Property-testing all the things in SerenityOS

Martin Janiczek
@janiczek

Property-testing all* the things in SerenityOS

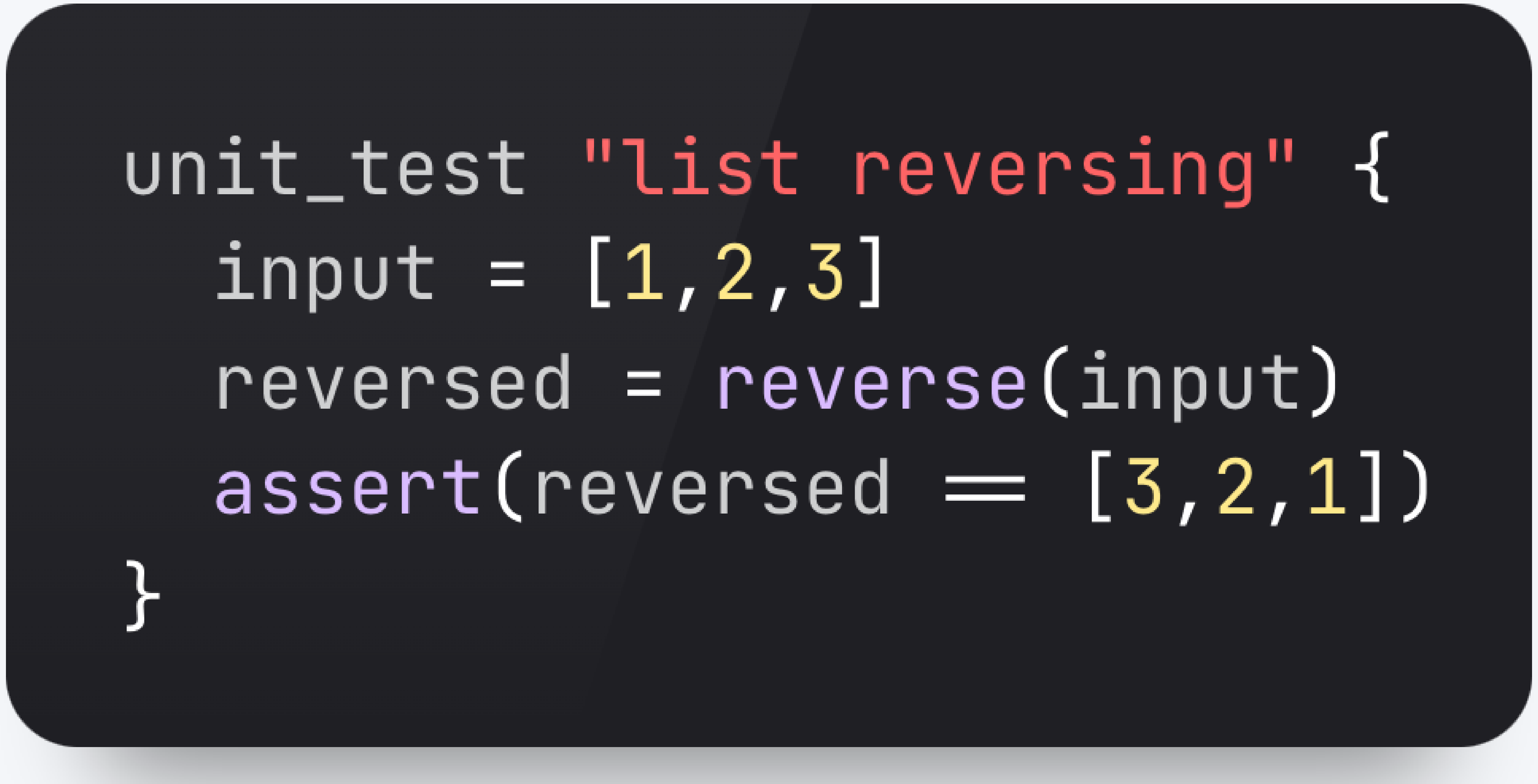
Martin Janiczek

@janiczek

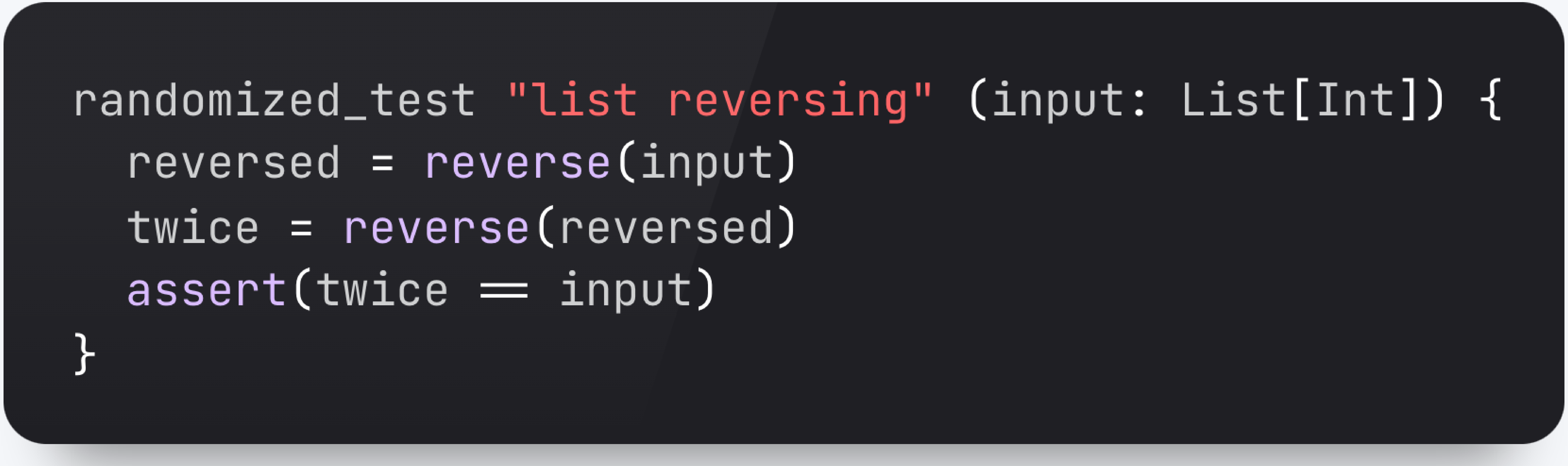
I love PBT!

I love **PBT**!**?**

Property Based Testing



```
unit_test "list reversing" {
  input = [1,2,3]
  reversed = reverse(input)
  assert(reversed == [3,2,1])
}
```



```
randomized_test "list reversing" (input: List[Int]) {
  reversed = reverse(input)
  twice = reverse(reversed)
  assert(twice == input)
}
```

Property Based Testing

Test with many random inputs

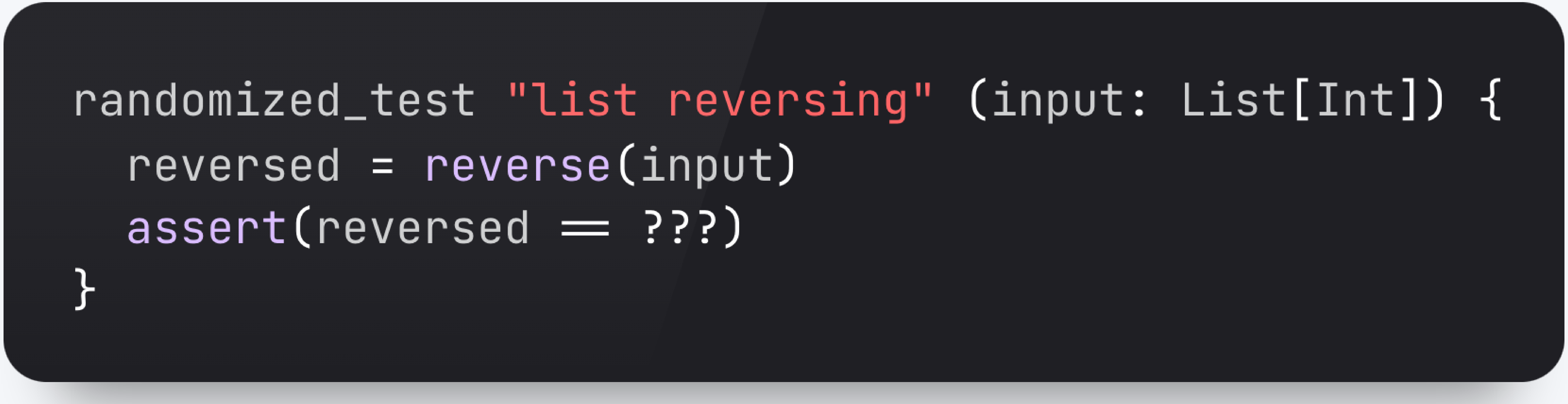
Property Based Testing

Test with many random inputs

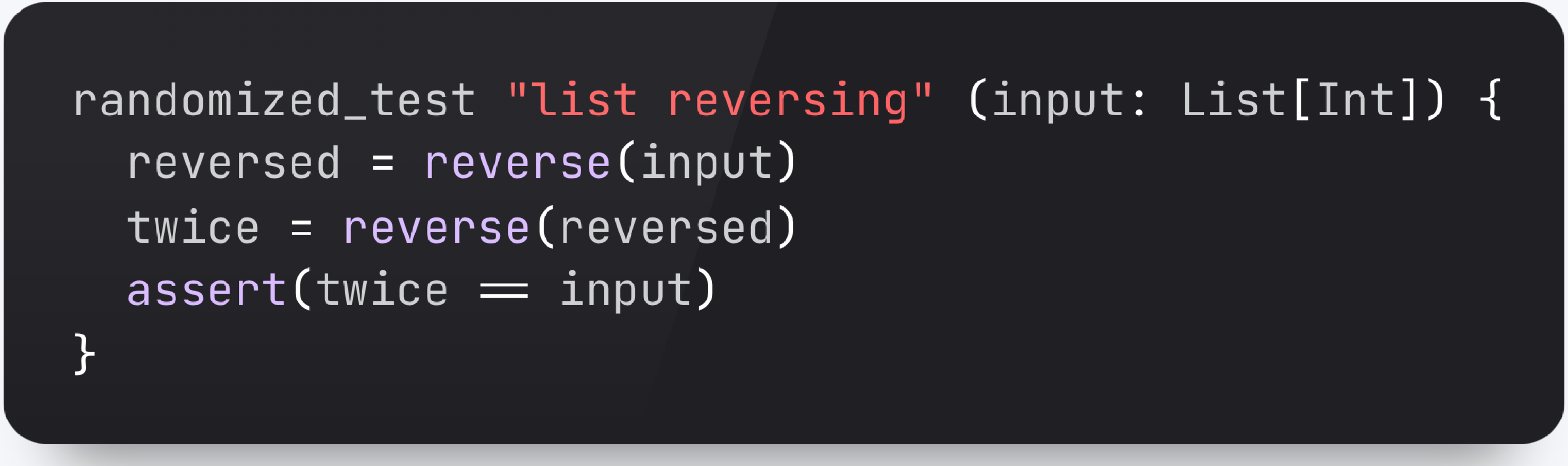
Finds a minimal example of a failure

Property Based Testing

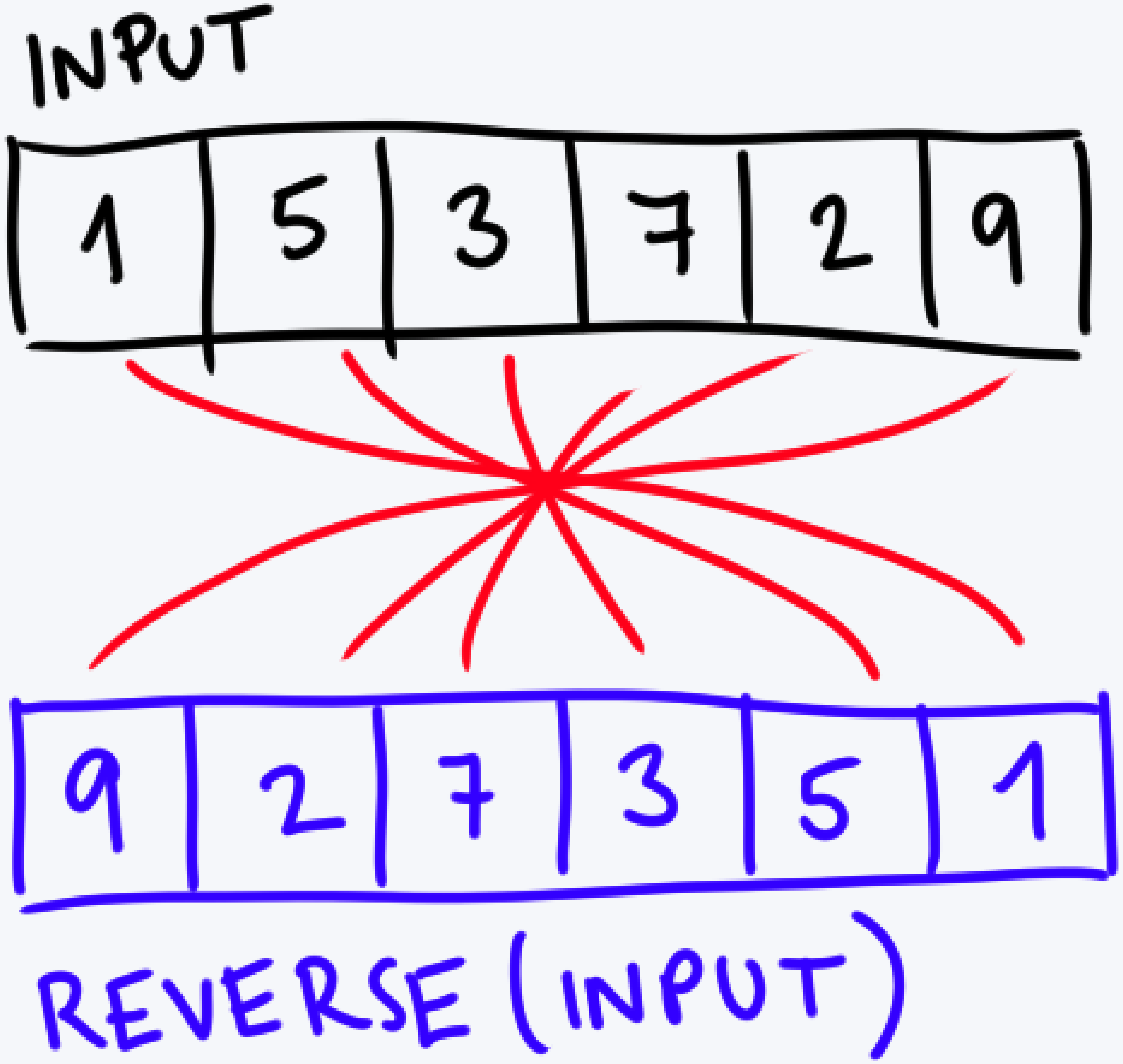
Test with many random inputs
Finds a minimal example of a failure
Focus on specification, not specific examples



```
randomized_test "list reversing" (input: List[Int]) {
  reversed = reverse(input)
  assert(reversed == ???)
}
```



```
randomized_test "list reversing" (input: List[Int]) {
  reversed = reverse(input)
  twice = reverse(reversed)
  assert(twice == input)
}
```



