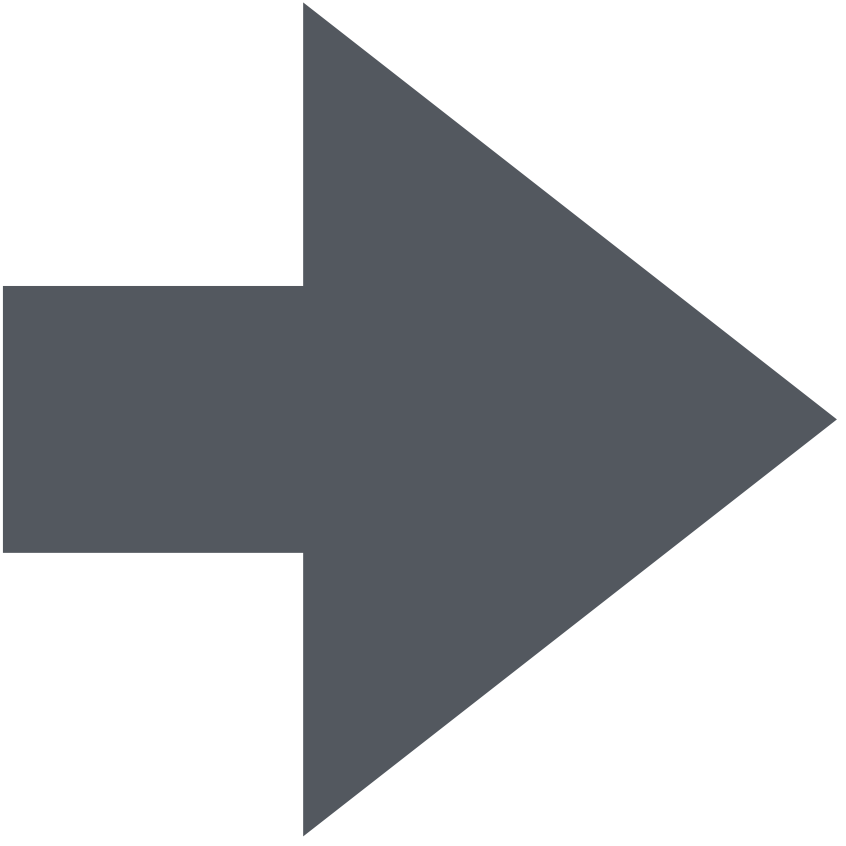
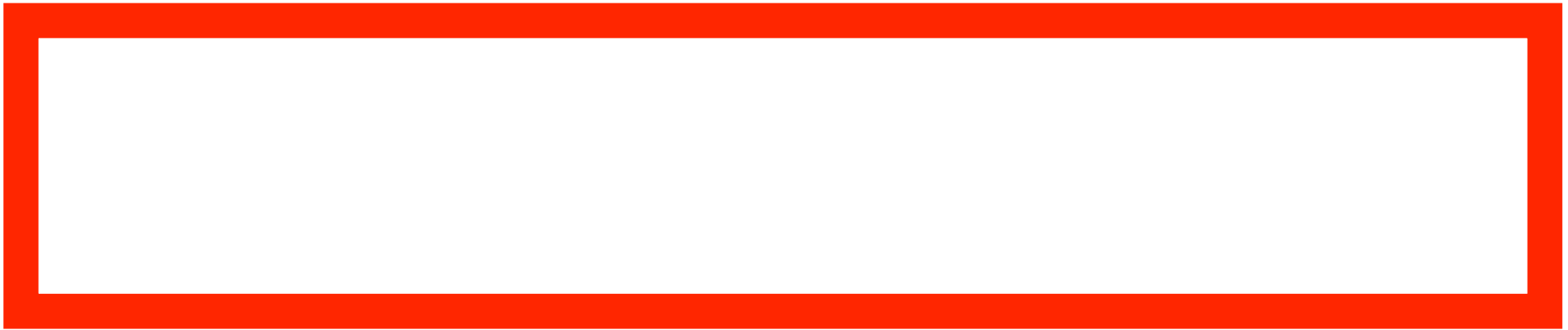
Implicit `shared_mutex` locking

66

```
class(shared_mutex) person
{
public:
    person() = default;

    std::string get_first_name() const
    {
        return first_name;
    }

    void set_first_name (std::string_view new_first)
    {
        first_name = new_first;
    }

    // Repeat for last_name

private:
    std::string first_name, last_name;
};
```

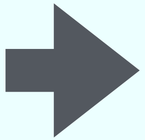
```
class person
{
public:
    person() = default;

    std::string get_first_name() const
    {
        std::shared_lock _ (mutex);
        return person_.get_first_name();
    }

    void set_first_name (std::string_view new_first)
    {
        std::unique_lock _ (mutex);
        person_.set_first_name (new_first);
    }

    // Repeat for last_name

private:
    class __person;
    std::shared_mutex mutex;
    mutable __person person_;
};

template<>
struct is_sync<person> : std::true_type {};
```





Implicit `shared_mutex` locking

```
class(shared_mutex) person
{
public:
    person() = default;

    std::string get_first_name() const
    {
        return first_name;
    }

    void set_first_name (std::string_view new_first)
    {
        first_name = new_first;
    }

    // Repeat for last_name

private:
    std::string first_name, last_name;
};
```



```
class person
{
public:
    person() = default;

    std::string get_first_name() const
    {
        std::shared_lock _ (mutex);
        return person_.get_first_name();
    }

    void set_first_name (std::string_view new_first)
    {
        std::unique_lock _ (mutex);
        person_.set_first_name (new_first);
    }

    // Repeat for last_name

private:
    class __person;
    std::shared_mutex mutex;
    mutable __person person_;
};

template<>
struct is_sync<person> : std::true_type {};
```

```cpp
void entry_point (std::shared_ptr<synchronized_value<std::string>> sync_s, int tid)
{
    apply ([tid] (auto& s) {
        s.append ("🔥");
        std::println ("{} {}", s, tid);
        return s;
    },
    *sync_s);
}

int main()
{
    auto p = std::make_shared<synchronized_value<std::string>> ("Hello threads");
    //...
}
```