Unifying `source_location` and `contract_violation`

Document #: D163PR0
Date: 2019-04-27
Project: Programming Language C++
Audience: LEWG, LWG
Reply-to: Corentin Jabot <corentin.jabot@gmail.com>

Proposed change

We propose that `contract_violation` uses `source_location` to report the location where a contract violation happens. The goal is to avoid API duplication and to make it easier to log contract violations in systems designed around `source_location`.

This modification matches the original intent of [P0542] as discussed in Kona 2017.

Note that `source_location::file_name` and `source_location::function_name` return a `const char*` unlike `contract_violation` whose `function_name` and `file_name` return a `string_view`. However, while LEWG has reaffirmed several times the design of `source_location`, we found no explanation why `contract_violation` is diverging from that design and the reasoning motivating `source_location`'s design equally applies to `contract_violation`.

Applicability

This papers depends on [P1208] being accepted by LWG. It has have been accepted by LEWG in Kona 2019.

Wording

� Class `contract_violation` [support.contract.cviol]

```
namespace std {
        class contract_violation {
                public:
                ~~uint_least32_t line_number() const noexcept;~~
                ~~string_view file_name() const noexcept;~~
                ~~string_view function_name() const noexcept;~~
                source_location location() const noexcept;
                string_view comment() const noexcept;
                string_view assertion_level() const noexcept;
        };
}
```

The class `contract_violation` describes information about a contract violation generated by the implementation.

```
uint_least32_t line_number() const noexcept;
```

Returns: The source code location where the contract violation happened. If the location is unknown, an implementation may return `0`.

```
string_view file_name() const noexcept;
```

Returns: The source file name where the contract violation happened. If the file name is unknown, an implementation may return `string_view{}`.

```
string_view function_name() const noexcept;
```

Returns: The name of the function where the contract violation happened. If the function name is unknown, an implementation may return `string_view{}`.

```
source_location location() const noexcept;
```

Returns: The source code location where the contract violation happened. If the location is unknown, an implementation may return a default constructed `source_location`.

```
string_view comment() const noexcept;
```

Returns: Implementation-defined text describing the predicate of the violated contract.

```
string_view assertion_level() const noexcept;
```

Returns: Text describing the *assertion-level* of the violated contract.

References

[P0542] G. Dos Reis, J. D. Garcia, J. Lakos, A. Meredith, N. Myers, B. Stroustrup *Support for contract based programming in C++* https://wg21.link/P0542

[P1208] Robert Douglas, Corentin Jabot *Adopt source location from Library Fundamentals V3 for C++20* https://wg21.link/P1208