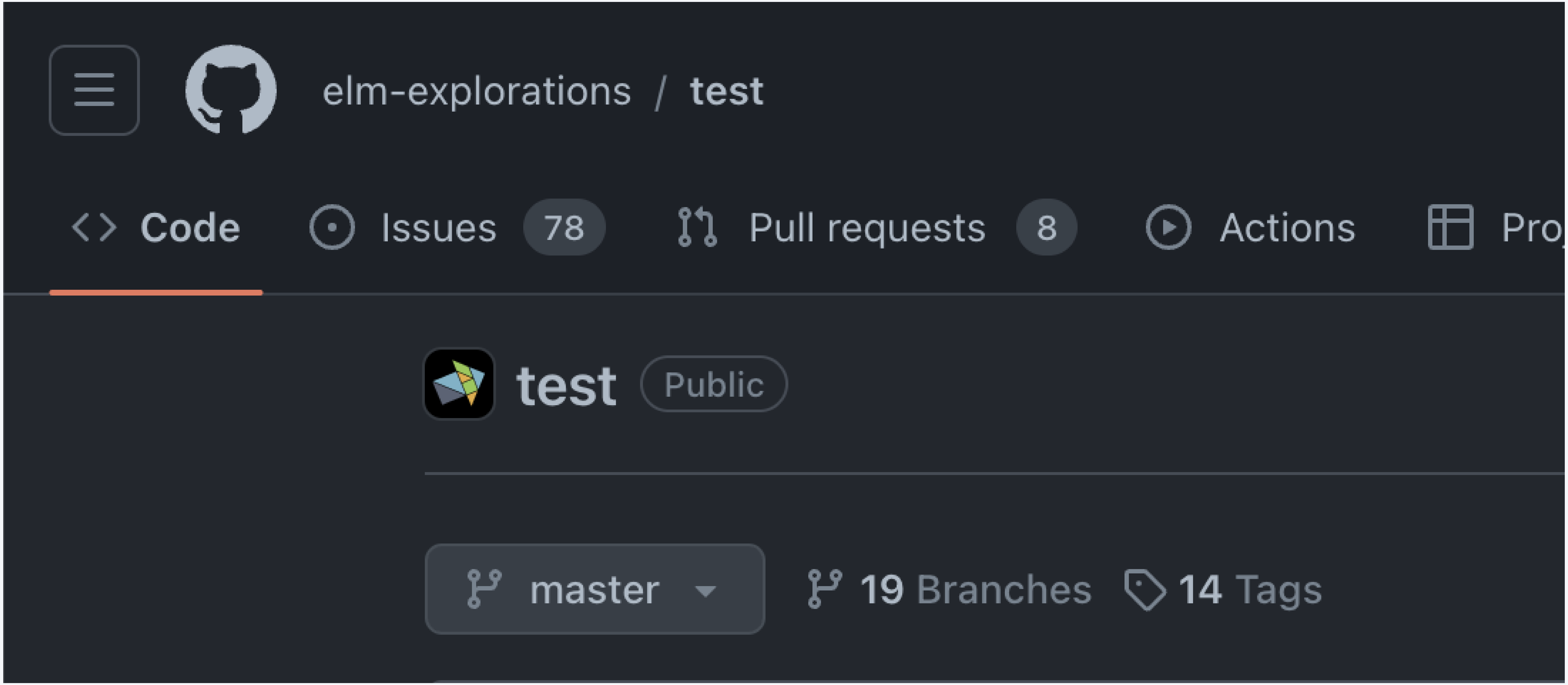
INPUT

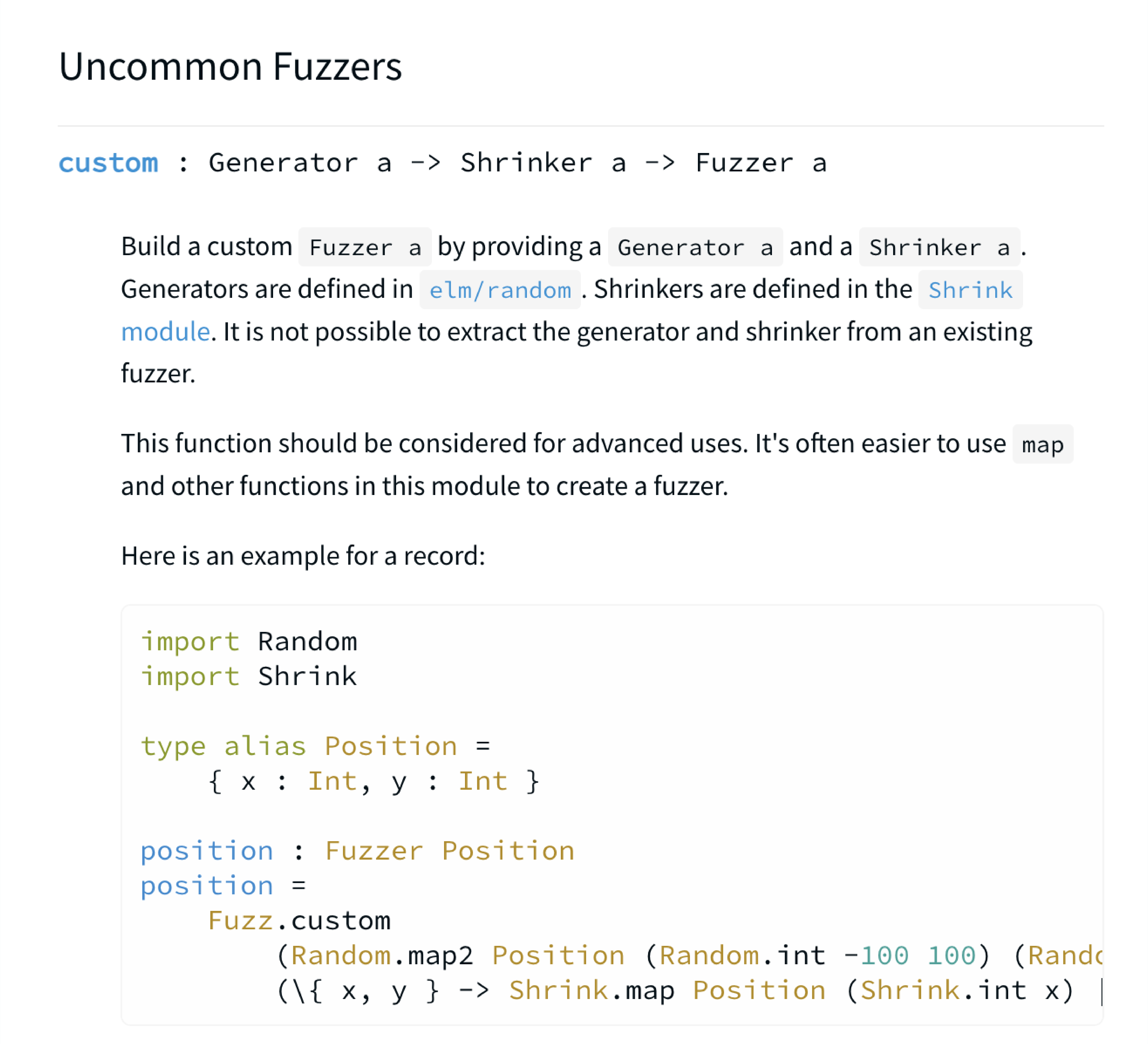
1	5	3	7	2	9

9	2	7	3	5	1

REVERSE(INPUT)

I love PBT!


elm-explorations / test
Code
Issues 78
Pull requests 8
Actions
Pro
test
Public
master
19 Branches
14 Tags



Uncommon Fuzzers

custom : Generator a -> Shrinker a -> Fuzzer a

Build a custom `Fuzzer a` by providing a `Generator a` and a `Shrinker a`. Generators are defined in `elm/random`. Shrinkers are defined in the `Shrink` module. It is not possible to extract the generator and shrinker from an existing fuzzer.

This function should be considered for advanced uses. It's often easier to use `map` and other functions in this module to create a fuzzer.

Here is an example for a record:

```
import Random
import Shrink

type alias Position =
    { x : Int, y : Int }

position : Fuzzer Position
position =
    Fuzz.custom
        (Random.map2 Position (Random.int -100 100) (Rando
        (\{ x, y } -> Shrink.map Position (Shrink.int x) |
```

Approaches

Approaches

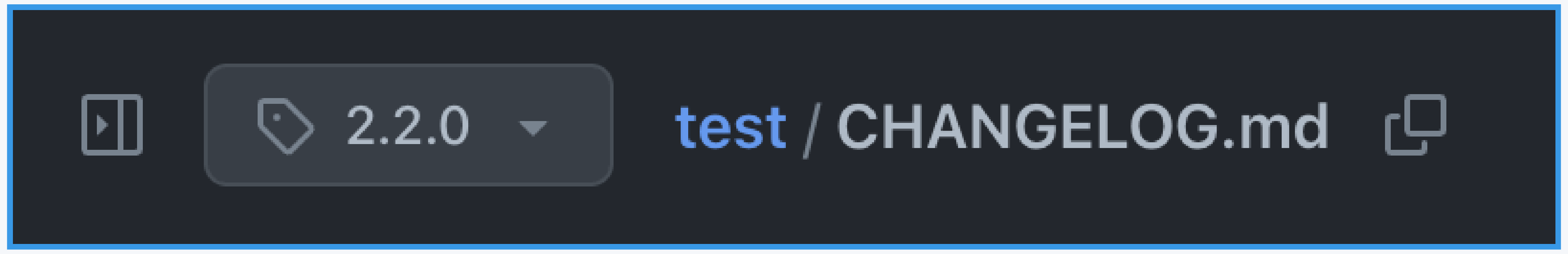
1. QuickCheck   doesn’t keep constraints

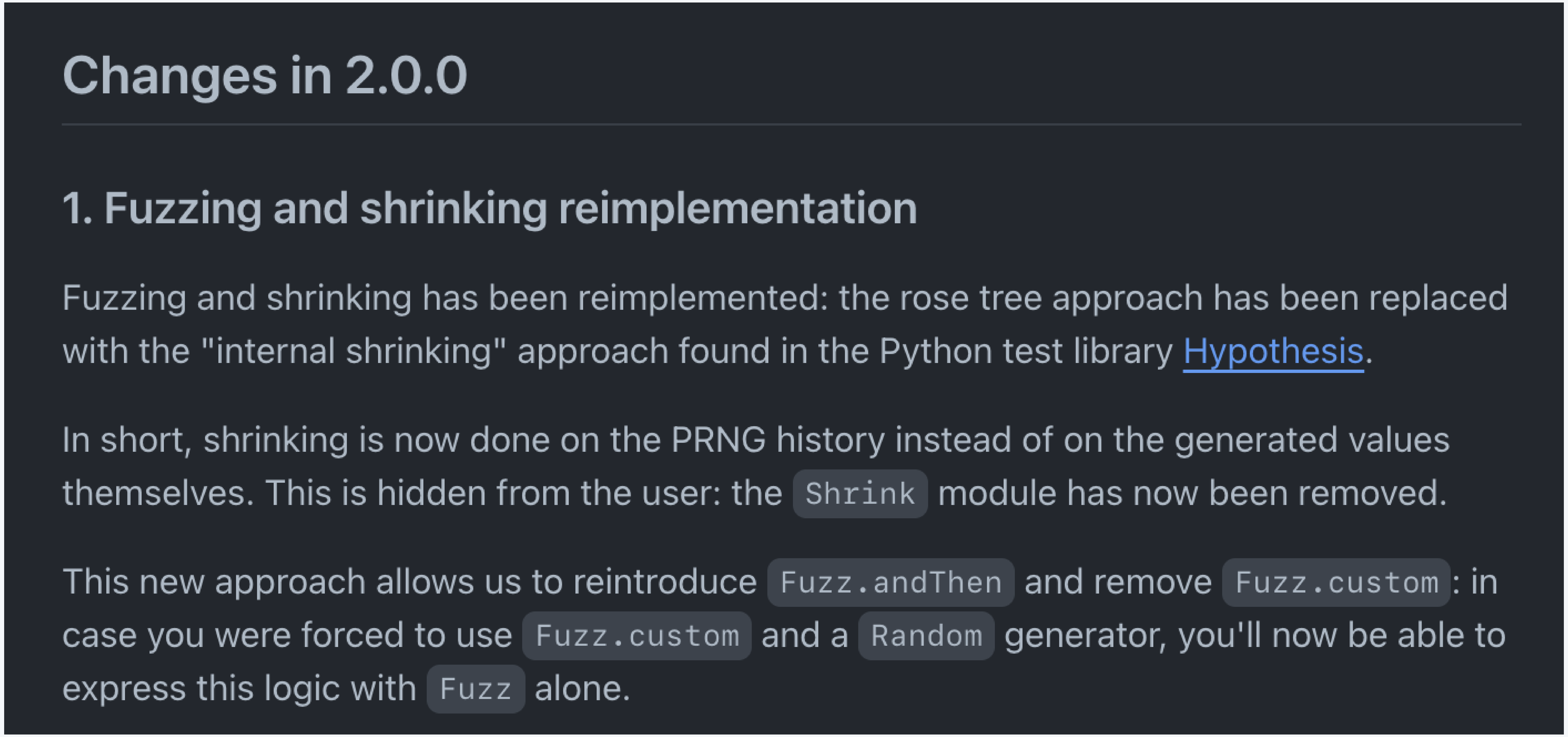
Approaches

1. QuickCheck doesn’t keep constraints
2. Hedgehog suffers when monadic bind is used

Approaches

1. QuickCheck	doesn’t keep constraints
2. Hedgehog	suffers when monadic bind is used
3. Hypothesis	...actually pretty awesome?





Changes in 2.0.0

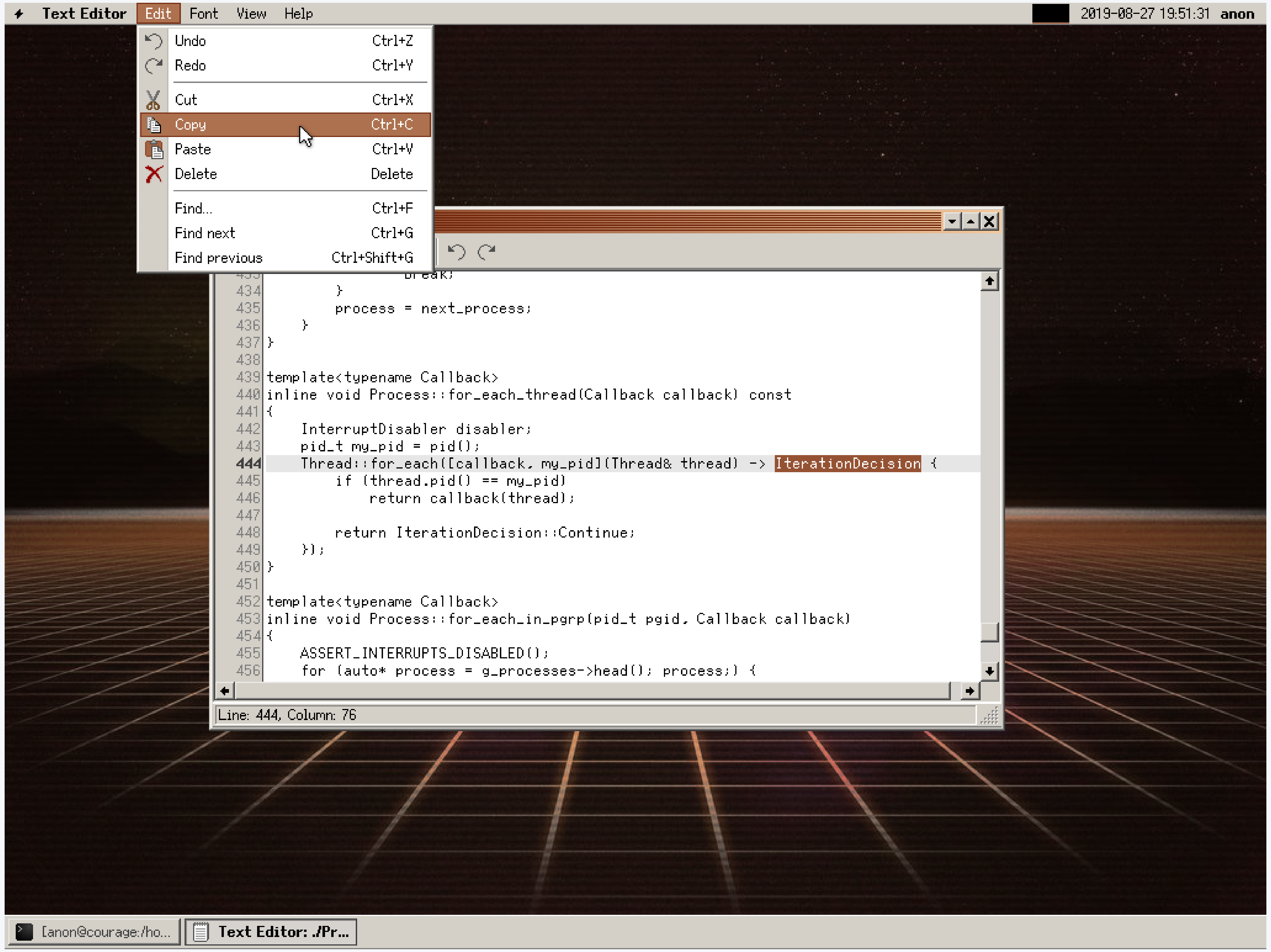
1. Fuzzing and shrinking reimplementation

Fuzzing and shrinking has been reimplemented: the rose tree approach has been replaced with the "internal shrinking" approach found in the Python test library Hypothesis.

In short, shrinking is now done on the PRNG history instead of on the generated values themselves. This is hidden from the user: the `Shrink` module has now been removed.

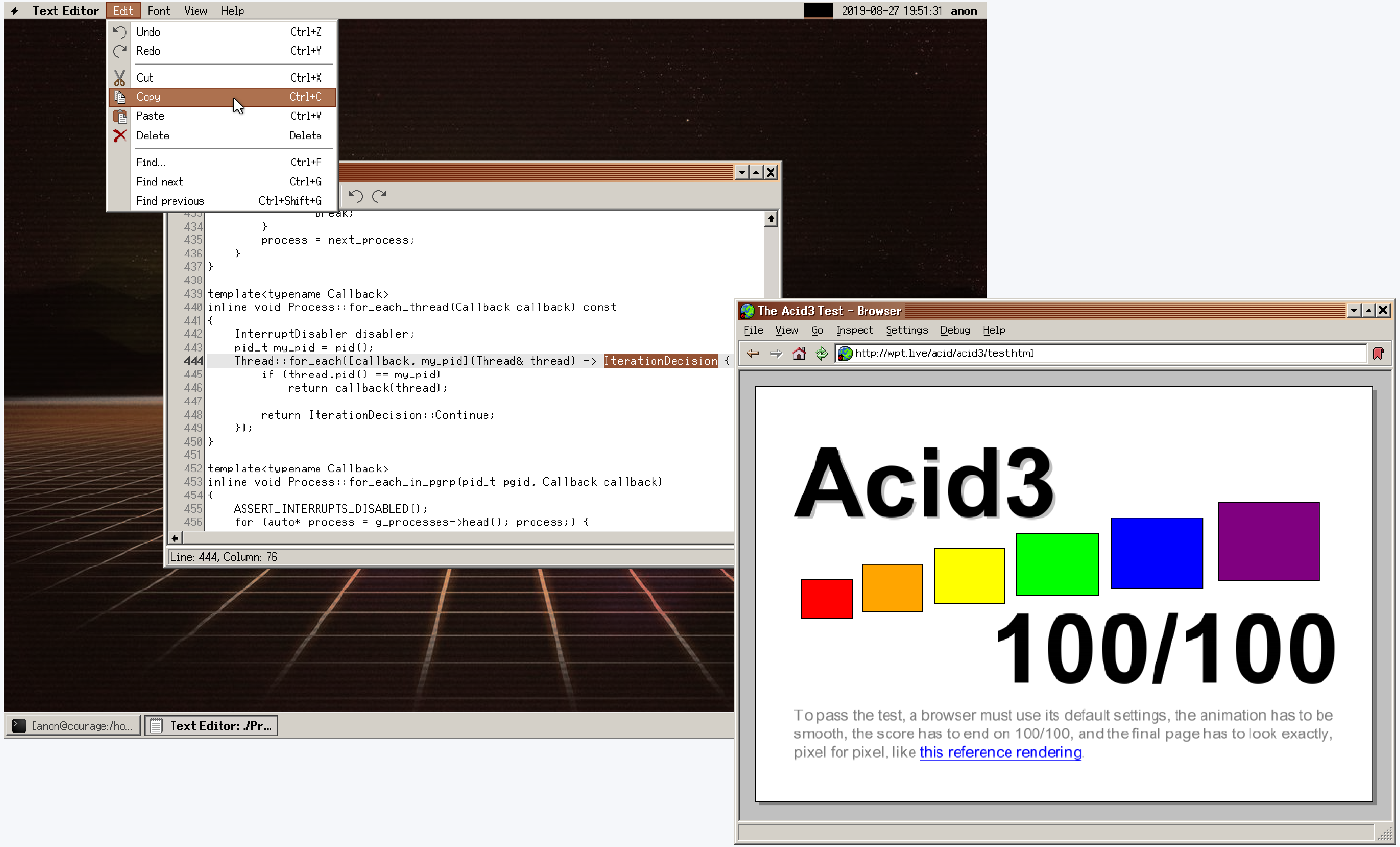
This new approach allows us to reintroduce `Fuzz.andThen` and remove `Fuzz.custom`: in case you were forced to use `Fuzz.custom` and a `Random` generator, you'll now be able to express this logic with `Fuzz` alone.

I admire SerenityOS!



```
            break;
        }
        process = next_process;
    }
}

template<typename Callback>
inline void Process::for_each_thread(Callback callback) const
{
    InterruptDisabler disabler;
    pid_t my_pid = pid();
    Thread::for_each([callback, my_pid](Thread& thread) -> IterationDecision {
        if (thread.pid() == my_pid)
            return callback(thread);

        return IterationDecision::Continue;
    });
}

template<typename Callback>
inline void Process::for_each_in_pgrp(pid_t pgid, Callback callback)
{
    ASSERT_INTERRUPTS_DISABLED();
    for (auto* process = g_processes->head(); process;) {
```

Line: 444, Column: 76



```
            break;
        }
        process = next_process;
    }
}

template<typename Callback>
inline void Process::for_each_thread(Callback callback) const
{
    InterruptDisabler disabler;
    pid_t my_pid = pid();
    Thread::for_each([callback, my_pid](Thread& thread) -> IterationDecision {
        if (thread.pid() == my_pid)
            return callback(thread);

        return IterationDecision::Continue;
    });
}

template<typename Callback>
inline void Process::for_each_in_pgrp(pid_t pgid, Callback callback)
{
    ASSERT_INTERRUPTS_DISABLED();
    for (auto* process = g_processes->head(); process;) {
```

Line: 444, Column: 76

Acid3

100/100

To pass the test, a browser must use its default settings, the animation has to be smooth, the score has to end on 100/100, and the final page has to look exactly, pixel for pixel, like this reference rendering.

