Future energy storage methods for solar systems

Future energy storage methods for solar systems

std::future::~future

Releases any shared state. This means: If the current object holds the last reference to its shared state, the shared state is destroyed. The current object gives up its reference to its shared

std::future

The class template std::future provides a mechanism to access the result of asynchronous operations: An asynchronous operation (created via std::async, std::packaged_task,

Comprehensive review of energy storage systems technologies,

This paper presents a comprehensive review of the most popular energy storage systems including electrical energy storage systems, electrochemical energy storage systems, mechanical

std::shared_future

Unlike std::future, which is only moveable (so only one instance can refer to any particular asynchronous result), std::shared_future is copyable and multiple shared future objects

Renewable Energy Storage: Complete Guide To Technologies

Comprehensive guide to renewable energy storage technologies, costs, benefits, and applications. Compare battery, mechanical, and thermal storage systems for 2025.

std::future::get

The get member function waits (by calling wait ()) until the shared state is ready, then retrieves the value stored in the shared state (if any). Right after calling this function, valid () is false.

std::future::valid

Checks if the future refers to a shared state. This is the case only for futures that were not default-constructed or moved from (i.e. returned by std::promise::get_future (),

std::future_status

Specifies state of a future as returned by wait_for and wait_until functions of std::future and std::shared_future. Constants

std::future::wait_until

wait_until waits for a result to become available. It blocks until specified timeout_time has been reached or the result becomes available, whichever comes first. The return value indicates why

std::future::wait_for

If the future is the result of a call to std::async that used lazy evaluation, this function returns immediately without waiting. This function may block for longer than timeout_duration due to

Standard library header (C++11)

future (const future &) = delete; ~future (); future & operator =(const future &) = delete;

future & operator =(future &&) noexcept; shared_future share () noexcept; // retrieving the value

Contact Us

For catalog requests, pricing, or partnerships, please visit: https://www.peyronies.us