Hidden COW
+ regular ELF
= broken Magic

robert_schneider_d@web.de

Allokatoren

```
using std::string = std::basic_string<char, std::char_traits<char>,
                                      std::allocator<char>>;
```

```
using cst::string = std::basic_string<char, std::char_traits<char>,
                                      cst::allocator<char>>;
```

stateful allocator
(statistics, polymorphism, ...)

Help, I can’t move!*

* but copying works

Multiple ELF

scalarfunction.hpp

```
class ScalarFunction
{
public:
  ScalarFunction(cst::string
name);
private:
  cst::string name_;
};
```

scalarfunction.so

```
ScalarFunction::
ScalarFunction(cst::string name)
  : name_(std::move(name))
{}
```

test_sinus (executable)

```
auto construct_sinus() -> ScalarFunction
{
  return ScalarFunction("sinus");
}
```

Help, I can’t move!

<u>Address Sanitizer</u>

```
ERROR: AddressSanitizer: attempting free on address which was not
malloc()-ed: 0x7fcbcf046100
```

<u>glibc</u>

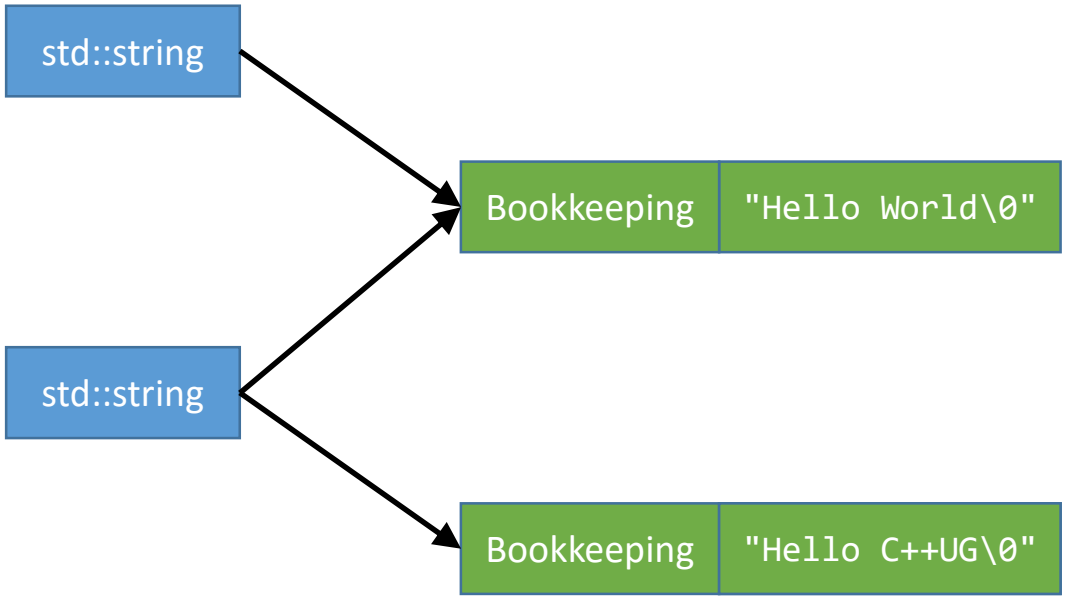
```
*** Error in './test_function': free(): invalid pointer:
0x0007fe301037060
```

Wie diagnostizieren?

- Kopieren funktioniert ➔ Move-Operation suspekt
- Allokator eingeführt ➔ Allokator suspekt
- basic_string's move-Operation im Debugger anschauen

gcc 4.8.5 auf Linux ➔ libstdc++ ➔ COW-Strings

COW-strings 101



```
// libstdc++ of gcc 4.8.5, basic_string.h
namespace std _GLIBCXX_VISIBILITY(default)
{
  template<typename _CharT, typename _Traits,
           typename _Alloc>
    class basic_string
    {
      ~basic_string() _GLIBCXX_NOEXCEPT
      {
        _M_rep()->_M_dispose(
          this->get_allocator());
      }

      struct _Rep : _Rep_base
      {
        static size_type
        _S_empty_rep_storage[];

        void
        _M_dispose(const _Alloc& __a);
      };
    };
} // namespace
```

```
void
M_dispose(const _Alloc& __a)
{
  if (this != &_S_empty_rep())
    {
      if (__exchange_and_add_dispatch(
          &this->_M_refcount, -1) <= 0)
        {
          _M_destroy(__a);
        }
    }
}
```