I love PBT!

I love PBT!

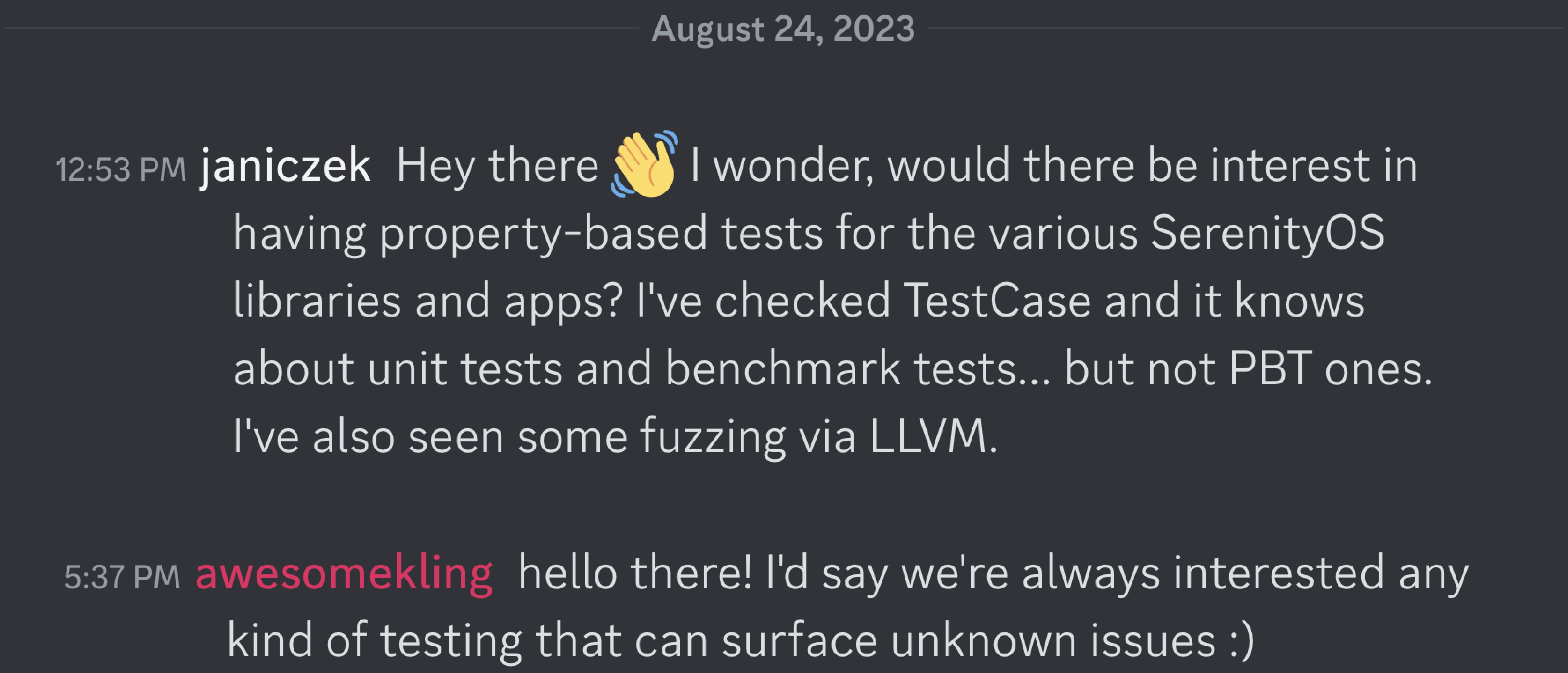
I admire SerenityOS!

I love **PBT!**

I admire **SerenityOS!**

TAKE A
THIGH

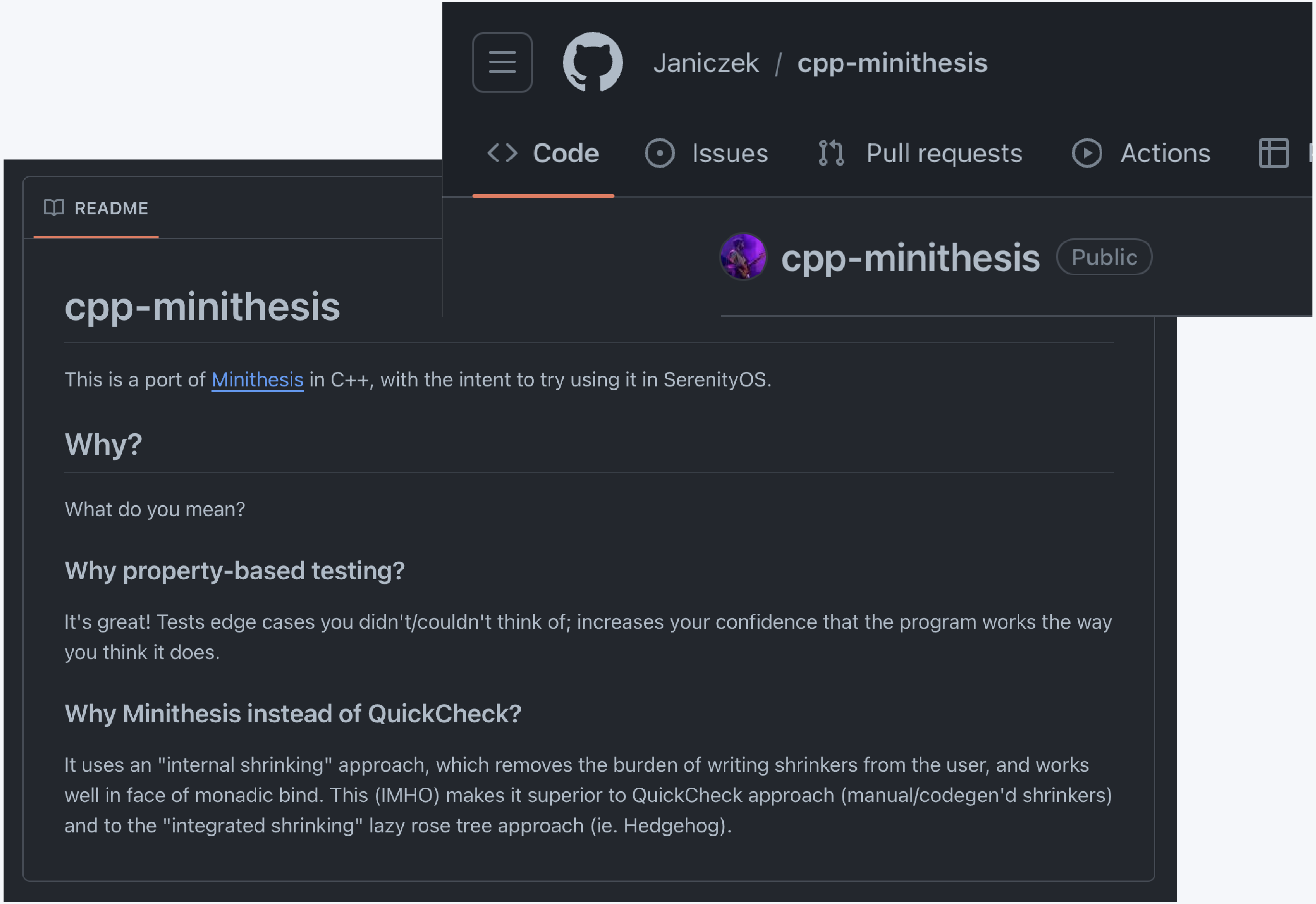
TAKE
THIG



August 24, 2023

12:53 PM **janiczek** Hey there 👋 I wonder, would there be interest in having property-based tests for the various SerenityOS libraries and apps? I've checked TestCase and it knows about unit tests and benchmark tests... but not PBT ones. I've also seen some fuzzing via LLVM.

5:37 PM **awesomekling** hello there! I'd say we're always interested any kind of testing that can surface unknown issues :)



cpp-minithesis

This is a port of Minithesis in C++, with the intent to try using it in SerenityOS.

Why?

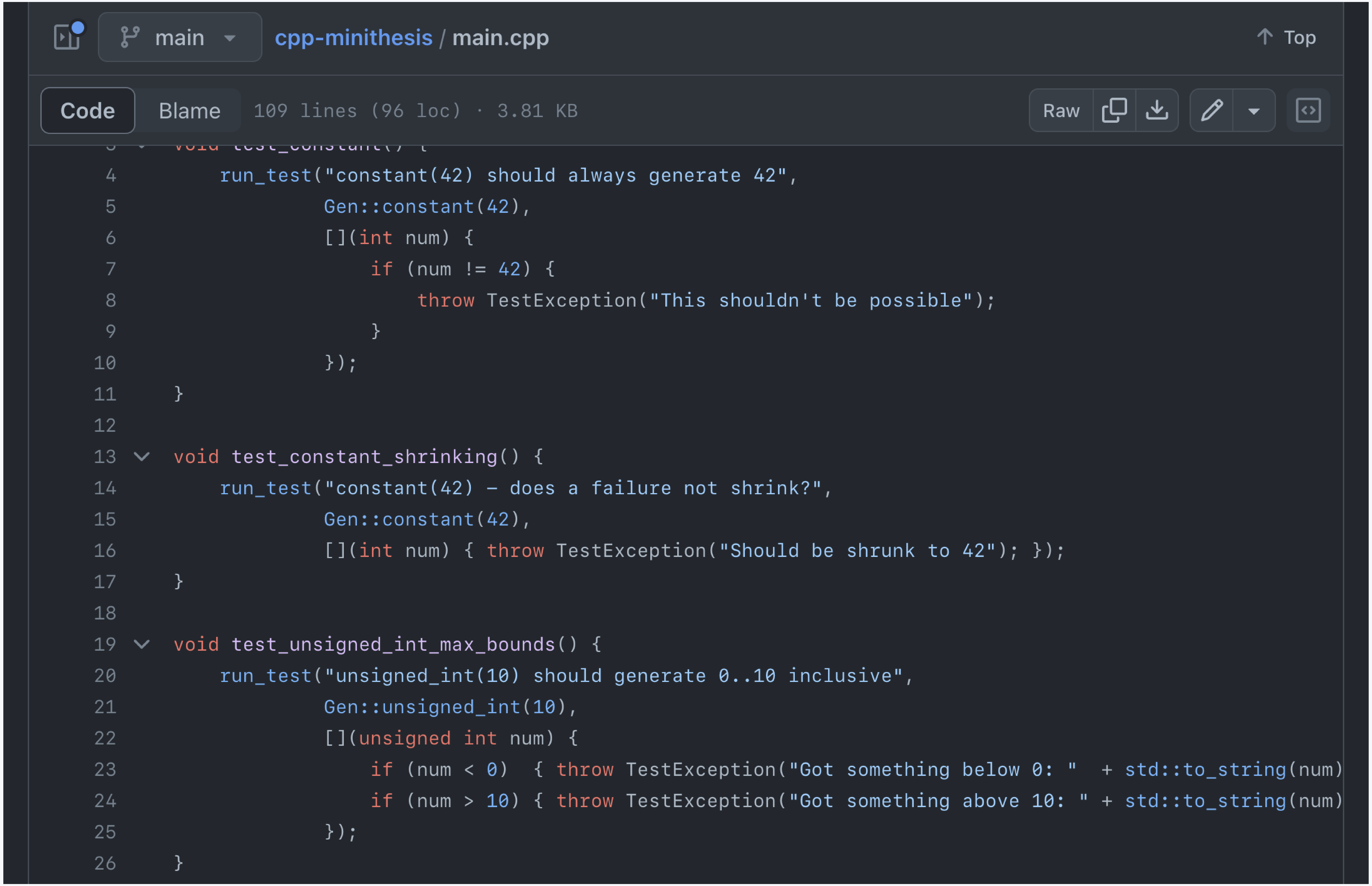
What do you mean?

Why property-based testing?

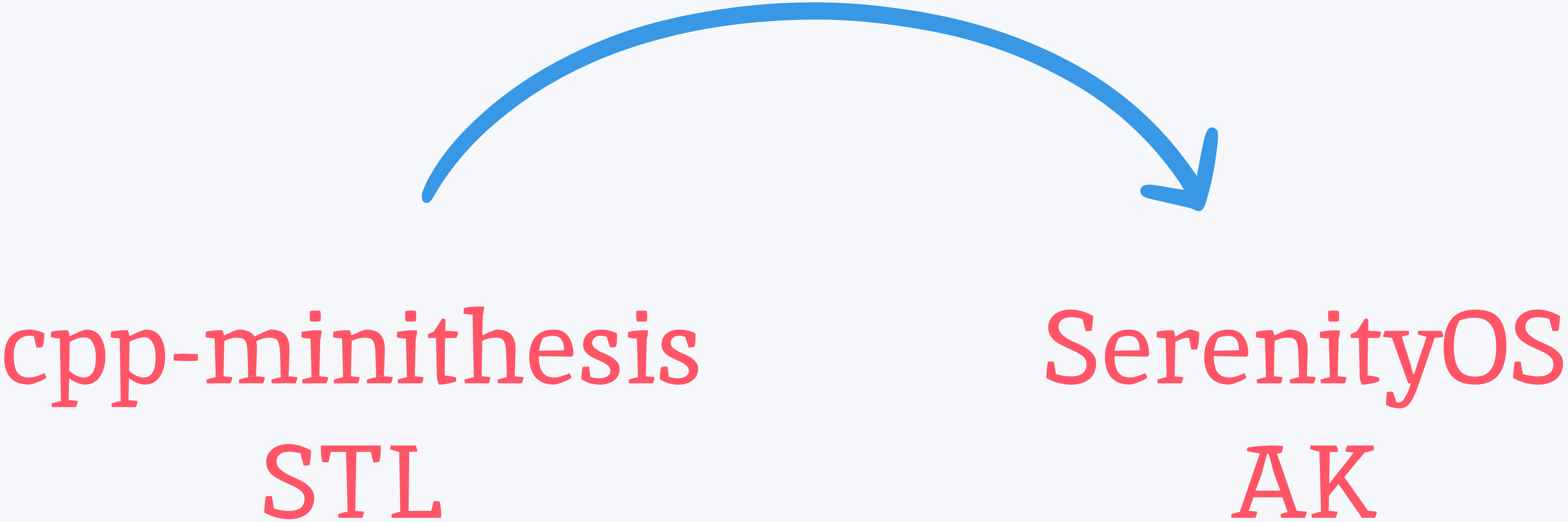
It's great! Tests edge cases you didn't/couldn't think of; increases your confidence that the program works the way you think it does.

Why Minithesis instead of QuickCheck?

It uses an "internal shrinking" approach, which removes the burden of writing shrinkers from the user, and works well in face of monadic bind. This (IMHO) makes it superior to QuickCheck approach (manual/codegen'd shrinkers) and to the "integrated shrinking" lazy rose tree approach (ie. Hedgehog).

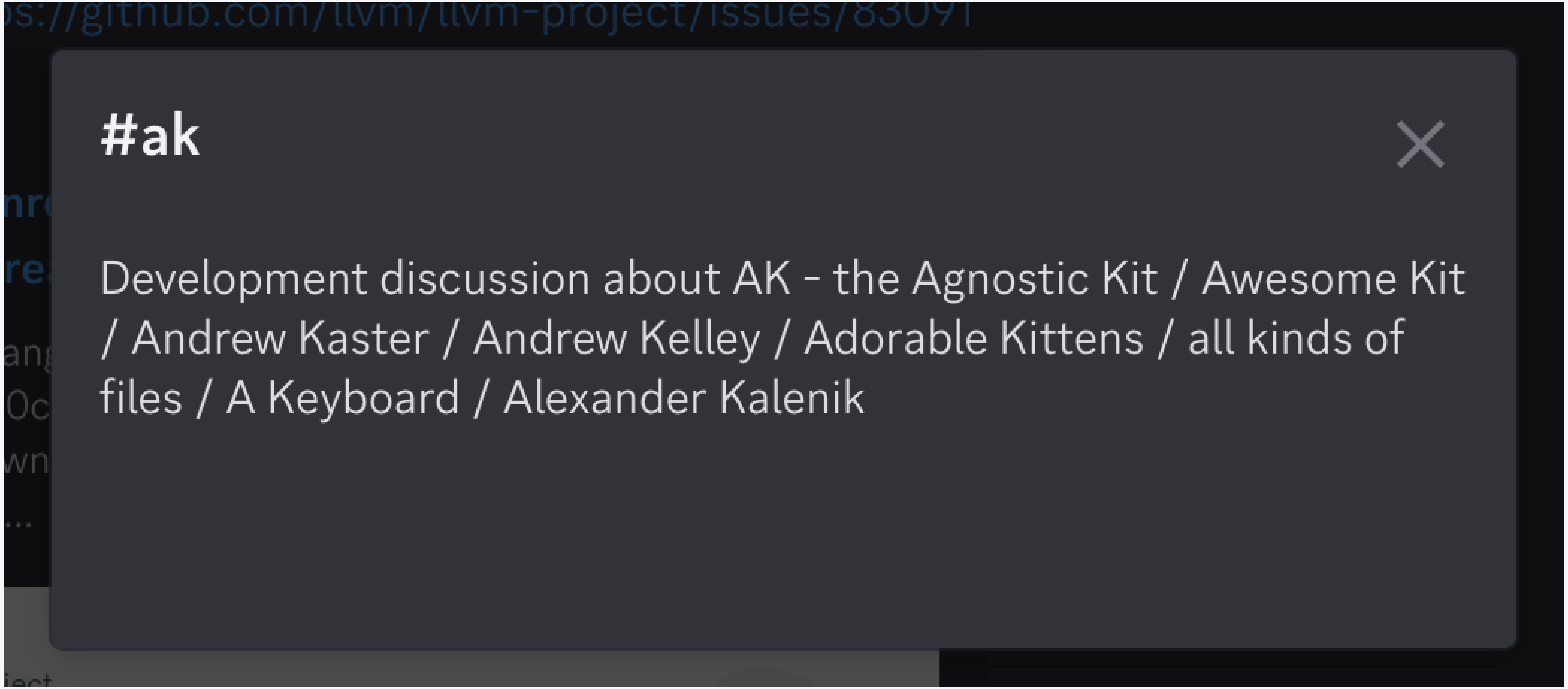


```cpp
void test_constant() {
    run_test("constant(42) should always generate 42",
             Gen::constant(42),
             [](int num) {
                 if (num != 42) {
                     throw TestException("This shouldn't be possible");
                 }
             });
}

void test_constant_shrinking() {
    run_test("constant(42) - does a failure not shrink?",
             Gen::constant(42),
             [](int num) { throw TestException("Should be shrunk to 42"); });
}

void test_unsigned_int_max_bounds() {
    run_test("unsigned_int(10) should generate 0..10 inclusive",
             Gen::unsigned_int(10),
             [](unsigned int num) {
                 if (num < 0)  { throw TestException("Got something below 0: "  + std::to_string(num)
                 if (num > 10) { throw TestException("Got something above 10: " + std::to_string(num)
             });
}
```



