



Implicit mutex locking

65

```
class(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;

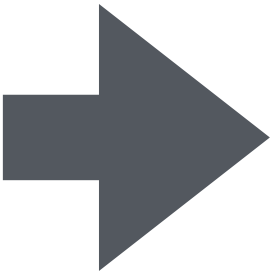
    std::string get_first_name() const
    {
        std::scoped_lock _ (mutex);
        return person_.get_first_name();
    }

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

    // Repeat for last_name

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

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











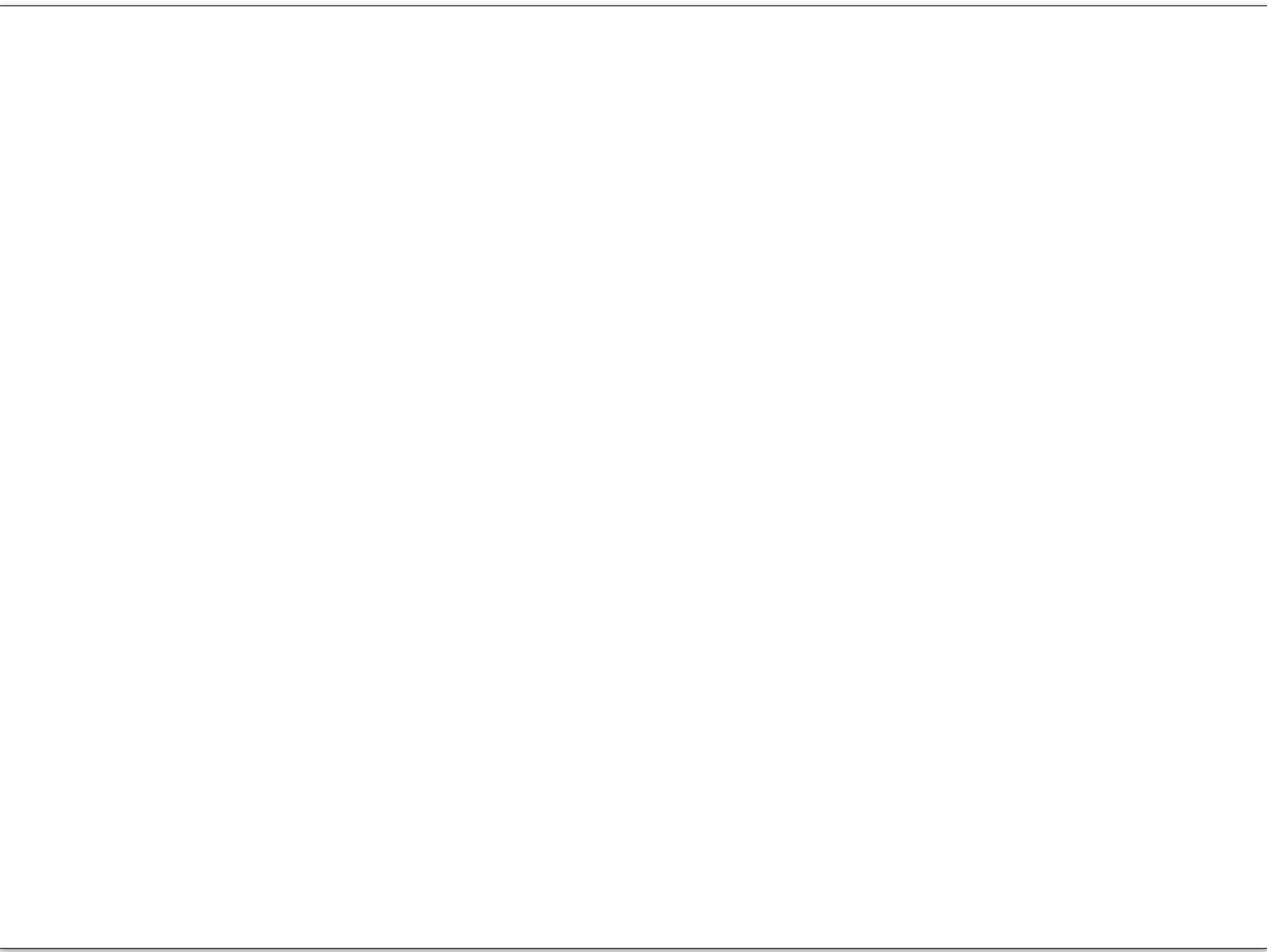
mutex

Implicit

locking

shared_mute_x





class/minutex

get_first_name()

(std::string_view





std::string

void

last _ name:

private:  

default:



first name,

first name





Repeat

cons +

person()

set first name

public:  

f

o

r

new friends!

first name:



neturun

persoon

last name



new_firrst)









class(shared_mutex)

std::scoped_lock

stcd: nmvtx

set first name

is synonymous

```
person_.get_first_name();
```

person._set_first_name

(std::string_view

get_first_name()

std::true_type

std::string

mutex;



void





template >

lastname

(new first):

stnucst

person()

Person's name:

(mutex):

default:

f

o

r

new_firrst)



multable

public



Repeat

class

class







neturun

cons +



per son

perison:

Personson

private:  







std::shared_lock

std::unique_lock

std::shared_mutex

locking



Personson



mutex;