Move-construction

```
basic_string(basic_string&& __str) noexcept
: _M_dataplus(__str._M_dataplus)
{
  __str._M_data(_S_empty_rep()._M_refdata());
}
```

ELF + Templates

```
class visible common_dough {};
class hidden  secret_dough {};

class visible choc_chips   {};

template<typename Dough, typename Extra>
class visible cookie
{
  void eat(mouth&);
};


void visible cookie<common_dough, choc_chips>::eat(mouth&)

void hidden  cookie<secret_dough, choc_chips>::eat(mouth&)
```

Allokatoren: Sichtbarkeit

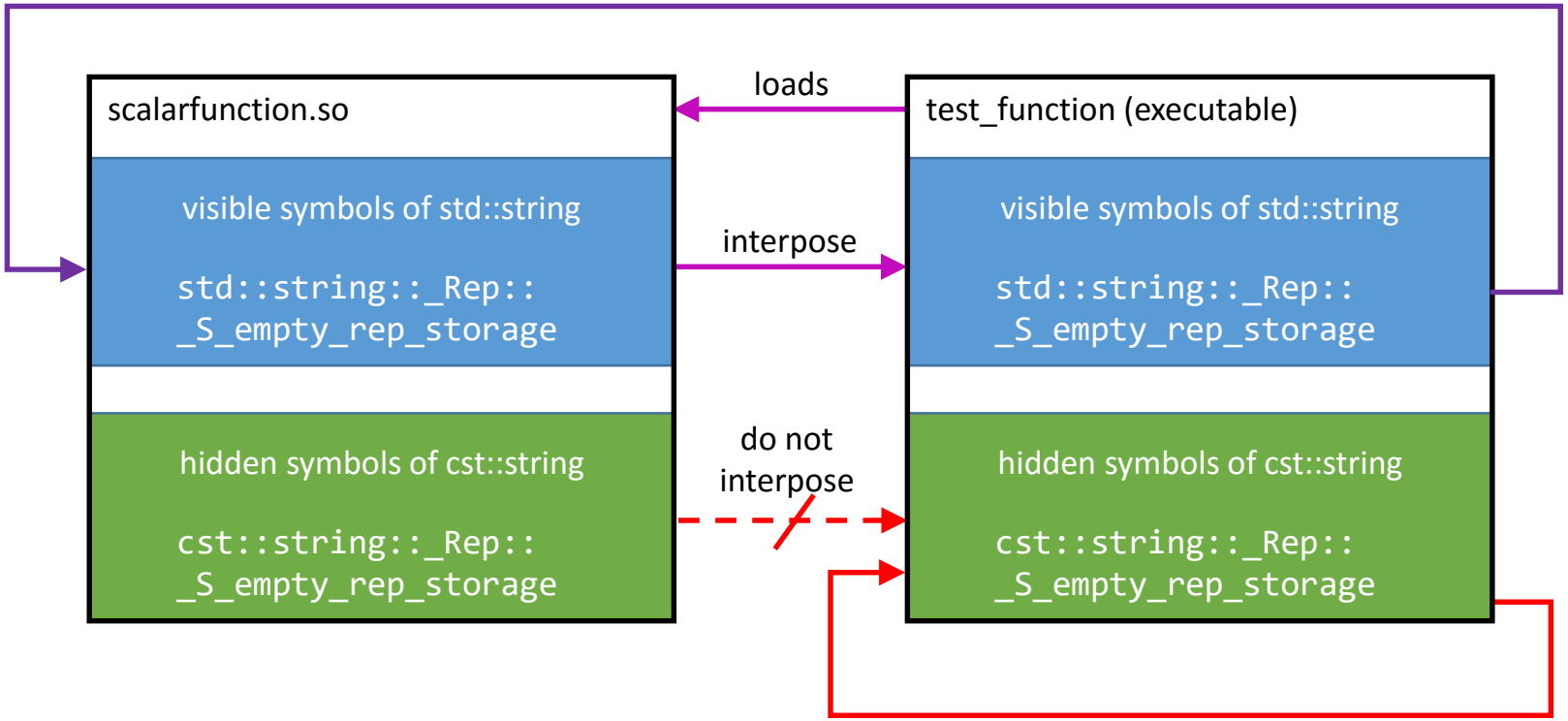
- gcc -fvisibility=hidden
- namespace std _GLIBCXX_VISIBILITY(default)

```
using std::string = std::basic_string<char, std::char_traits<char>,
                                      std::allocator<char>>;
```