cpp-minithesis
STL

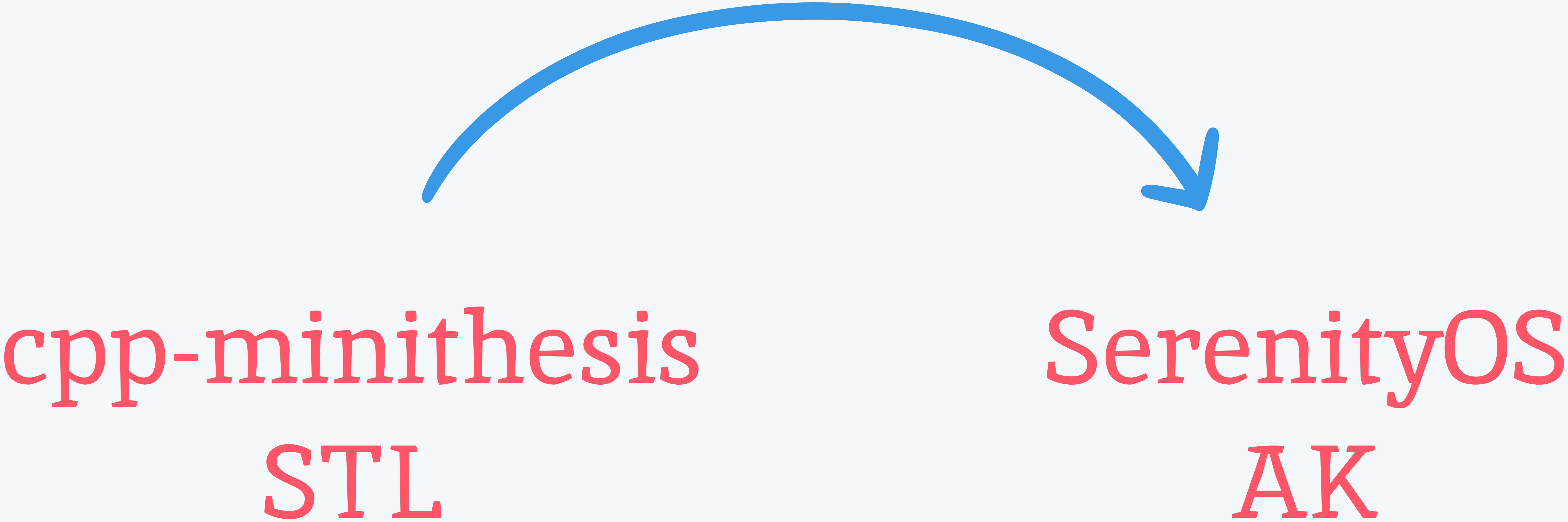
SerenityOS
AK



s://github.com/llvm/llvm-project/issues/83091

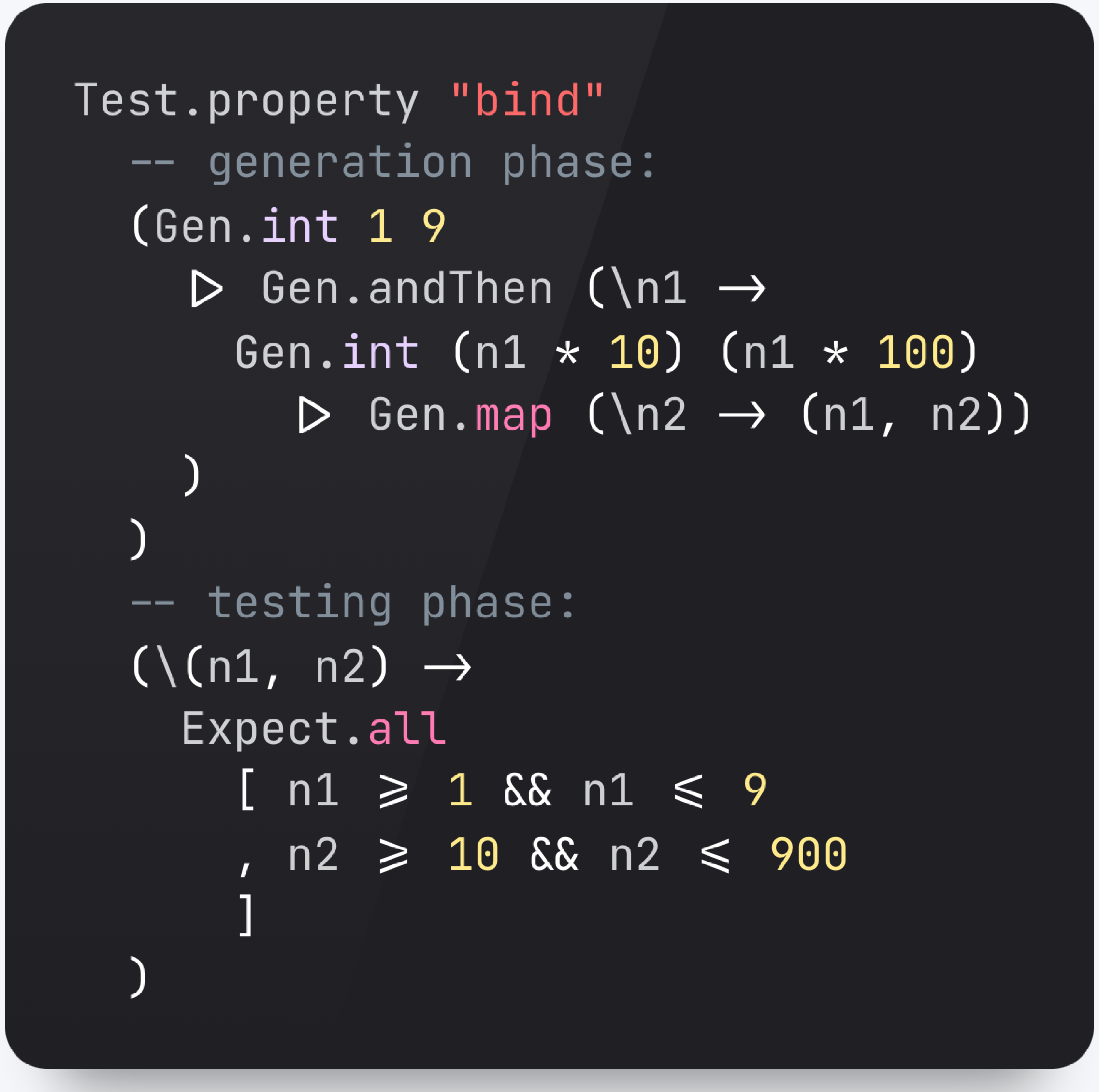
#ak

Development discussion about AK - the Agnostic Kit / Awesome Kit / Andrew Kaster / Andrew Kelley / Adorable Kittens / all kinds of files / A Keyboard / Alexander Kalenik

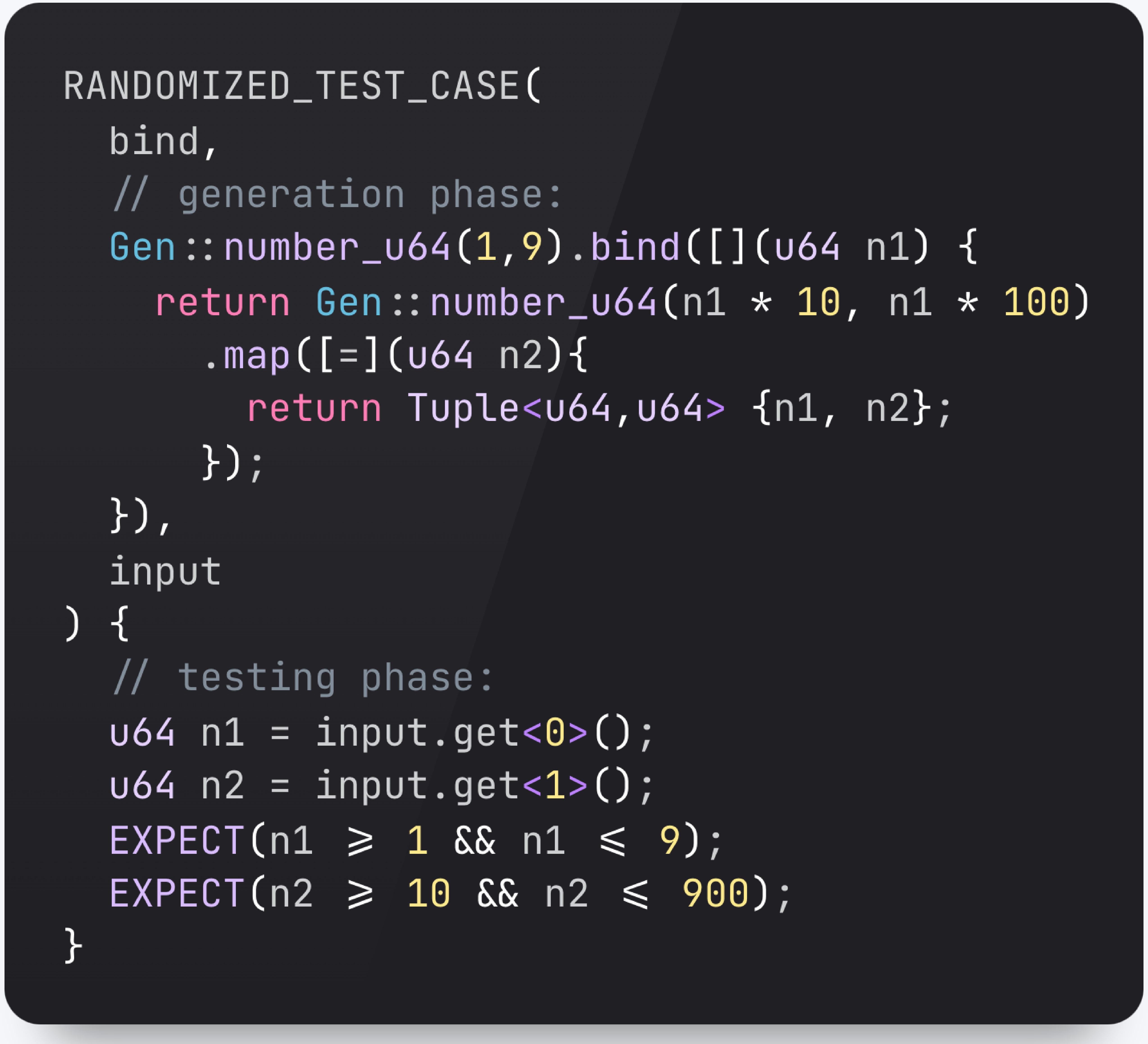


cpp-minithesis
STL

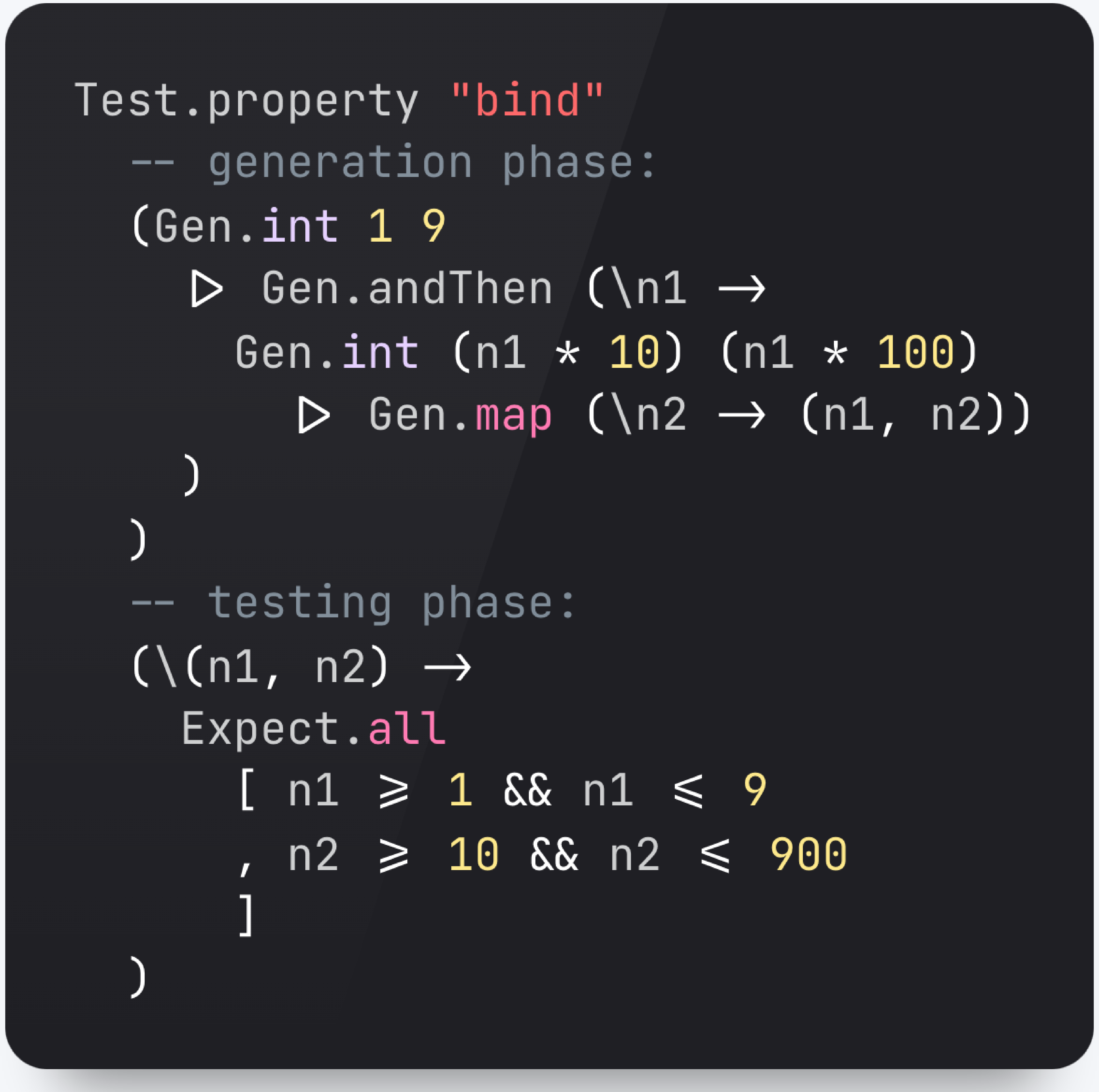
SerenityOS
AK



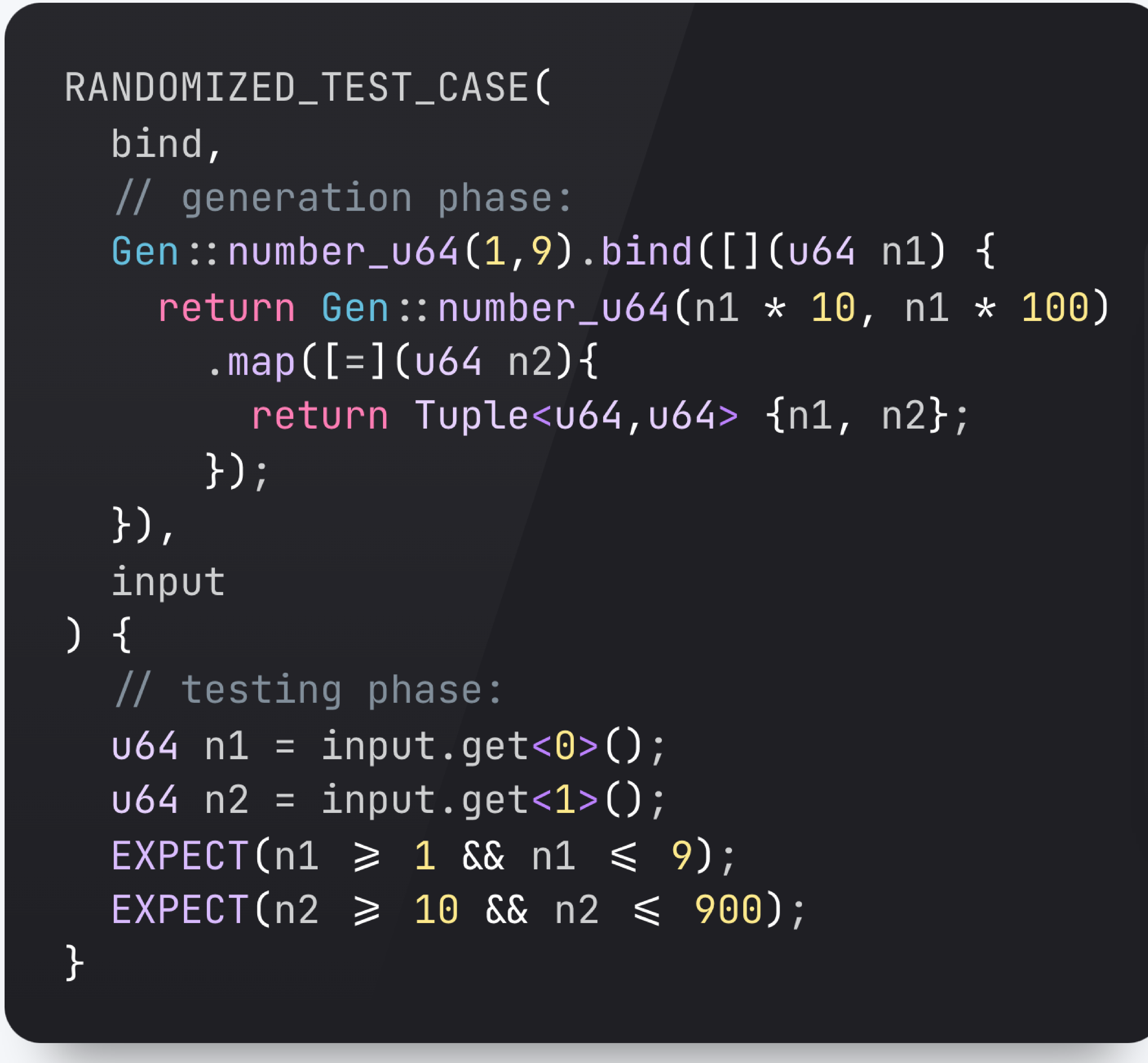
```
Test.property "bind"
  -- generation phase:
  (Gen.int 1 9
    |> Gen.andThen (\n1 ->
      Gen.int (n1 * 10) (n1 * 100)
        |> Gen.map (\n2 -> (n1, n2))
    )
  )
  -- testing phase:
  (\(n1, n2) ->
    Expect.all
      [ n1 >= 1 && n1 <= 9
      , n2 >= 10 && n2 <= 900
      ]
  )
```



