



- Each person has its own  
shared\_ptr instance

• This is never shared

- As soon as a non-const function is called, a unique copy is made

- The internal `__person` may be shared, but that's fine as there will only be *readers*

80

1

<https://godbolt.org/z/Kd57jV7V/z>

```
struct person
{
    std::string get_first_name() const {
        return person_>get_first_name();
    }

    void set_first_name (std::string_view new_first) {
        copy_if_shared();
        person_>set_first_name (new_first);
    }

    // Repeat for last_name

private:
    struct __person;
    static_assert (std::is_copy_constructible_v<__person>);

    std::shared_ptr<__person> person_
        = std::make_shared<__person>();

    void copy_if_shared() {
        if (person_.use_count() > 1)
            person_ = std::make_shared<__person> (*person_);
    }
};
```





Copy on Write structs





# Copy on Write `structs`

```
struct person
{
    std::string get_first_name() const {
        return person_>get_first_name();
    }

    void set_first_name (std::string_view new_first) {
        copy_if_shared();
        person_>set_first_name (new_first);
    }

    // Repeat for last_name

private:
    struct __person;
    static_assert (std::is_copy_constructible_v<__person>);

    std::shared_ptr<__person> person_
        = std::make_shared<__person>();

    void copy_if_shared() {
        if (person_.use_count() > 1)
            person_ = std::make_shared<__person> (*person_);
    }
};
```

- Each **person** has its own **shared\_ptr** instance
  - *This is never shared*
- As soon as a non-const function is called, a unique copy is made
- The internal **\_\_person** may be shared, but that's fine as there will only be *readers*



# Copy on Write **structs**

- Only works if there are no *pointers* or *references* to a **person**

```
struct person
{
    //...
    // Wrapped __person functions
    //...
};
```