Process Management

Generated on Mon Dec 23 2024 16:21:18 for Process Management by Doxygen 1.12.0

Mon Dec 23 2024 16:21:18

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

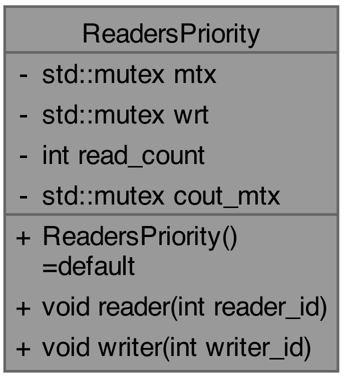
Chapter 3

Class Documentation

3.1 ReadersPriority Class Reference

Implements the Readers Priority solution to the Readers-Writers problem.

Collaboration diagram for ReadersPriority:


ReadersPriority
- std::mutex mtx
- std::mutex wrt
- int read_count
- std::mutex cout_mtx
+ ReadersPriority() =default
+ void reader(int reader_id)
+ void writer(int writer_id)

Public Member Functions

- ReadersPriority ()=default

  Default constructor.
- void reader (int reader_id)

  Function executed by reader threads.
- void writer (int writer_id)

  Function executed by writer threads.

Private Attributes

- std::mutex mtx

  Mutex to protect the shared counter read_count.
- std::mutex wrt

  Mutex used by writers. If locked, writers are writing (or waiting to write).
- int read_count = 0

  Number of active readers.
- std::mutex cout_mtx

  Mutex to protect output operations to std::cout.

3.1.1 Detailed Description

Implements the Readers Priority solution to the Readers-Writers problem.

In this Readers Priority approach:

- Multiple readers can read concurrently if no writer is writing.
- A writer can write only if no reader is reading and no other writer is writing.
- Readers have priority: once a reader starts reading, it prevents writers from acquiring the lock until all readers have finished.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 ReadersPriority()

ReadersPriority::ReadersPriority () [default]

Default constructor.

3.1.3 Member Function Documentation

3.1.3.1 reader()

```
void ReadersPriority::reader (
                    int reader_id)   [inline]
```

Function executed by reader threads.

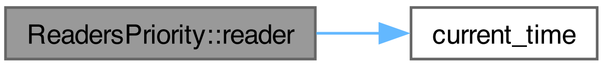
Each reader enters a loop:

- Acquires a lock on mtx to safely increment read_count.
- If this thread is the first reader (read_count == 1), it locks wrt to block writers.
- Simulates a read operation and logs it to std::cout.
- Decrements read_count, and if it is the last reader (read_count == 0), unlocks wrt.
- Sleeps briefly before attempting to read again.

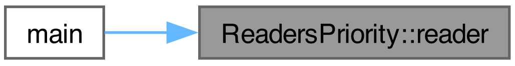
Parameters

reader_id	An integer identifier for the reader (for logging).

Here is the call graph for this function:



Here is the caller graph for this function:



3.1.3.2 writer()

void ReadersPriority::writer (
int writer_id) [inline]

Function executed by writer threads.

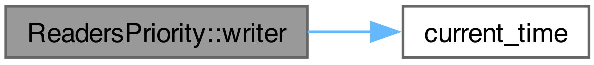
Each writer enters a loop:

- Locks wrt to gain exclusive writing access.
- Simulates a write operation and logs it to std::cout.
- Unlocks wrt to allow other readers or writers.
- Sleeps briefly before attempting to write again.

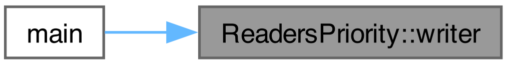
Parameters

writer_id	An integer identifier for the writer (for logging).

Here is the call graph for this function:



Here is the caller graph for this function:



3.1.4 Member Data Documentation

3.1.4.1 cout_mtx

std::mutex ReadersPriority::cout_mtx [private]

Mutex to protect output operations to std::cout.

3.1.4.2 mtx

std::mutex ReadersPriority::mtx [private]

Mutex to protect the shared counter read_count.

3.1.4.3 read_count

int ReadersPriority::read_count = 0 [private]

Number of active readers.

3.1.4.4 wrt

std::mutex ReadersPriority::wrt [private]

Mutex used by writers. If locked, writers are writing (or waiting to write).

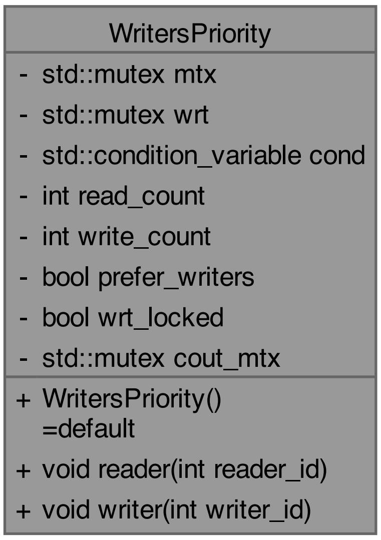
The documentation for this class was generated from the following file:

- src/ReadersPriority.cpp

3.2 WritersPriority Class Reference

Implements a Writers Priority solution to the Readers-Writers problem.