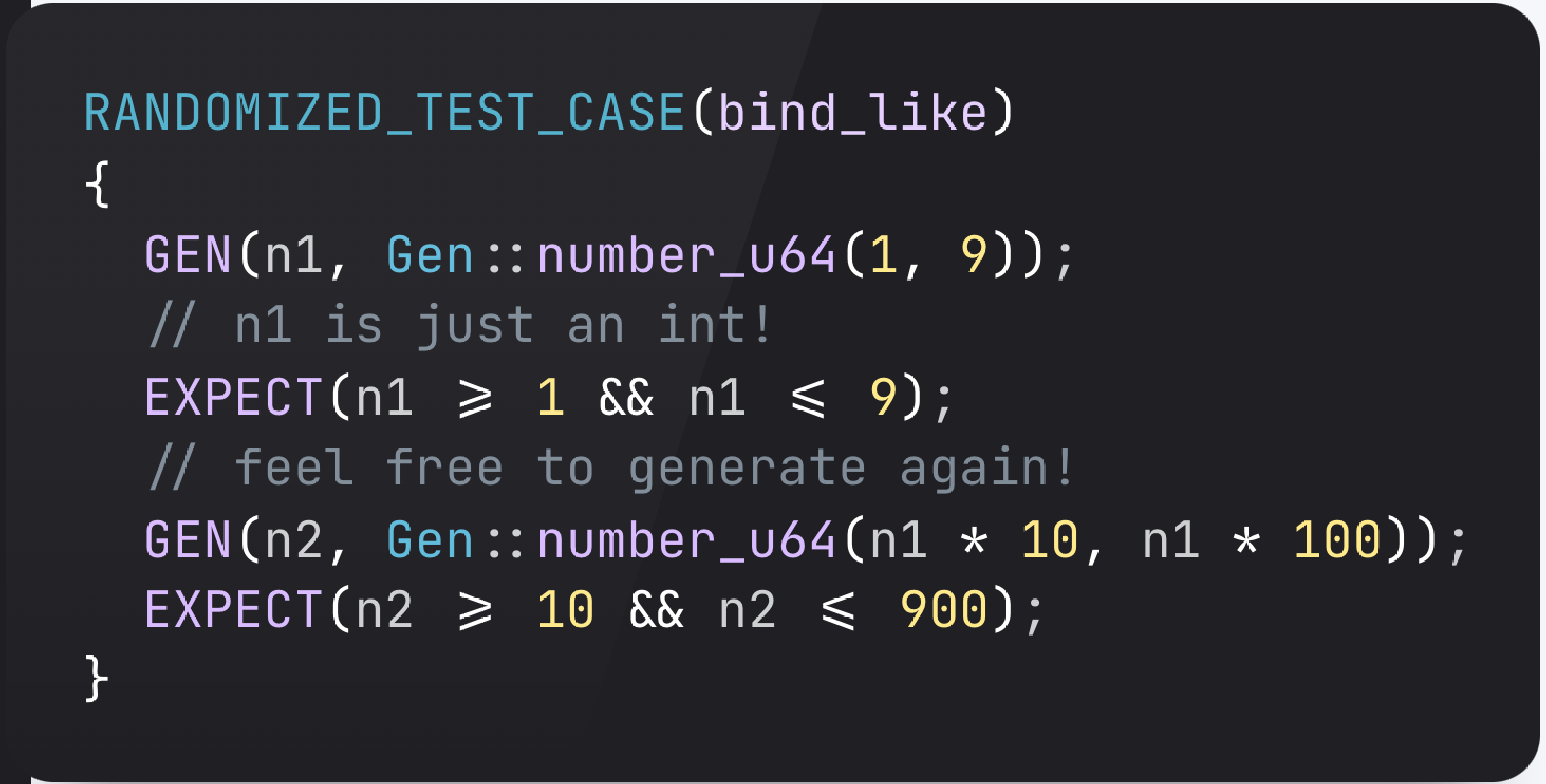
```
RANDOMIZED_TEST_CASE(
  bind,
  // generation phase:
  Gen::number_u64(1,9).bind([](u64 n1) {
    return Gen::number_u64(n1 * 10, n1 * 100)
      .map([=](u64 n2){
        return Tuple<u64,u64> {n1, n2};
      });
  }),
  input
) {
  // testing phase:
  u64 n1 = input.get<0>();
  u64 n2 = input.get<1>();
  EXPECT(n1 ⩾ 1 && n1 ⩽ 9);
  EXPECT(n2 ⩾ 10 && n2 ⩽ 900);
}
```



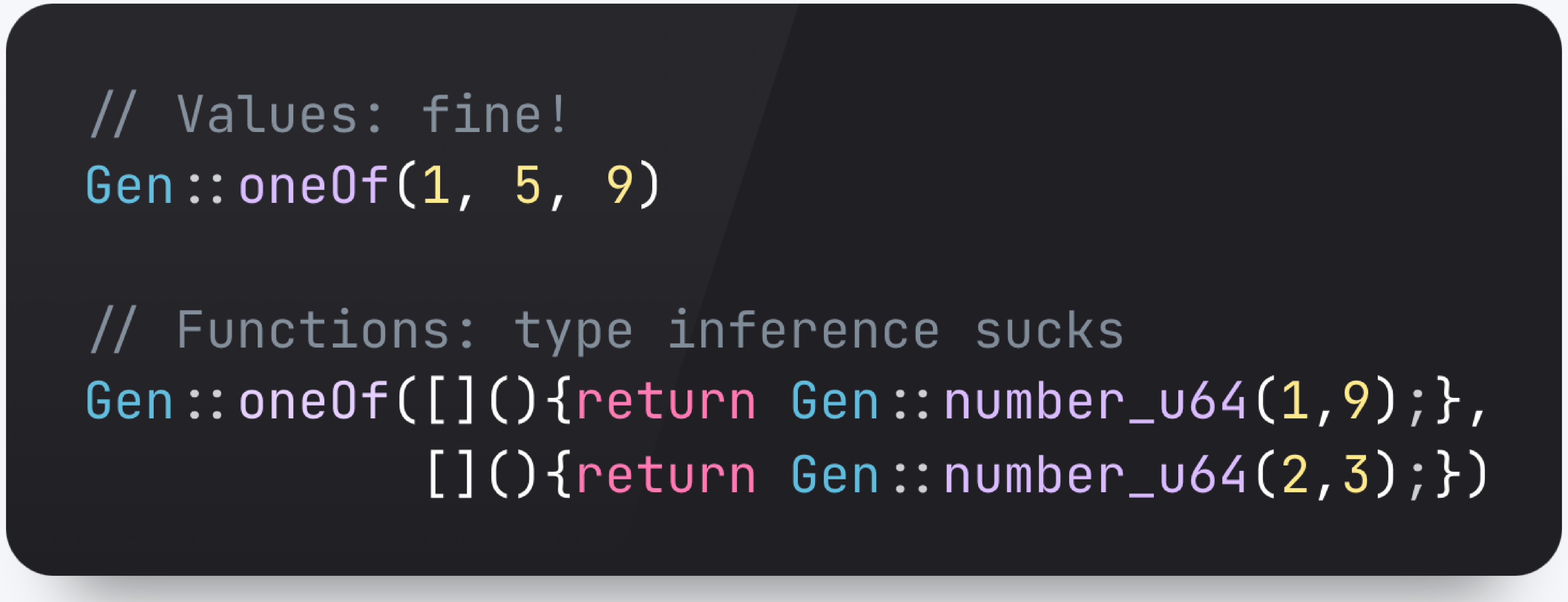
```
Test.property "bind"
  -- generation phase:
  (Gen.int 1 9
    ▷ Gen.andThen (\n1 →
      Gen.int (n1 * 10) (n1 * 100)
        ▷ Gen.map (\n2 → (n1, n2))
    )
  )
  -- testing phase:
  (\(n1, n2) →
    Expect.all
      [ n1 ⩾ 1 && n1 ⩽ 9
      , n2 ⩾ 10 && n2 ⩽ 900
      ]
  )
```



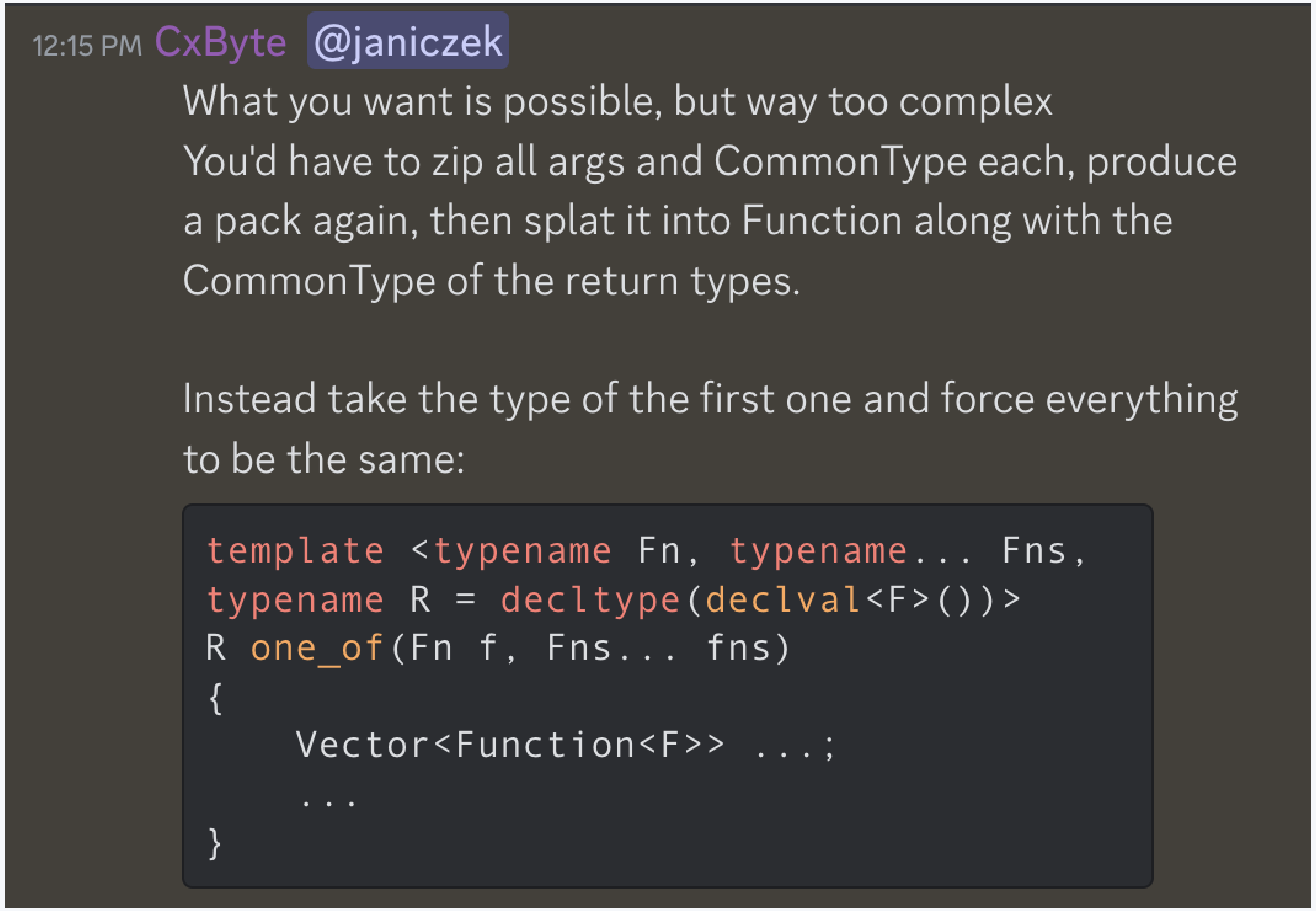
```
RANDOMIZED_TEST_CASE(
  bind,
  // generation phase:
  Gen::number_u64(1,9).bind([](u64 n1) {
    return Gen::number_u64(n1 * 10, n1 * 100)
      .map([=](u64 n2){
        return Tuple<u64,u64> {n1, n2};
      });
  }),
  input
) {
  // testing phase:
  u64 n1 = input.get<0>();
  u64 n2 = input.get<1>();
  EXPECT(n1 >= 1 && n1 <= 9);
  EXPECT(n2 >= 10 && n2 <= 900);
}
```



```
RANDOMIZED_TEST_CASE(bind_like)
{
  GEN(n1, Gen::number_u64(1, 9));
  // n1 is just an int!
  EXPECT(n1 >= 1 && n1 <= 9);
  // feel free to generate again!
  GEN(n2, Gen::number_u64(n1 * 10, n1 * 100));
  EXPECT(n2 >= 10 && n2 <= 900);
}
```



```
// Values: fine!
Gen::oneOf(1, 5, 9)

// Functions: type inference sucks
Gen::oneOf([](){return Gen::number_u64(1,9);},
           [](){return Gen::number_u64(2,3);})
```

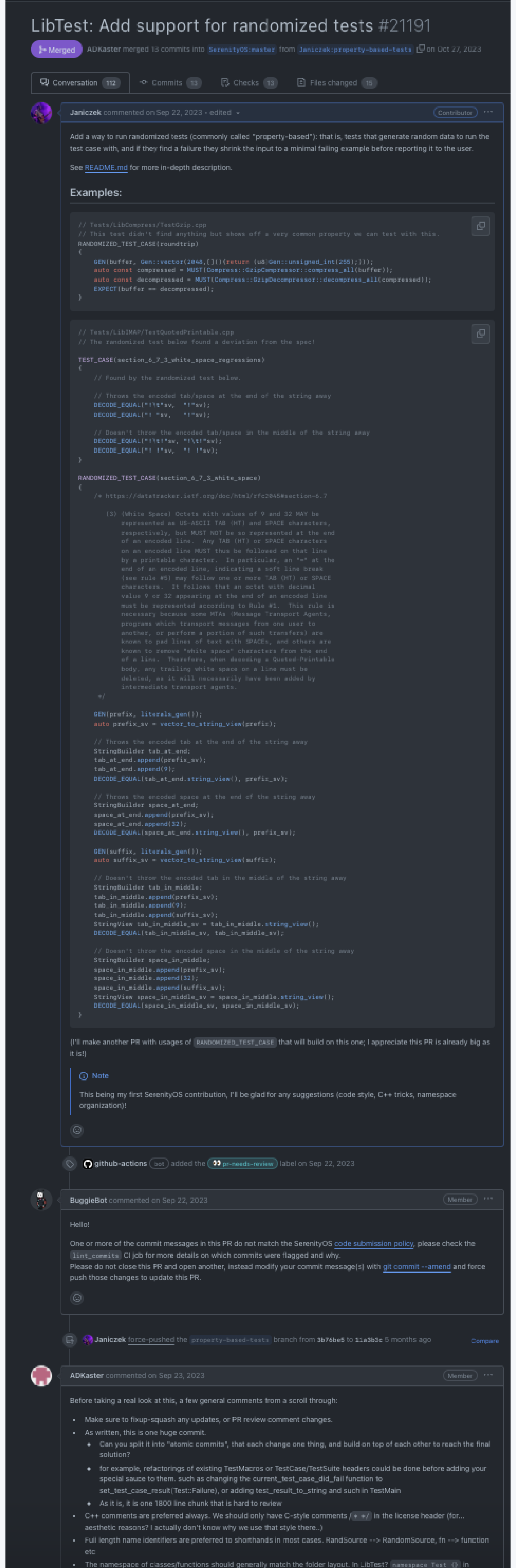


12:15 PM CxByte @janiczek

What you want is possible, but way too complex
You'd have to zip all args and CommonType each, produce a pack again, then splat it into Function along with the CommonType of the return types.

Instead take the type of the first one and force everything to be the same:

```
template <typename Fn, typename... Fns,
typename R = decltype(declval<F>())>
R one_of(Fn f, Fns... fns)
{
    Vector<Function<F>> ...;
    ...
}
```



LibTest: Add support for randomized tests #21191

Merged ADKaster merged 13 commits into `SerenityOS:master` from `Janiczek:property-based-tests` on Oct 27, 2023

Janiczek commented on Sep 22, 2023 • edited

Add a way to run randomized tests (commonly called "property-based"): that is, tests that generate random data to run the test case with, and if they find a failure they shrink the input to a minimal failing example before reporting it to the user.

See README.md for more in-depth description.

Examples:

```
// Tests/LibCompress/TestGzip.cpp
// This test didn't find anything but shows off a very common property we can test with this.
RANDOMIZED_TEST_CASE(roundtrip)
{
    GEN(buffer, Gen::vector(2048,[](){return (u8)Gen::unsigned_int(255);}));
    auto const compressed = MUST(Compress::GzipCompressor::compress_all(buffer));
    auto const decompressed = MUST(Compress::GzipDecompressor::decompress_all(compressed));
    EXPECT(buffer == decompressed);
}
```

```
// Tests/LibIMAP/TestQuotedPrintable.cpp
// The randomized test below found a deviation from the spec!

TEST_CASE(section_6_7_3_white_space_regressions)
{
    // Found by the randomized test below.

    // Throws the encoded tab/space at the end of the string away
    DECODE_EQUAL("!\t"sv,  "!"sv);
    DECODE_EQUAL("! "sv,   "!"sv);

    // Doesn't throw the encoded tab/space in the middle of the string away
    DECODE_EQUAL("!\t!"sv, "!\t!"sv);
    DECODE_EQUAL("! !"sv,  "! !"sv);
}

RANDOMIZED_TEST_CASE(section_6_7_3_white_space)
{
    /* https://datatracker.ietf.org/doc/html/rfc2045#section-6.7

       (3) (White Space) Octets with values of 9 and 32 MAY be
           represented as US-ASCII TAB (HT) and SPACE characters,
           respectively, but MUST NOT be so represented at the end
           of an encoded line.  Any TAB (HT) or SPACE characters
           on an encoded line MUST thus be followed on that line
           by a printable character.  In particular, an "=" at the
           end of an encoded line, indicating a soft line break
           (see rule #5) may follow one or more TAB (HT) or SPACE
           characters.  It follows that an octet with decimal
           value 9 or 32 appearing at the end of an encoded line
           must be represented according to Rule #1.  This rule is
           necessary because some MTAs (Message Transport Agents,
           programs which transport messages from one user to
           another, or perform a portion of such transfers) are
           known to pad lines of text with SPACEs, and others are
           known to remove "white space" characters from the end
           of a line.  Therefore, when decoding a Quoted-Printable
           body, any trailing white space on a line must be
           deleted, as it will necessarily have been added by
           intermediate transport agents.
     */

    GEN(prefix, literals_gen());
    auto prefix_sv = vector_to_string_view(prefix);

    // Throws the encoded tab at the end of the string away
    StringBuilder tab_at_end;
    tab_at_end.append(prefix_sv);
    tab_at_end.append(9);
    DECODE_EQUAL(tab_at_end.string_view(), prefix_sv);

    // Throws the encoded space at the end of the string away
    StringBuilder space_at_end;
    space_at_end.append(prefix_sv);
    space_at_end.append(32);
    DECODE_EQUAL(space_at_end.string_view(), prefix_sv);

    GEN(suffix, literals_gen());
    auto suffix_sv = vector_to_string_view(suffix);

    // Doesn't throw the encoded tab in the middle of the string away
    StringBuilder tab_in_middle;
    tab_in_middle.append(prefix_sv);
    tab_in_middle.append(9);
    tab_in_middle.append(suffix_sv);
    StringView tab_in_middle_sv = tab_in_middle.string_view();
    DECODE_EQUAL(tab_in_middle_sv, tab_in_middle_sv);

    // Doesn't throw the encoded space in the middle of the string away
    StringBuilder space_in_middle;
    space_in_middle.append(prefix_sv);
    space_in_middle.append(32);
    space_in_middle.append(suffix_sv);
    StringView space_in_middle_sv = space_in_middle.string_view();
    DECODE_EQUAL(space_in_middle_sv, space_in_middle_sv);
}
```

(I'll make another PR with usages of `RANDOMIZED_TEST_CASE` that will build on this one; I appreciate this PR is already big as it is!)

