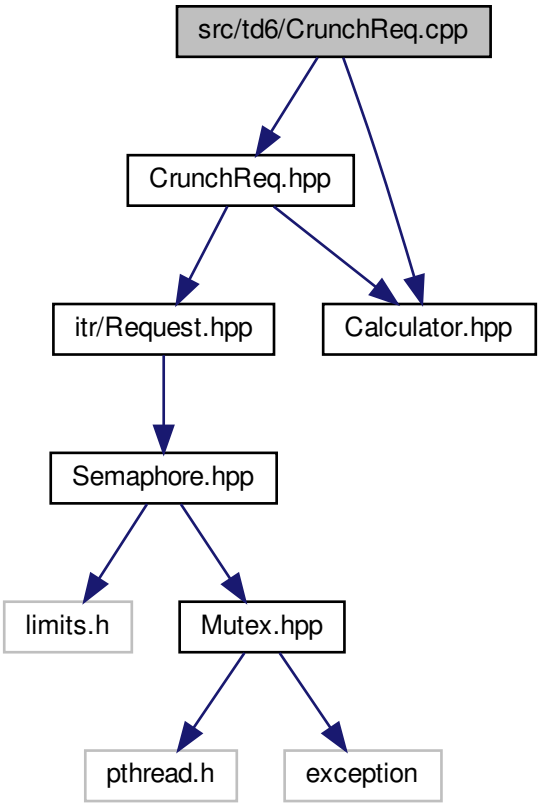
11.42.1 Detailed Description

- **TD:** 6.
- **Example class:** Client

11.43 src/td6/CrunchReq.cpp File Reference

```
#include "CrunchReq.hpp"
#include "Calculator.hpp"
```

Include dependency graph for CrunchReq.cpp:



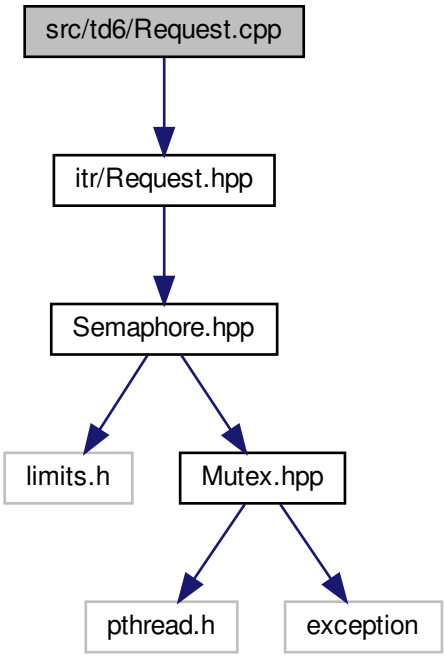
11.43.1 Detailed Description

- **TD:** 6.
- **Example class:** CrunchReq

11.44 src/td6/Request.cpp File Reference

```
#include "itr/Request.hpp"
```

Include dependency graph for Request.cpp:



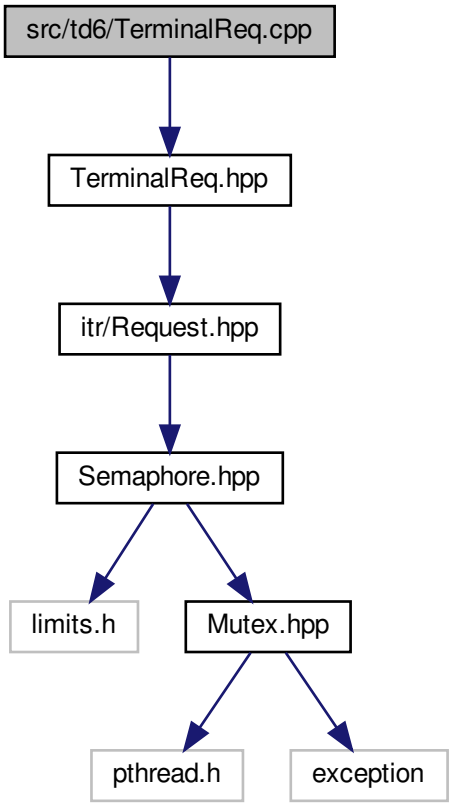
11.44.1 Detailed Description

- **TD:** 6.
- **Lib implementation:** Request

11.45 src/td6/TerminalReq.cpp File Reference

```
#include "TerminalReq.hpp"
```

Include dependency graph for TerminalReq.cpp:



11.45.1 Detailed Description

- **TD:** 6.
- **Example class:** TerminalReq

Index