```
template<typename T>
class(cow) cow_vector
{
public:
   //... vector-like definition forwarding
   //    to internal

private:
    std::vector<T> internal;
};
```

Cheat `inout`

84

```
void push_42 (cow_vector<int> v)
{
    v.push_back (42); // write creates copy
}
```

https://godbolt.org/z/fd9oGWoTo

```
cow_vector<int> vec;
vec.push_back (40);
vec.push_back (41);

push_42 (vec);

//... vec doesn't contain 42
```

```
void push_42 (cow_vector<int>* v)
{
    std::thread t ([v]
                   {
                       auto vec2 = *v; // create a copy
                   });

    v->push_back (42); // data-race here as copy_if_shared()
                       // checks use_count()!
}
```

```
cow_vector<int> vec;
vec.push_back (40);
vec.push_back (41);

push_42 (std::addressof (vec));

//... vec contains 42
```

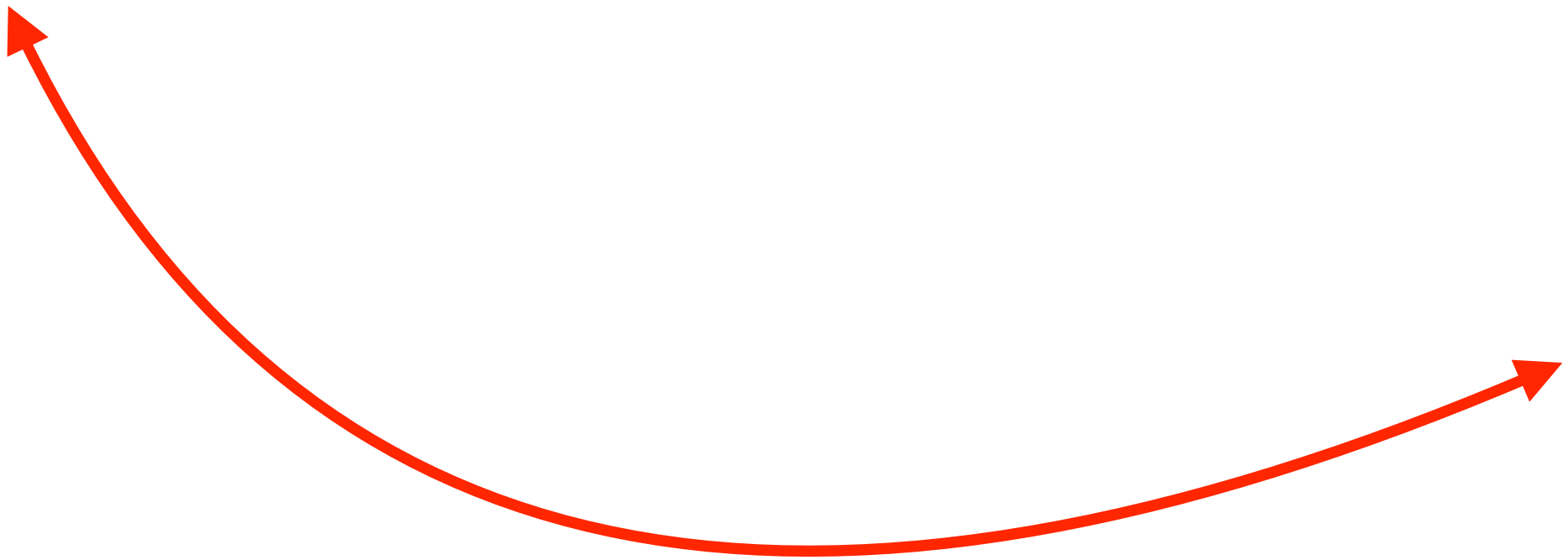
```
template<typename T>
class cow_vector
{
public:
   //... vector-like definition forwarding to internal

private:
   // std::shared_ptr wrapping and copy_if_shared as before
};

template<T>
struct is_send<cow_vector<T>>
    : is_send_v<__cow_vector<T>>{};
```

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

```
cow_vector (const cow_vector& o) {
    cow_vector_ = o.cow_vector_;
}
```



Cheat inout

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

```
cow_vector (const cow_vector& o) {
    cow_vector_ = o.cow_vector_;
}
```

```
void push_42 (cow_vector<int> v)
{
    v.push_back (42); // write creates copy
}
```

```
cow_vector<int> vec;
vec.push_back (40);
vec.push_back (41);

push_42 (vec);

//... vec doesn't contain 42
```

```
void push_42 (cow_vector<int>* v)
{
    std::thread t ([v]
                   {
                       auto vec2 = *v; // create a copy
                   });

    v->push_back (42); // data-race here as copy_if_shared()
                       // checks use_count()!
}
```

```
cow_vector<int> vec;
vec.push_back (40);
vec.push_back (41);

push_42 (std::addressof (vec));

//... vec contains 42
```

Modelled `inout`