> ⓘ Note
>
> This being my first SerenityOS contribution, I'll be glad for any suggestions (code style, C++ tricks, namespace organization)!

github-actions bot added the 👀 `pr-needs-review` label on Sep 22, 2023

BuggieBot commented on Sep 22, 2023

Hello!

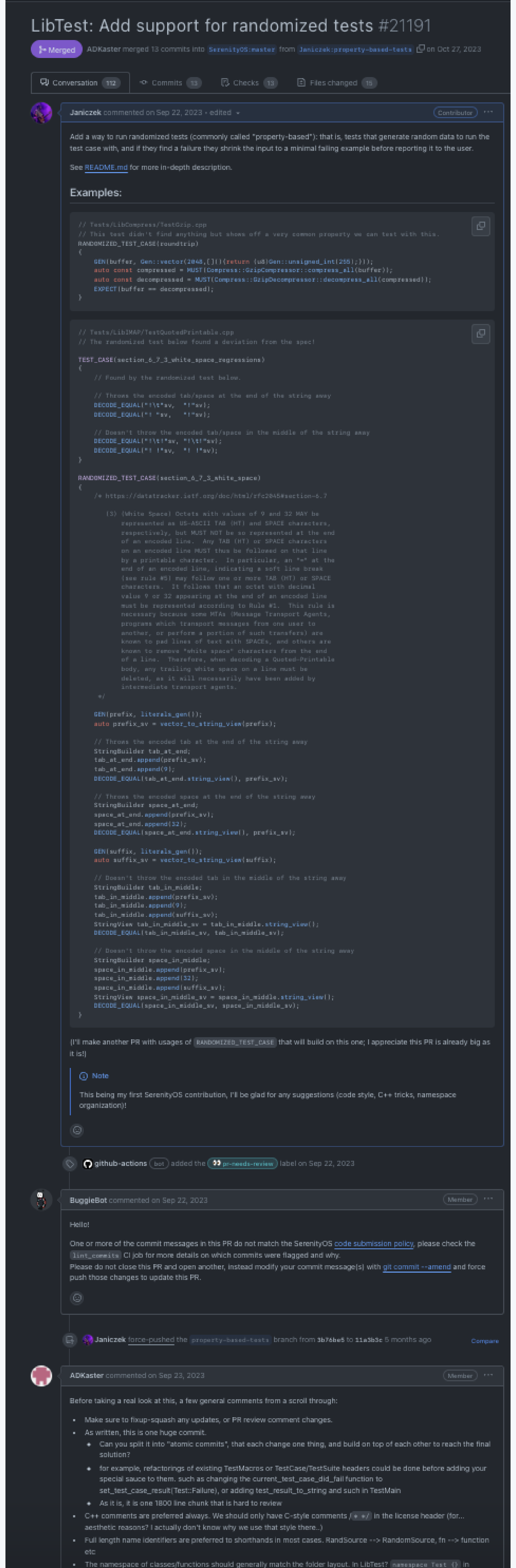
One or more of the commit messages in this PR do not match the SerenityOS code submission policy, please check the `lint_commits` CI job for more details on which commits were flagged and why.
Please do not close this PR and open another, instead modify your commit message(s) with git commit --amend and force push those changes to update this PR.

Janiczek force-pushed the `property-based-tests` branch from **3b76be5** to **11a3b3c** 5 months ago

ADKaster commented on Sep 23, 2023

Before taking a real look at this, a few general comments from a scroll through:

- Make sure to fixup-squash any updates, or PR review comment changes.
- As written, this is one huge commit.
  - Can you split it into "atomic commits", that each change one thing, and build on top of each other to reach the final solution?
  - for example, refactorings of existing TestMacros or TestCase/TestSuite headers could be done before adding your special sauce to them. such as changing the current_test_case_did_fail function to set_test_case_result(Test::Failure), or adding test_result_to_string and such in TestMain
  - As it is, it is one 1800 line chunk that is hard to review
- C++ comments are preferred always. We should only have C-style comments `/* */` in the license header (for... aesthetic reasons? I actually don't know why we use that style there..)
- Full length name identifiers are preferred to shorthands in most cases. RandSource --> RandomSource, fn --> function etc
- The namespace of classes/functions should generally match the folder layout. In LibTest? `namespace Test {}` in



LibTest: Add support for randomized tests #21191

Merged ADKaster merged 13 commits into `SerenityOS:master` from `Janiczek:property-based-tests` on Oct 27, 2023

Janiczek commented on Sep 22, 2023 • edited — Contributor

Add a way to run randomized tests (commonly called "property-based"): that is, tests that generate random data to run the test case with, and if they find a failure they shrink the input to a minimal failing example before reporting it to the user.

See README.md for more in-depth description.

Examples:

```
// Tests/LibCompress/TestGzip.cpp
// This test didn't find anything but shows off a very common property we can test with this.
RANDOMIZED_TEST_CASE(roundtrip)
{
    GEN(buffer, Gen::vector(2048,[](){return (u8)Gen::unsigned_int(255);}));
    auto const compressed = MUST(Compress::GzipCompressor::compress_all(buffer));
    auto const decompressed = MUST(Compress::GzipDecompressor::decompress_all(compressed));
    EXPECT(buffer == decompressed);
}
```

```
// Tests/LibIMAP/TestQuotedPrintable.cpp
// The randomized test below found a deviation from the spec!

TEST_CASE(section_6_7_3_white_space_regressions)
{
    // Found by the randomized test below.

    // Throws the encoded tab/space at the end of the string away
    DECODE_EQUAL("!\t"sv,  "!"sv);
    DECODE_EQUAL("! "sv,   "!"sv);

    // Doesn't throw the encoded tab/space in the middle of the string away
    DECODE_EQUAL("!\t!"sv, "!\t!"sv);
    DECODE_EQUAL("! !"sv,  "! !"sv);
}

RANDOMIZED_TEST_CASE(section_6_7_3_white_space)
{
    /* https://datatracker.ietf.org/doc/html/rfc2045#section-6.7

       (3) (White Space) Octets with values of 9 and 32 MAY be
           represented as US-ASCII TAB (HT) and SPACE characters,
           respectively, but MUST NOT be so represented at the end
           of an encoded line.  Any TAB (HT) or SPACE characters
           on an encoded line MUST thus be followed on that line
           by a printable character.  In particular, an "=" at the
           end of an encoded line, indicating a soft line break
           (see rule #5) may follow one or more TAB (HT) or SPACE
           characters.  It follows that an octet with decimal
           value 9 or 32 appearing at the end of an encoded line
           must be represented according to Rule #1.  This rule is
           necessary because some MTAs (Message Transport Agents,
           programs which transport messages from one user to
           another, or perform a portion of such transfers) are
           known to pad lines of text with SPACEs, and others are
           known to remove "white space" characters from the end
           of a line.  Therefore, when decoding a Quoted-Printable
           body, any trailing white space on a line must be
           deleted, as it will necessarily have been added by
           intermediate transport agents.
     */

    GEN(prefix, literals_gen());
    auto prefix_sv = vector_to_string_view(prefix);

    // Throws the encoded tab at the end of the string away
    StringBuilder tab_at_end;
    tab_at_end.append(prefix_sv);
    tab_at_end.append(9);
    DECODE_EQUAL(tab_at_end.string_view(), prefix_sv);

    // Throws the encoded space at the end of the string away
    StringBuilder space_at_end;
    space_at_end.append(prefix_sv);
    space_at_end.append(32);
    DECODE_EQUAL(space_at_end.string_view(), prefix_sv);

    GEN(suffix, literals_gen());
    auto suffix_sv = vector_to_string_view(suffix);

    // Doesn't throw the encoded tab in the middle of the string away
    StringBuilder tab_in_middle;
    tab_in_middle.append(prefix_sv);
    tab_in_middle.append(9);
    tab_in_middle.append(suffix_sv);
    StringView tab_in_middle_sv = tab_in_middle.string_view();
    DECODE_EQUAL(tab_in_middle_sv, tab_in_middle_sv);

    // Doesn't throw the encoded space in the middle of the string away
    StringBuilder space_in_middle;
    space_in_middle.append(prefix_sv);
    space_in_middle.append(32);
    space_in_middle.append(suffix_sv);
    StringView space_in_middle_sv = space_in_middle.string_view();
    DECODE_EQUAL(space_in_middle_sv, space_in_middle_sv);
}
```

(I'll make another PR with usages of `RANDOMIZED_TEST_CASE` that will build on this one; I appreciate this PR is already big as it is!)

> ⓘ Note
>
> This being my first SerenityOS contribution, I'll be glad for any suggestions (code style, C++ tricks, namespace organization)!

github-actions bot added the pr-needs-review label on Sep 22, 2023

BuggieBot commented on Sep 22, 2023 — Member

Hello!

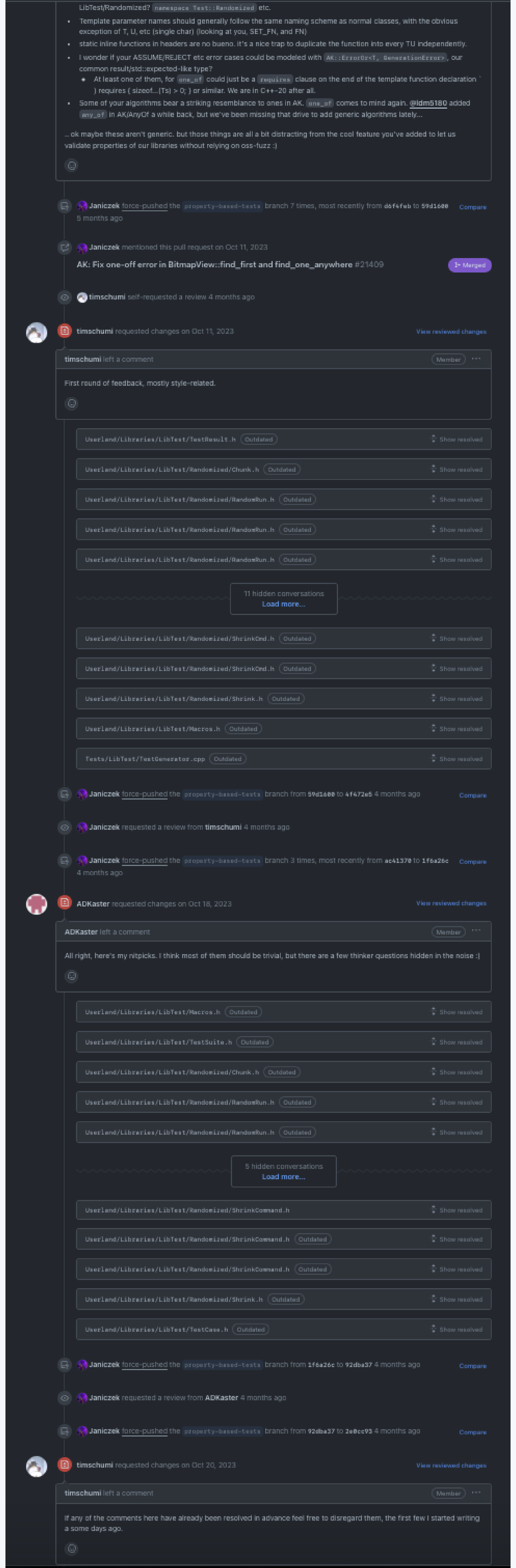
One or more of the commit messages in this PR do not match the SerenityOS code submission policy, please check the `lint_commits` CI job for more details on which commits were flagged and why.
Please do not close this PR and open another, instead modify your commit message(s) with git commit --amend and force push those changes to update this PR.

Janiczek force-pushed the `property-based-tests` branch from **3b76be5** to **11a3b3c** 5 months ago — Compare

ADKaster commented on Sep 23, 2023 — Member

Before taking a real look at this, a few general comments from a scroll through:

- Make sure to fixup-squash any updates, or PR review comment changes.
- As written, this is one huge commit.
  - Can you split it into "atomic commits", that each change one thing, and build on top of each other to reach the final solution?
  - for example, refactorings of existing TestMacros or TestCase/TestSuite headers could be done before adding your special sauce to them. such as changing the current_test_case_did_fail function to set_test_case_result(Test::Failure), or adding test_result_to_string and such in TestMain
  - As it is, it is one 1800 line chunk that is hard to review
- C++ comments are preferred always. We should only have C-style comments `/* */` in the license header (for... aesthetic reasons? I actually don't know why we use that style there..)
- Full length name identifiers are preferred to shorthands in most cases. RandSource --> RandomSource, fn --> function etc
- The namespace of classes/functions should generally match the folder layout. In LibTest? `namespace Test {}` in



LibTest/Randomized? `namespace Test::Randomized` etc.
- Template parameter names should generally follow the same naming scheme as normal classes, with the obvious exception of T, U, etc (single char) (looking at you, SET_FN, and FN)
- static inline functions in headers are no bueno. it's a nice trap to duplicate the function into every TU independently.
- I wonder if your ASSUME/REJECT etc error cases could be modeled with `AK::ErrorOr<T, GenerationError>`, our common result/std::expected-like type?
  - At least one of them, for `one_of` could just be a `requires` clause on the end of the template function declaration `) requires (sizeof...(Ts) > 0;) or similar. We are in C++-20 after all.
- Some of your algorithms bear a striking resemblance to ones in AK. `one_of` comes to mind again. @ldm5180 added `any_of` in AK/AnyOf a while back, but we've been missing that drive to add generic algorithms lately...

... ok maybe these aren't generic. but those things are all a bit distracting from the cool feature you've added to let us validate properties of our libraries without relying on oss-fuzz :)

Janiczek force-pushed the `property-based-tests` branch 7 times, most recently from **d6f4feb** to **59d1600** 5 months ago — Compare

Janiczek mentioned this pull request on Oct 11, 2023

AK: Fix one-off error in BitmapView::find_first and find_one_anywhere #21409 — Merged

timschumi self-requested a review 4 months ago

timschumi requested changes on Oct 11, 2023 — View reviewed changes

timschumi left a comment — Member

First round of feedback, mostly style-related.

```
Userland/Libraries/LibTest/TestResult.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/Chunk.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/RandomRun.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/RandomRun.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/RandomRun.h  Outdated  Show resolved
```

11 hidden conversations
Load more...

```
Userland/Libraries/LibTest/Randomized/ShrinkCmd.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/ShrinkCmd.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/Shrink.h  Outdated  Show resolved
Userland/Libraries/LibTest/Macros.h  Outdated  Show resolved
Tests/LibTest/TestGenerator.cpp  Outdated  Show resolved
```

Janiczek force-pushed the `property-based-tests` branch from **59d1600** to **4f472e5** 4 months ago — Compare

Janiczek requested a review from **timschumi** 4 months ago

Janiczek force-pushed the `property-based-tests` branch 3 times, most recently from **ac41370** to **1f6a26c** 4 months ago — Compare

ADKaster requested changes on Oct 18, 2023 — View reviewed changes

ADKaster left a comment — Member

All right, here's my nitpicks. I think most of them should be trivial, but there are a few thinker questions hidden in the noise :)

```
Userland/Libraries/LibTest/Macros.h  Outdated  Show resolved
Userland/Libraries/LibTest/TestSuite.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/Chunk.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/RandomRun.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/RandomRun.h  Outdated  Show resolved
```

5 hidden conversations
Load more...

```
Userland/Libraries/LibTest/Randomized/ShrinkCommand.h  Show resolved
Userland/Libraries/LibTest/Randomized/ShrinkCommand.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/ShrinkCommand.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/Shrink.h  Outdated  Show resolved
Userland/Libraries/LibTest/TestCase.h  Outdated  Show resolved
```

Janiczek force-pushed the `property-based-tests` branch from **1f6a26c** to **92dba37** 4 months ago — Compare

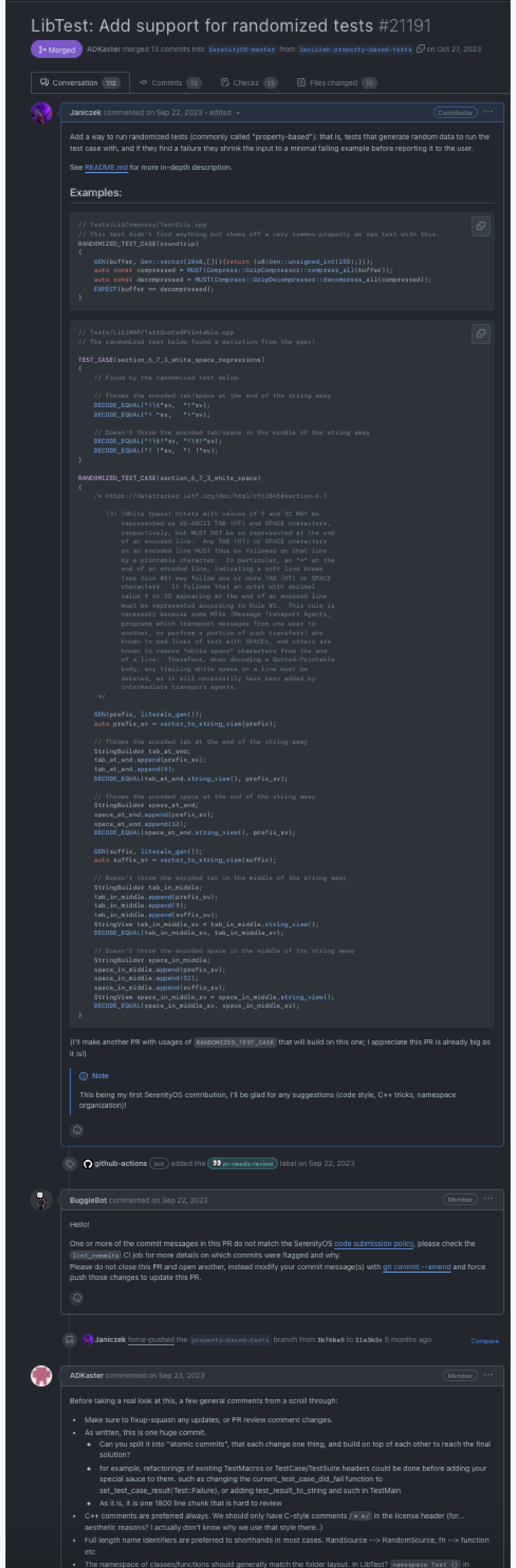
Janiczek requested a review from **ADKaster** 4 months ago

Janiczek force-pushed the `property-based-tests` branch from **92dba37** to **2e8cc93** 4 months ago — Compare

timschumi requested changes on Oct 20, 2023 — View reviewed changes

timschumi left a comment — Member

If any of the comments here have already been resolved in advance feel free to disregard them, the first few I started writing a some days ago.



LibTest: Add support for randomized tests #21191

Merged ADKaster merged 13 commits into `SerenityOS:master` from `Janiczek:property-based-tests` on Oct 27, 2023

Janiczek commented on Sep 22, 2023 • edited — Contributor

Add a way to run randomized tests (commonly called "property-based"): that is, tests that generate random data to run the test case with, and if they find a failure they shrink the input to a minimal failing example before reporting it to the user.

See README.md for more in-depth description.