Collaboration diagram for WritersPriority:



Public Member Functions

- WritersPriority ()=default

  Default constructor.
- void reader (int reader_id)

  Function executed by reader threads.
- void writer (int writer_id)

  Function executed by writer threads.

Private Attributes

- std::mutex mtx

  Mutex to protect shared state (e.g., counters, flags).
- std::mutex wrt

  Mutex used by writers. If locked, a writer is writing (or waiting to write).
- std::condition_variable cond

  Condition variable to signal state changes to waiting threads.
- int read_count = 0

  Current number of active readers.
- int write_count = 0

  Current number of writers either waiting or writing.
- bool prefer_writers = true

  Flag to indicate if the system currently prefers writers over readers.
- bool wrt_locked = false

  Indicates whether the wrt mutex is currently locked by a writer.
- std::mutex cout_mtx

  Mutex to protect output operations (e.g. writing to std::cout).

3.2.1 Detailed Description

Implements a Writers Priority solution to the Readers-Writers problem.

In this Writers Priority approach:

- Writers have priority to acquire the lock over new readers.
- A writer will block incoming readers when it is waiting for or holding the write lock.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 WritersPriority()

WritersPriority::WritersPriority () [default]

Default constructor.

3.2.3 Member Function Documentation

3.2.3.1 reader()

```
void WritersPriority::reader (
                int reader_id)  [inline]
```

Function executed by reader threads.

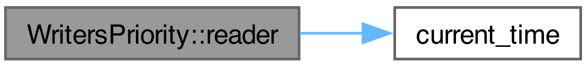
Each reader enters a loop:

- Waits while any writer is waiting.
- Increments the reader count, locking wrt if this is the first reader.
- Performs reading operations (simulated with a sleep).
- Decrements reader count, unlocking wrt if this is the last reader.
- Notifies all threads waiting on the condition variable.
- Sleeps briefly before attempting to read again.

Parameters

reader_id	An integer identifier for the reader (for logging).

Here is the call graph for this function:



Here is the caller graph for this function:

main
WritersPriority::reader

3.2.3.2 writer()

```
void WritersPriority::writer (
                    int writer_id)    [inline]
```

Function executed by writer threads.

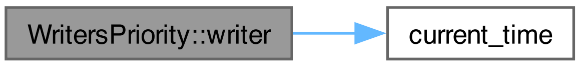
Each writer enters a loop:

- Increments the writer count.
- Waits until no readers are reading (read_count == 0) and wrt is not locked.
- Locks wrt and sets wrt_locked to true.
- Performs writing operations (simulated with a sleep).
- Decrements writer count, unlocks wrt, and resets wrt_locked.
- Notifies all threads waiting on the condition variable.
- Sleeps briefly before attempting to write again.

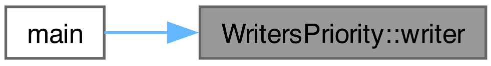
Parameters

writer_id	An integer identifier for the writer (for logging).

Here is the call graph for this function:



Here is the caller graph for this function:



3.2.4 Member Data Documentation

3.2.4.1 cond

std::condition_variable WritersPriority::cond [private]

Condition variable to signal state changes to waiting threads.

3.2.4.2 cout_mtx

std::mutex WritersPriority::cout_mtx [private]

Mutex to protect output operations (e.g. writing to std::cout).

3.2.4.3 mtx

std::mutex WritersPriority::mtx [private]

Mutex to protect shared state (e.g., counters, flags).

3.2.4.4 prefer_writers

bool WritersPriority::prefer_writers = true [private]

Flag to indicate if the system currently prefers writers over readers.

3.2.4.5 read_count

int WritersPriority::read_count = 0 [private]

Current number of active readers.

3.2.4.6 write_count

int WritersPriority::write_count = 0 [private]

Current number of writers either waiting or writing.

3.2.4.7 wrt

std::mutex WritersPriority::wrt [private]

Mutex used by writers. If locked, a writer is writing (or waiting to write).

3.2.4.8 wrt_locked

bool WritersPriority::wrt_locked = false [private]

Indicates whether the wrt mutex is currently locked by a writer.

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

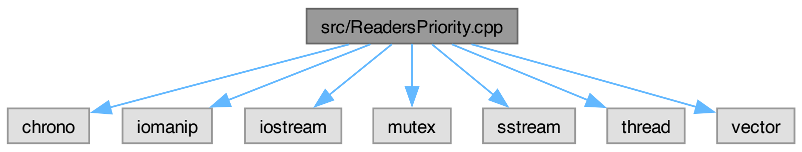
- src/WritersPriority.cpp

Chapter 4

File Documentation

4.1 src/ReadersPriority.cpp File Reference

#include <chrono>
#include <iomanip>
#include <iostream>
#include <mutex>
#include <sstream>
#include <thread>
#include <vector>
Include dependency graph for ReadersPriority.cpp:



Classes

- class ReadersPriority

    Implements the Readers Priority solution to the Readers-Writers problem.

