118

Avoiding ABI Breaks: Extrinsic Storage

```
constexpr const_reference operator[](size_type __pos) const noexcept {
  _LIBCPP_ASSERT_VALID_ELEMENT_ACCESS(__pos <= size(), "string index out of bounds”);
  scoped_check<check_type::read> _ (data_race_registry::get_state (this));

  if (__builtin_constant_p(__pos) && !__fits_in_sso(__pos))
    return *(__get_long_pointer() + __pos);

  return *(data() + __pos);
}
```

```
class data_race_registry {
    static inline auto tags     = extrinsic_storage<check_state>{};

public:
   static inline auto get_state(void* pobj) noexcept {
        return *tags.find_or_insert(pobj);
    }

    static inline auto on_destroy(void* pobj) noexcept -> void {
        tags.erase(pobj);
    }
};
```

Avoiding ABI Breaks: Extrinsic Storage

```
class data_race_registry {
    static inline auto tags    = extrinsic_storage<check_state>{};

public:
   static inline auto get_state(void* pobj) noexcept {
        return *tags.find_or_insert(pobj);
    }

    static inline auto on_destroy(void* pobj) noexcept -> void {
        tags.erase(pobj);
    }
};
```

```
constexpr const_reference operator[](size_type __pos) const noexcept {
  _LIBCPP_ASSERT_VALID_ELEMENT_ACCESS(__pos <= size(), "string index out of bounds");
  scoped_check<check_type::read> _ (data_race_registry::get_state (this));

  if (__builtin_constant_p(__pos) && !__fits_in_sso(__pos))
    return *(__get_long_pointer() + __pos);

  return *(data() + __pos);
}
```

Data Races as Contract Violations

```
constexpr const_reference operator[](size_type __pos) const noexcept {
  _LIBCPP_ASSERT_VALID_ELEMENT_ACCESS(__pos <= size(), "string index out of bounds");
  scoped_check<check_type::read> _ (data_race_registry::get_state (this));

  if (__builtin_constant_p(__pos) && !__fits_in_sso(__pos))
    return *(__get_long_pointer() + __pos);

  return *(data() + __pos);
}
```