Examples:

```
// Tests/LibCompress/TestGzip.cpp
// This test didn't find anything but shows off a very common property we can test with this.
RANDOMIZED_TEST_CASE(roundtrip)
{
    GEN(buffer, Gen::vector(2048,[](){return (u8)Gen::unsigned_int(255);}));
    auto const compressed = MUST(Compress::GzipCompressor::compress_all(buffer));
    auto const decompressed = MUST(Compress::GzipDecompressor::decompress_all(compressed));
    EXPECT(buffer == decompressed);
}
```

```
// Tests/LibIMAP/TestQuotedPrintable.cpp
// The randomized test below found a deviation from the spec!

TEST_CASE(section_6_7_3_white_space_regressions)
{
    // Found by the randomized test below.

    // Throws the encoded tab/space at the end of the string away
    DECODE_EQUAL("!\t"sv,  "!"sv);
    DECODE_EQUAL("! "sv,   "!"sv);

    // Doesn't throw the encoded tab/space in the middle of the string away
    DECODE_EQUAL("!\t!"sv, "!\t!"sv);
    DECODE_EQUAL("! !"sv,  "! !"sv);
}

RANDOMIZED_TEST_CASE(section_6_7_3_white_space)
{
    /* https://datatracker.ietf.org/doc/html/rfc2045#section-6.7

       (3) (White Space) Octets with values of 9 and 32 MAY be
           represented as US-ASCII TAB (HT) and SPACE characters,
           respectively, but MUST NOT be so represented at the end
           of an encoded line.  Any TAB (HT) or SPACE characters
           on an encoded line MUST thus be followed on that line
           by a printable character.  In particular, an "=" at the
           end of an encoded line, indicating a soft line break
           (see rule #5) may follow one or more TAB (HT) or SPACE
           characters.  It follows that an octet with decimal
           value 9 or 32 appearing at the end of an encoded line
           must be represented according to Rule #1.  This rule is
           necessary because some MTAs (Message Transport Agents,
           programs which transport messages from one user to
           another, or perform a portion of such transfers) are
           known to pad lines of text with SPACEs, and others are
           known to remove "white space" characters from the end
           of a line.  Therefore, when decoding a Quoted-Printable
           body, any trailing white space on a line must be
           deleted, as it will necessarily have been added by
           intermediate transport agents.
     */

    GEN(prefix, literals_gen());
    auto prefix_sv = vector_to_string_view(prefix);

    // Throws the encoded tab at the end of the string away
    StringBuilder tab_at_end;
    tab_at_end.append(prefix_sv);
    tab_at_end.append(9);
    DECODE_EQUAL(tab_at_end.string_view(), prefix_sv);

    // Throws the encoded space at the end of the string away
    StringBuilder space_at_end;
    space_at_end.append(prefix_sv);
    space_at_end.append(32);
    DECODE_EQUAL(space_at_end.string_view(), prefix_sv);

    GEN(suffix, literals_gen());
    auto suffix_sv = vector_to_string_view(suffix);

    // Doesn't throw the encoded tab in the middle of the string away
    StringBuilder tab_in_middle;
    tab_in_middle.append(prefix_sv);
    tab_in_middle.append(9);
    tab_in_middle.append(suffix_sv);
    StringView tab_in_middle_sv = tab_in_middle.string_view();
    DECODE_EQUAL(tab_in_middle_sv, tab_in_middle_sv);

    // Doesn't throw the encoded space in the middle of the string away
    StringBuilder space_in_middle;
    space_in_middle.append(prefix_sv);
    space_in_middle.append(32);
    space_in_middle.append(suffix_sv);
    StringView space_in_middle_sv = space_in_middle.string_view();
    DECODE_EQUAL(space_in_middle_sv, space_in_middle_sv);
}
```

(I'll make another PR with usages of `RANDOMIZED_TEST_CASE` that will build on this one; I appreciate this PR is already big as it is!)

> ⓘ Note
>
> This being my first SerenityOS contribution, I'll be glad for any suggestions (code style, C++ tricks, namespace organization)!

github-actions bot added the 👀 pr-needs-review label on Sep 22, 2023

BuggieBot commented on Sep 22, 2023 — Member

Hello!

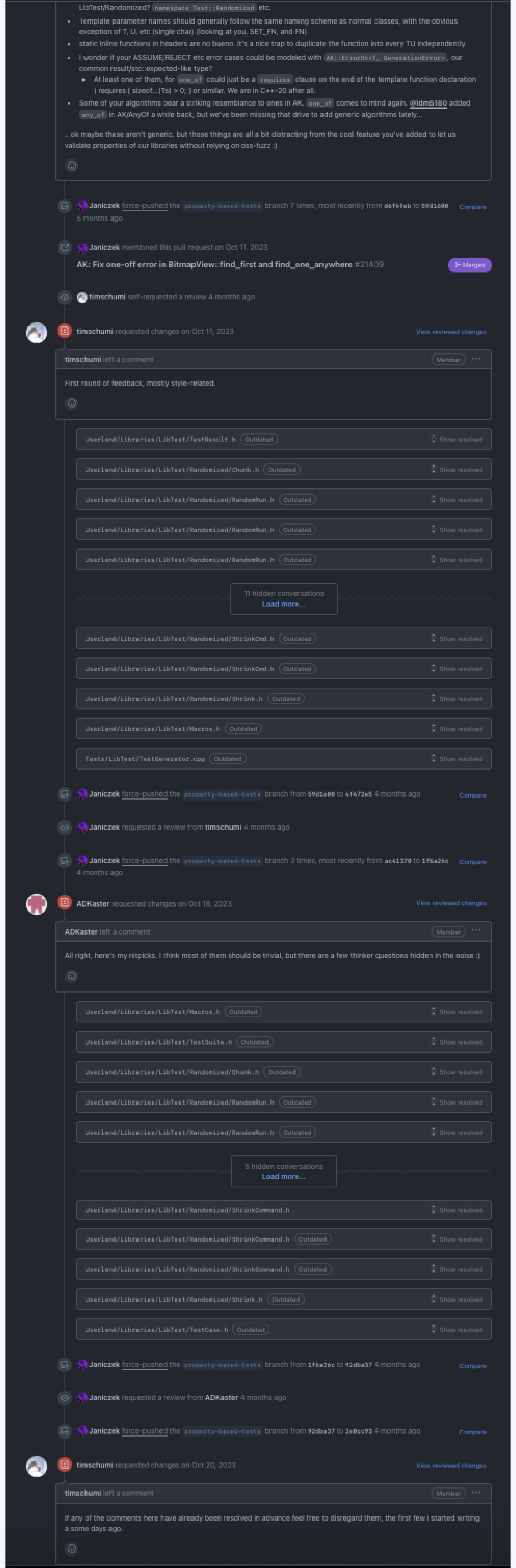
One or more of the commit messages in this PR do not match the SerenityOS code submission policy, please check the `lint_commits` CI job for more details on which commits were flagged and why.
Please do not close this PR and open another, instead modify your commit message(s) with git commit --amend and force push those changes to update this PR.

Janiczek force-pushed the `property-based-tests` branch from `3b76be5` to `11a3b3c` 5 months ago — Compare

ADKaster commented on Sep 23, 2023 — Member

Before taking a real look at this, a few general comments from a scroll through:

- Make sure to fixup-squash any updates, or PR review comment changes.
- As written, this is one huge commit.
  - Can you split it into "atomic commits", that each change one thing, and build on top of each other to reach the final solution?
  - for example, refactorings of existing TestMacros or TestCase/TestSuite headers could be done before adding your special sauce to them. such as changing the current_test_case_did_fail function to set_test_case_result(Test::Failure), or adding test_result_to_string and such in TestMain
  - As it is, it is one 1800 line chunk that is hard to review
- C++ comments are preferred always. We should only have C-style comments `/* */` in the license header (for... aesthetic reasons? I actually don't know why we use that style there..)
- Full length name identifiers are preferred to shorthands in most cases. RandSource --> RandomSource, fn --> function etc
- The namespace of classes/functions should generally match the folder layout. In LibTest? `namespace Test {}` in



LibTest/Randomized? `namespace Test::Randomized` etc.

- Template parameter names should generally follow the same naming scheme as normal classes, with the obvious exception of T, U, etc (single char) (looking at you, SET_FN, and FN)
- static inline functions in headers are no bueno. it's a nice trap to duplicate the function into every TU independently.
- I wonder if your ASSUME/REJECT etc error cases could be modeled with `AK::ErrorOr<T, GenerationError>`, our common result/std::expected-like type?
  - At least one of them, for `one_of` could just be a `requires` clause on the end of the template function declaration `) requires (sizeof...(Ts) > 0;) or similar. We are in C++-20 after all.
- Some of your algorithms bear a striking resemblance to ones in AK. `one_of` comes to mind again. @ldm5180 added `any_of` in AK/AnyOf a while back, but we've been missing that drive to add generic algorithms lately...

... ok maybe these aren't generic. but those things are all a bit distracting from the cool feature you've added to let us validate properties of our libraries without relying on oss-fuzz :)

Janiczek force-pushed the `property-based-tests` branch 7 times, most recently from `d6f4feb` to `59d1600` 5 months ago — Compare

Janiczek mentioned this pull request on Oct 11, 2023

AK: Fix one-off error in BitmapView::find_first and find_one_anywhere #21409 — Merged

timschumi self-requested a review 4 months ago

timschumi requested changes on Oct 11, 2023 — View reviewed changes

timschumi left a comment — Member

First round of feedback, mostly style-related.

```
Userland/Libraries/LibTest/TestResult.h  Outdated                       Show resolved
Userland/Libraries/LibTest/Randomized/Chunk.h  Outdated                 Show resolved
Userland/Libraries/LibTest/Randomized/RandomRun.h  Outdated             Show resolved
Userland/Libraries/LibTest/Randomized/RandomRun.h  Outdated             Show resolved
Userland/Libraries/LibTest/Randomized/RandomRun.h  Outdated             Show resolved
                        11 hidden conversations
                              Load more...
Userland/Libraries/LibTest/Randomized/ShrinkCmd.h  Outdated             Show resolved
Userland/Libraries/LibTest/Randomized/ShrinkCmd.h  Outdated             Show resolved
Userland/Libraries/LibTest/Randomized/Shrink.h  Outdated                Show resolved
Userland/Libraries/LibTest/Macros.h  Outdated                           Show resolved
Tests/LibTest/TestGenerator.cpp  Outdated                               Show resolved
```

Janiczek force-pushed the `property-based-tests` branch from `59d1600` to `4f472e5` 4 months ago — Compare

Janiczek requested a review from **timschumi** 4 months ago

Janiczek force-pushed the `property-based-tests` branch 3 times, most recently from `ac41370` to `1f6a26c` 4 months ago — Compare

ADKaster requested changes on Oct 18, 2023 — View reviewed changes

ADKaster left a comment — Member

All right, here's my nitpicks. I think most of them should be trivial, but there are a few thinker questions hidden in the noise :)

```
Userland/Libraries/LibTest/Macros.h  Outdated                           Show resolved
Userland/Libraries/LibTest/TestSuite.h  Outdated                        Show resolved
Userland/Libraries/LibTest/Randomized/Chunk.h  Outdated                 Show resolved
Userland/Libraries/LibTest/Randomized/RandomRun.h  Outdated             Show resolved
Userland/Libraries/LibTest/Randomized/RandomRun.h  Outdated             Show resolved
                        5 hidden conversations
                              Load more...
Userland/Libraries/LibTest/Randomized/ShrinkCommand.h                   Show resolved
Userland/Libraries/LibTest/Randomized/ShrinkCommand.h  Outdated         Show resolved
Userland/Libraries/LibTest/Randomized/ShrinkCommand.h  Outdated         Show resolved
Userland/Libraries/LibTest/Randomized/Shrink.h  Outdated                Show resolved
Userland/Libraries/LibTest/TestCase.h  Outdated                         Show resolved
```

Janiczek force-pushed the `property-based-tests` branch from `1f6a26c` to `92dba37` 4 months ago — Compare

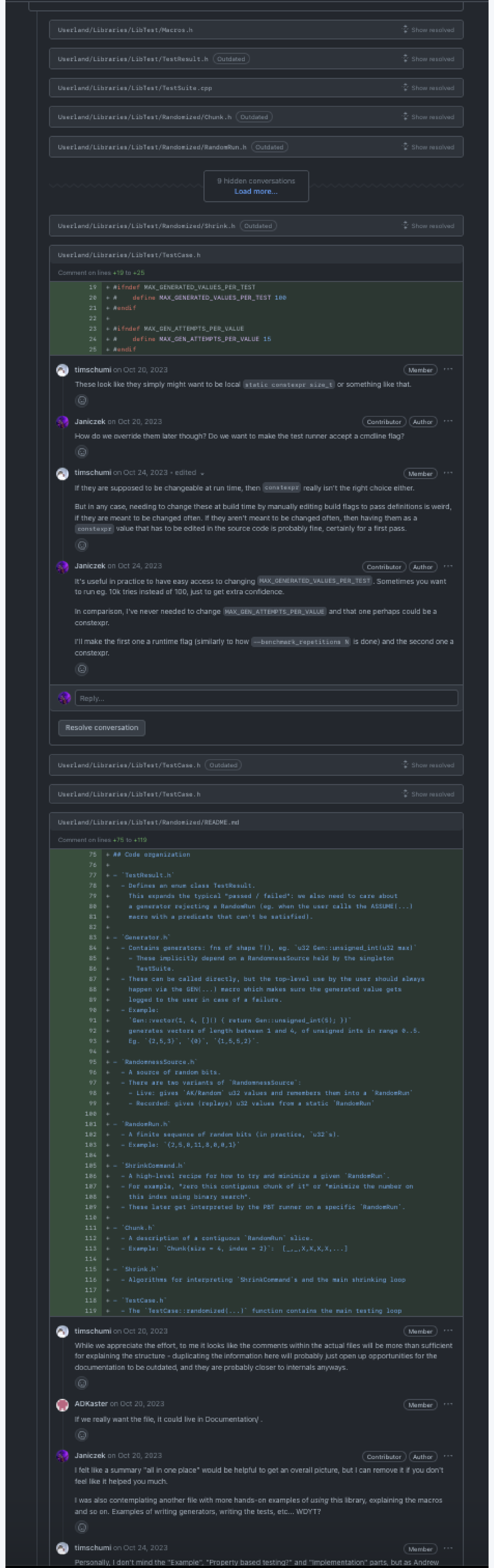
Janiczek requested a review from **ADKaster** 4 months ago

Janiczek force-pushed the `property-based-tests` branch from `92dba37` to `2e8cc93` 4 months ago — Compare

timschumi requested changes on Oct 20, 2023 — View reviewed changes

timschumi left a comment — Member

If any of the comments here have already been resolved in advance feel free to disregard them, the first few I started writing a some days ago.



```
Userland/Libraries/LibTest/Macros.h                                     Show resolved
Userland/Libraries/LibTest/TestResult.h  Outdated                       Show resolved
Userland/Libraries/LibTest/TestSuite.cpp                                Show resolved
Userland/Libraries/LibTest/Randomized/Chunk.h  Outdated                 Show resolved
Userland/Libraries/LibTest/Randomized/RandomRun.h  Outdated             Show resolved
                        9 hidden conversations
                              Load more...
Userland/Libraries/LibTest/Randomized/Shrink.h  Outdated                Show resolved
```

`Userland/Libraries/LibTest/TestCase.h`

Comment on lines +19 to +25

```
19 + #ifndef MAX_GENERATED_VALUES_PER_TEST
20 + #    define MAX_GENERATED_VALUES_PER_TEST 100
21 + #endif
22 +
23 + #ifndef MAX_GEN_ATTEMPTS_PER_VALUE
24 + #    define MAX_GEN_ATTEMPTS_PER_VALUE 15
25 + #endif
```

timschumi on Oct 20, 2023 — Member

These look like they simply might want to be local `static constexpr size_t` or something like that.

Janiczek on Oct 20, 2023 — Contributor Author

How do we override them later though? Do we want to make the test runner accept a cmdline flag?

timschumi on Oct 24, 2023 • edited — Member

If they are supposed to be changeable at run time, then `constexpr` really isn't the right choice either.

But in any case, needing to change these at build time by manually editing build flags to pass definitions is weird, if they are meant to be changed often. If they aren't meant to be changed often, then having them as a `constexpr` value that has to be edited in the source code is probably fine, certainly for a first pass.

Janiczek on Oct 24, 2023 — Contributor Author

It's useful in practice to have easy access to changing `MAX_GENERATED_VALUES_PER_TEST`. Sometimes you want to run eg. 10k tries instead of 100, just to get extra confidence.

In comparison, I've never needed to change `MAX_GEN_ATTEMPTS_PER_VALUE` and that one perhaps could be a constexpr.

I'll make the first one a runtime flag (similarly to how `--benchmark_repetitions N` is done) and the second one a constexpr.

```
Userland/Libraries/LibTest/TestCase.h  Outdated                         Show resolved
Userland/Libraries/LibTest/TestCase.h                                   Show resolved
```

`Userland/Libraries/LibTest/Randomized/README.md`

Comment on lines +75 to +119

```
 75 + ## Code organization
 76 +
 77 + - `TestResult.h`
 78 +   - Defines an enum class TestResult.
 79 +     This expands the typical "passed / failed": we also need to care about
 80 +     a generator rejecting a RandomRun (eg. when the user calls the ASSUME(...)
 81 +     macro with a predicate that can't be satisfied).
 82 +
 83 + - `Generator.h`
 84 +   - Contains generators: fns of shape T(), eg. `u32 Gen::unsigned_int(u32 max)`
 85 +     - These implicitly depend on a RandomnessSource held by the singleton
 86 +       TestSuite.
 87 +   - These can be called directly, but the top-level use by the user should always
 88 +     happen via the GEN(...) macro which makes sure the generated value gets
 89 +     logged to the user in case of a failure.
 90 +   - Example:
 91 +     `Gen::vector(1, 4, []() { return Gen::unsigned_int(5); })`
 92 +     generates vectors of length between 1 and 4, of unsigned ints in range 0..5.
 93 +     Eg. `{2,5,3}`, `{0}`, `{1,5,5,2}`.
 94 +
 95 + - `RandomnessSource.h`
 96 +   - A source of random bits.
 97 +   - There are two variants of `RandomnessSource`:
 98 +     - Live: gives `AK/Random` u32 values and remembers them into a `RandomRun`
 99 +     - Recorded: gives (replays) u32 values from a static `RandomRun`
100 +
101 + - `RandomRun.h`
102 +   - A finite sequence of random bits (in practice, `u32`s).
103 +   - Example: `{2,5,0,11,8,0,0,1}`
104 +
105 + - `ShrinkCommand.h`
106 +   - A high-level recipe for how to try and minimize a given `RandomRun`.
107 +   - For example, "zero this contiguous chunk of it" or "minimize the number on
108 +     this index using binary search".
109 +   - These later get interpreted by the PBT runner on a specific `RandomRun`.
110 +
111 + - `Chunk.h`
112 +   - A description of a contiguous `RandomRun` slice.
113 +   - Example: `Chunk{size = 4, index = 2}`:  [_,_,X,X,X,X,...]
114 +
115 + - `Shrink.h`
116 +   - Algorithms for interpreting `ShrinkCommand`s and the main shrinking loop
117 +
118 + - `TestCase.h`
119 +   - The `TestCase::randomized(...)` function contains the main testing loop
```

timschumi on Oct 20, 2023 — Member

While we appreciate the effort, to me it looks like the comments within the actual files will be more than sufficient for explaining the structure - duplicating the information here will probably just open up opportunities for the documentation to be outdated, and they are probably closer to internals anyways.