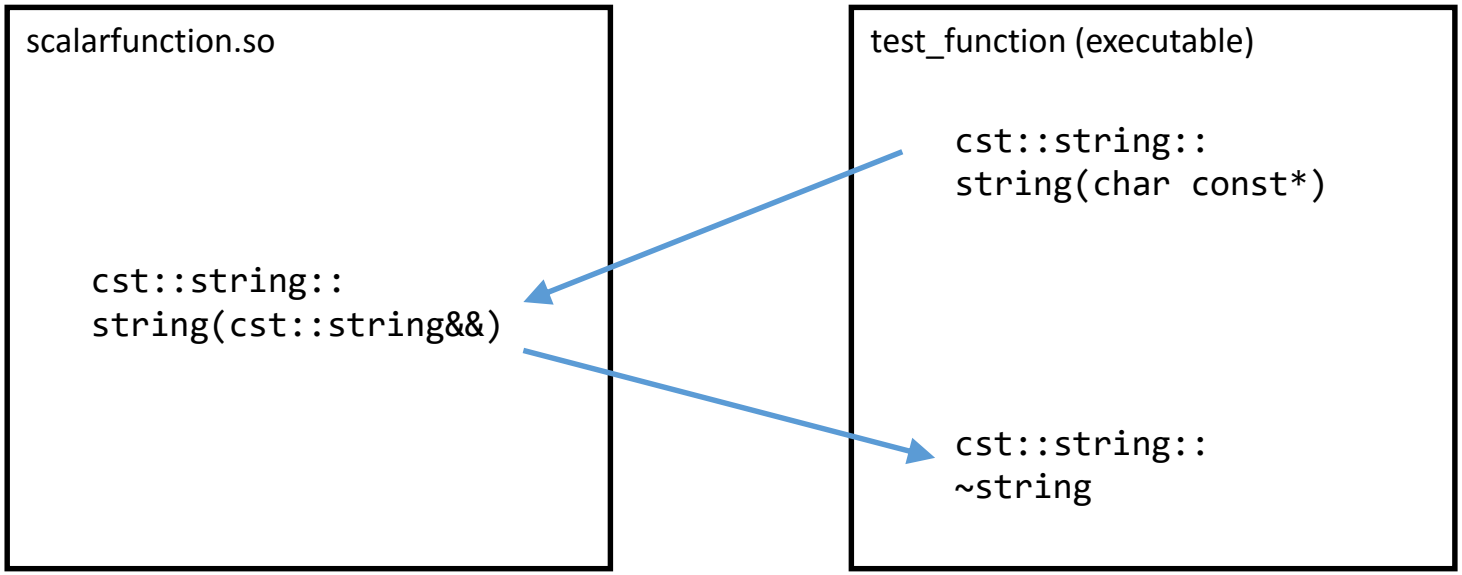
```
using cst::string = std::basic_string<char, std::char_traits<char>,
                                      cst::allocator<char>>;
```

nicht explizit exportiert

Multiple ELF – multiple symbols



Not my empty string



See also

- How To Write Shared Libraries, Ulrich Drepper (PDF)
- The strange details of std::string at Facebook, Nicholas Ormrod (CppCon 2016 talk)