Macros

- #define RESET "\033[0m"

    ANSI color reset code.
- #define GREEN "\033[32m"

    ANSI color code for green text.
- #define RED "\033[31m"

    ANSI color code for red text.

Functions

- std::string current_time ()

  Retrieves the current local time in a string format with milliseconds.
- int main ()

  Main function where reader and writer threads are created.

4.1.1 Macro Definition Documentation

4.1.1.1 GREEN

#define GREEN "\033[32m"

ANSI color code for green text.

4.1.1.2 RED

#define RED "\033[31m"

ANSI color code for red text.

4.1.1.3 RESET

#define RESET "\033[0m"

ANSI color reset code.

4.1.2 Function Documentation

4.1.2.1 current_time()

std::string current_time ()

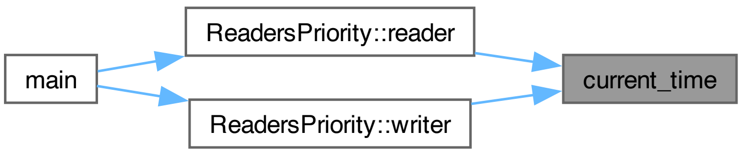
Retrieves the current local time in a string format with milliseconds.

This function uses std::chrono to get the current system time, and formats it into a string in the format YYYY-MM-DD HH:MM:SS.mmm.

Returns

A string representing the current local time with millisecond precision.

Here is the caller graph for this function:



4.1.2.2 main()

int main ()

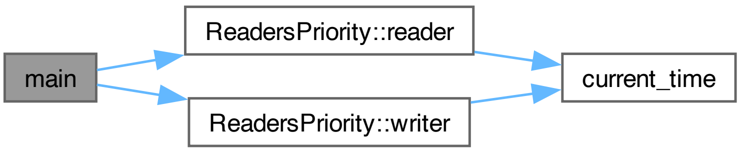
Main function where reader and writer threads are created.

Creates multiple reader threads and writer threads. Each thread runs indefinitely, demonstrating the Readers Priority approach for the Readers-Writers problem.

Returns

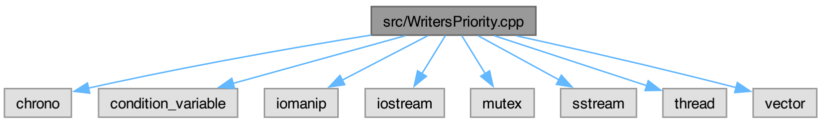
Exit code (0 for normal termination).

Here is the call graph for this function:



4.2 src/WritersPriority.cpp File Reference

#include <chrono>
#include <condition_variable>
#include <iomanip>
#include <iostream>
#include <mutex>
#include <sstream>
#include <thread>
#include <vector>
Include dependency graph for WritersPriority.cpp:



Classes

- class WritersPriority

  Implements a Writers Priority solution to the Readers-Writers problem.

Macros

- #define RESET "\033[0m"

  ANSI color reset code.
- #define GREEN "\033[32m"

  ANSI color code for green text (used by readers).
- #define RED "\033[31m"

  ANSI color code for red text (used by writers).

Functions

- std::string current_time ()

  Retrieves the current local time in a string format with milliseconds (thread-safe).
- int main ()

  Main function where reader and writer threads are created.

4.2.1 Macro Definition Documentation

4.2.1.1 GREEN

#define GREEN "\033[32m"

ANSI color code for green text (used by readers).

4.2.1.2 RED

#define RED "\033[31m"

ANSI color code for red text (used by writers).

4.2.1.3 RESET

#define RESET "\033[0m"

ANSI color reset code.

4.2.2 Function Documentation

4.2.2.1 current_time()

std::string current_time ()

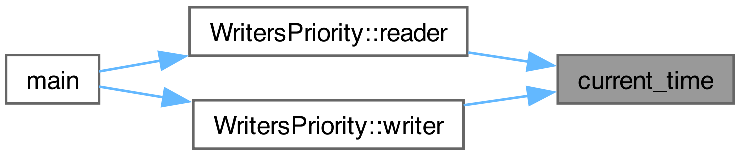
Retrieves the current local time in a string format with milliseconds (thread-safe).

This function uses std::chrono to get the current system time and formats it into a string in the format YYYY-MM-DD HH:MM:SS.mmm.

Returns

A string representing the current local time with millisecond precision.

Here is the caller graph for this function:



4.2.2.2 main()

int main ()

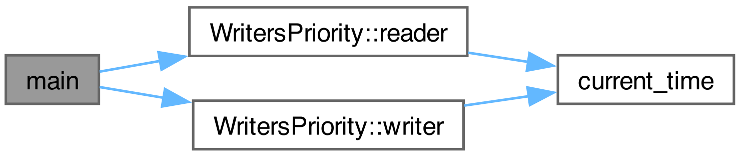
Main function where reader and writer threads are created.

Creates multiple reader threads and writer threads. Each thread runs indefinitely, demonstrating the Writers Priority approach for the Readers-Writers problem.

Returns

Exit code (0 for normal termination).

Here is the call graph for this function:



Index