ADKaster on Oct 20, 2023 — Member

If we really want the file, it could live in Documentation/ .

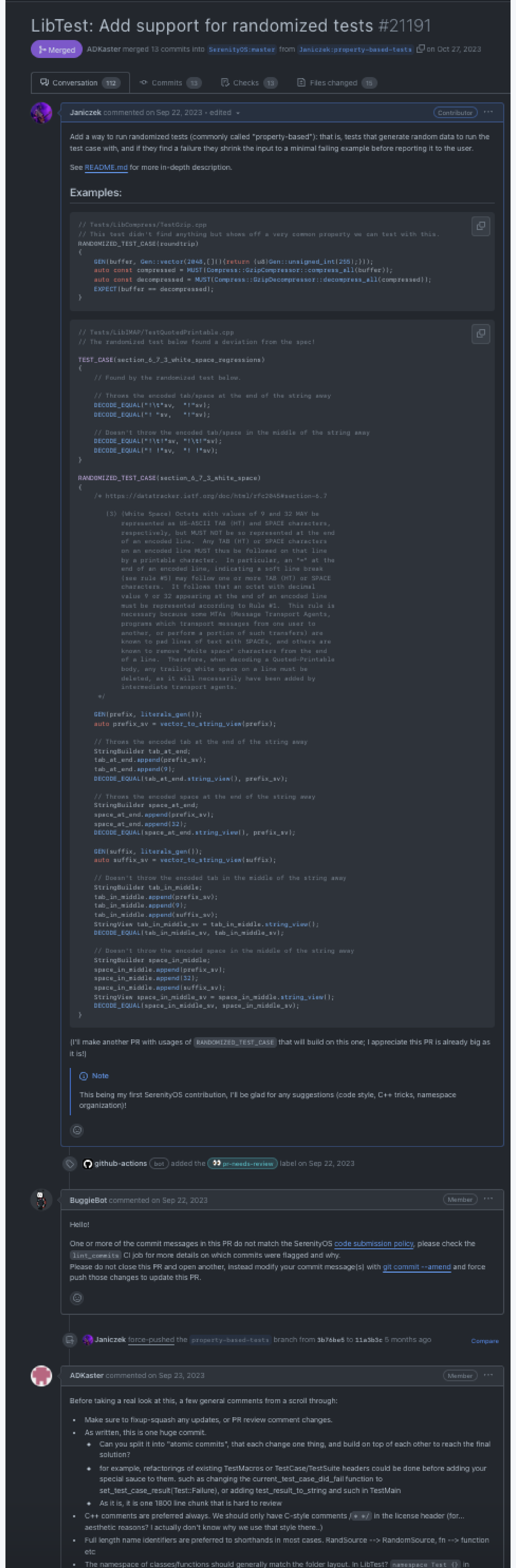
Janiczek on Oct 20, 2023 — Contributor Author

I felt like a summary "all in one place" would be helpful to get an overall picture, but I can remove it if you don't feel like it helped you much.

I was also contemplating another file with more hands-on examples of using this library, explaining the macros and so on. Examples of writing generators, writing the tests, etc... WDYT?

timschumi on Oct 24, 2023 — Member

Personally, I don't mind the "Example", "Property based testing?" and "Implementation" parts, but as Andrew



LibTest: Add support for randomized tests #21191

Merged ADKaster merged 13 commits into `SerenityOS:master` from `Janiczek:property-based-tests` on Oct 27, 2023

Janiczek commented on Sep 22, 2023 • edited

Add a way to run randomized tests (commonly called "property-based"): that is, tests that generate random data to run the test case with, and if they find a failure they shrink the input to a minimal failing example before reporting it to the user.

See README.md for more in-depth description.

Examples:

```
// Tests/LibCompress/TestGzip.cpp
// This test didn't find anything but shows off a very common property we can test with this.
RANDOMIZED_TEST_CASE(roundtrip)
{
    GEN(buffer, Gen::vector(2048,[](){return (u8)Gen::unsigned_int(255);}));
    auto const compressed = MUST(Compress::GzipCompressor::compress_all(buffer));
    auto const decompressed = MUST(Compress::GzipDecompressor::decompress_all(compressed));
    EXPECT(buffer == decompressed);
}
```

```
// Tests/LibIMAP/TestQuotedPrintable.cpp
// The randomized test below found a deviation from the spec!

TEST_CASE(section_6_7_3_white_space_regressions)
{
    // Found by the randomized test below.

    // Throws the encoded tab/space at the end of the string away
    DECODE_EQUAL("!\t"sv,  "!"sv);
    DECODE_EQUAL("! "sv,   "!"sv);

    // Doesn't throw the encoded tab/space in the middle of the string away
    DECODE_EQUAL("!\t!"sv, "!\t!"sv);
    DECODE_EQUAL("! !"sv,  "! !"sv);
}

RANDOMIZED_TEST_CASE(section_6_7_3_white_space)
{
    /* https://datatracker.ietf.org/doc/html/rfc2045#section-6.7

       (3) (White Space) Octets with values of 9 and 32 MAY be
           represented as US-ASCII TAB (HT) and SPACE characters,
           respectively, but MUST NOT be so represented at the end
           of an encoded line.  Any TAB (HT) or SPACE characters
           on an encoded line MUST thus be followed on that line
           by a printable character.  In particular, an "=" at the
           end of an encoded line, indicating a soft line break
           (see rule #5) may follow one or more TAB (HT) or SPACE
           characters.  It follows that an octet with decimal
           value 9 or 32 appearing at the end of an encoded line
           must be represented according to Rule #1.  This rule is
           necessary because some MTAs (Message Transport Agents,
           programs which transport messages from one user to
           another, or perform a portion of such transfers) are
           known to pad lines of text with SPACEs, and others are
           known to remove "white space" characters from the end
           of a line.  Therefore, when decoding a Quoted-Printable
           body, any trailing white space on a line must be
           deleted, as it will necessarily have been added by
           intermediate transport agents.
     */

    GEN(prefix, literals_gen());
    auto prefix_sv = vector_to_string_view(prefix);

    // Throws the encoded tab at the end of the string away
    StringBuilder tab_at_end;
    tab_at_end.append(prefix_sv);
    tab_at_end.append(9);
    DECODE_EQUAL(tab_at_end.string_view(), prefix_sv);

    // Throws the encoded space at the end of the string away
    StringBuilder space_at_end;
    space_at_end.append(prefix_sv);
    space_at_end.append(32);
    DECODE_EQUAL(space_at_end.string_view(), prefix_sv);

    GEN(suffix, literals_gen());
    auto suffix_sv = vector_to_string_view(suffix);

    // Doesn't throw the encoded tab in the middle of the string away
    StringBuilder tab_in_middle;
    tab_in_middle.append(prefix_sv);
    tab_in_middle.append(9);
    tab_in_middle.append(suffix_sv);
    StringView tab_in_middle_sv = tab_in_middle.string_view();
    DECODE_EQUAL(tab_in_middle_sv, tab_in_middle_sv);

    // Doesn't throw the encoded space in the middle of the string away
    StringBuilder space_in_middle;
    space_in_middle.append(prefix_sv);
    space_in_middle.append(32);
    space_in_middle.append(suffix_sv);
    StringView space_in_middle_sv = space_in_middle.string_view();
    DECODE_EQUAL(space_in_middle_sv, space_in_middle_sv);
}
```

(I'll make another PR with usages of `RANDOMIZED_TEST_CASE` that will build on this one; I appreciate this PR is already big as it is!)

> Note
>
> This being my first SerenityOS contribution, I'll be glad for any suggestions (code style, C++ tricks, namespace organization)!

github-actions added the label on Sep 22, 2023

BuggieBot commented on Sep 22, 2023

Hello!

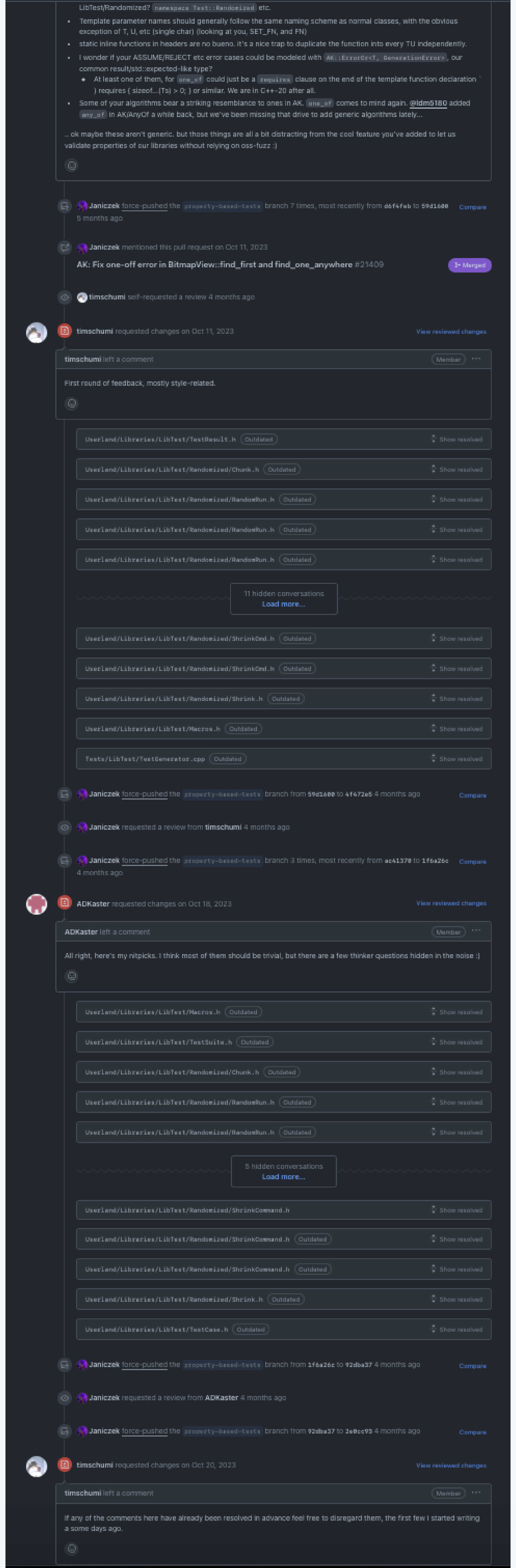
One or more of the commit messages in this PR do not match the SerenityOS code submission policy, please check the `lint_commits` CI job for more details on which commits were flagged and why.
Please do not close this PR and open another, instead modify your commit message(s) with git commit --amend and force push those changes to update this PR.

Janiczek force-pushed the `property-based-tests` branch from `3b76be5` to `11a3b3c` 5 months ago

ADKaster commented on Sep 23, 2023

Before taking a real look at this, a few general comments from a scroll through:

- Make sure to fixup-squash any updates, or PR review comment changes.
- As written, this is one huge commit.
  - Can you split it into "atomic commits", that each change one thing, and build on top of each other to reach the final solution?
  - for example, refactorings of existing TestMacros or TestCase/TestSuite headers could be done before adding your special sauce to them. such as changing the current_test_case_did_fail function to set_test_case_result(Test::Failure), or adding test_result_to_string and such in TestMain
  - As it is, it is one 1800 line chunk that is hard to review
- C++ comments are preferred always. We should only have C-style comments `/* */` in the license header (for... aesthetic reasons? I actually don't know why we use that style there..)
- Full length name identifiers are preferred to shorthands in most cases. RandSource --> RandomSource, fn --> function etc
- The namespace of classes/functions should generally match the folder layout. In LibTest? `namespace Test {}` in



LibTest/Randomized? `namespace Test::Randomized` etc.
- Template parameter names should generally follow the same naming scheme as normal classes, with the obvious exception of T, U, etc (single char) (looking at you, SET_FN, and FN)
- static inline functions in headers are no bueno. it's a nice trap to duplicate the function into every TU independently.
- I wonder if your ASSUME/REJECT etc error cases could be modeled with `AK::ErrorOr<T, GenerationError>`, our common result/std::expected-like type?
  - At least one of them, for `one_of` could just be a `requires` clause on the end of the template function declaration `) requires (sizeof...(Ts) > 0;) or similar. We are in C++-20 after all.
- Some of your algorithms bear a striking resemblance to ones in AK. `one_of` comes to mind again. @ldm5180 added `any_of` in AK/AnyOf a while back, but we've been missing that drive to add generic algorithms lately...

... ok maybe these aren't generic. but those things are all a bit distracting from the cool feature you've added to let us validate properties of our libraries without relying on oss-fuzz :)

Janiczek force-pushed the `property-based-tests` branch 7 times, most recently from `d6f4feb` to `59d1600` 5 months ago

Janiczek mentioned this pull request on Oct 11, 2023

AK: Fix one-off error in BitmapView::find_first and find_one_anywhere #21409

timschumi self-requested a review 4 months ago

timschumi requested changes on Oct 11, 2023

timschumi left a comment

First round of feedback, mostly style-related.

```
Userland/Libraries/LibTest/TestResult.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/Chunk.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/RandomRun.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/RandomRun.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/RandomRun.h  Outdated  Show resolved
```

11 hidden conversations
Load more...

```
Userland/Libraries/LibTest/Randomized/ShrinkCmd.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/ShrinkCmd.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/Shrink.h  Outdated  Show resolved
Userland/Libraries/LibTest/Macros.h  Outdated  Show resolved
Tests/LibTest/TestGenerator.cpp  Outdated  Show resolved
```

Janiczek force-pushed the `property-based-tests` branch from `59d1600` to `4f472e5` 4 months ago

Janiczek requested a review from timschumi 4 months ago

Janiczek force-pushed the `property-based-tests` branch 3 times, most recently from `ac41370` to `1f6a26c` 4 months ago

ADKaster requested changes on Oct 18, 2023

ADKaster left a comment

All right, here's my nitpicks. I think most of them should be trivial, but there are a few thinker questions hidden in the noise :)

```
Userland/Libraries/LibTest/Macros.h  Outdated  Show resolved
Userland/Libraries/LibTest/TestSuite.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/Chunk.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/RandomRun.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/RandomRun.h  Outdated  Show resolved
```

5 hidden conversations
Load more...

```
Userland/Libraries/LibTest/Randomized/ShrinkCommand.h  Show resolved
Userland/Libraries/LibTest/Randomized/ShrinkCommand.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/ShrinkCommand.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/Shrink.h  Outdated  Show resolved
Userland/Libraries/LibTest/TestCase.h  Outdated  Show resolved
```

Janiczek force-pushed the `property-based-tests` branch from `1f6a26c` to `92dba37` 4 months ago

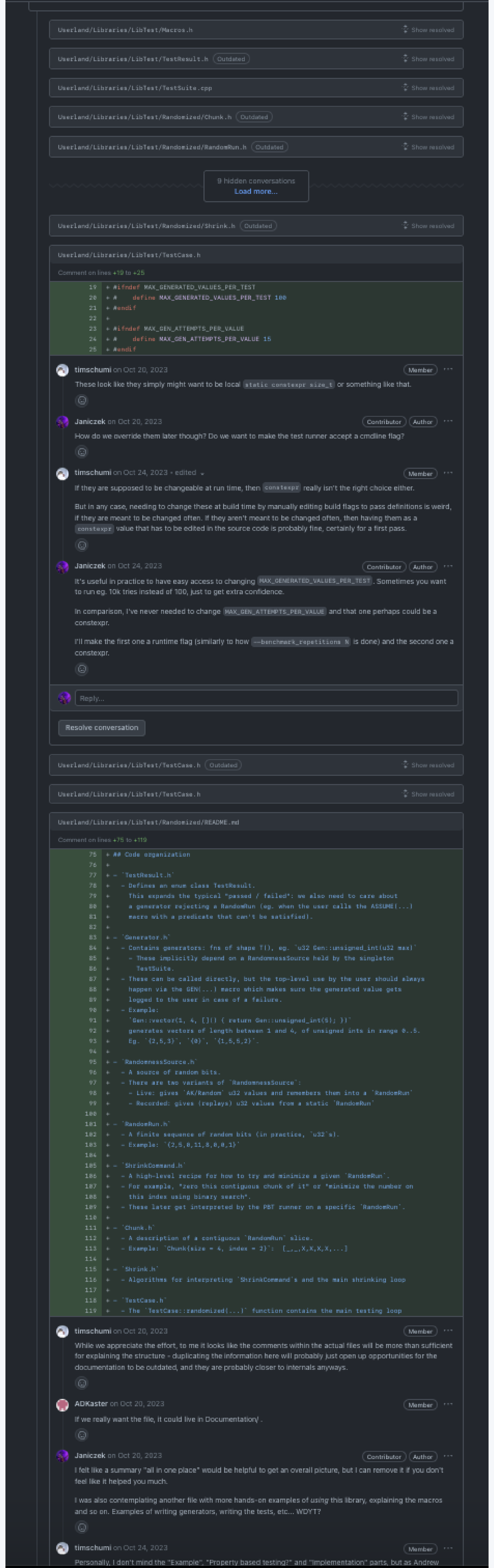
Janiczek requested a review from ADKaster 4 months ago

Janiczek force-pushed the `property-based-tests` branch from `92dba37` to `2e8cc93` 4 months ago

timschumi requested changes on Oct 20, 2023

timschumi left a comment

If any of the comments here have already been resolved in advance feel free to disregard them, the first few I started writing a some days ago.



```
Userland/Libraries/LibTest/Macros.h  Show resolved
Userland/Libraries/LibTest/TestResult.h  Outdated  Show resolved
Userland/Libraries/LibTest/TestSuite.cpp  Show resolved
Userland/Libraries/LibTest/Randomized/Chunk.h  Outdated  Show resolved
Userland/Libraries/LibTest/Randomized/RandomRun.h  Outdated  Show resolved
```

9 hidden conversations
Load more...

```
Userland/Libraries/LibTest/Randomized/Shrink.h  Outdated  Show resolved
```

`Userland/Libraries/LibTest/TestCase.h`

Comment on lines +19 to +25

```
19 + #ifndef MAX_GENERATED_VALUES_PER_TEST
20 + #    define MAX_GENERATED_VALUES_PER_TEST 100
21 + #endif
22 +
23 + #ifndef MAX_GEN_ATTEMPTS_PER_VALUE
24 + #    define MAX_GEN_ATTEMPTS_PER_VALUE 15
25 + #endif
```

timschumi on Oct 20, 2023

These look like they simply might want to be local `static constexpr size_t` or something like that.

Janiczek on Oct 20, 2023

How do we override them later though? Do we want to make the test runner accept a cmdline flag?

timschumi on Oct 24, 2023 • edited

If they are supposed to be changeable at run time, then `constexpr` really isn't the right choice either.

But in any case, needing to change these at build time by manually editing build flags to pass definitions is weird, if they are meant to be changed often. If they aren't meant to be changed often, then having them as a `constexpr` value that has to be edited in the source code is probably fine, certainly for a first pass.

Janiczek on Oct 24, 2023

It's useful in practice to have easy access to changing `MAX_GENERATED_VALUES_PER_TEST`. Sometimes you want to run eg. 10k tries instead of 100, just to get extra confidence.

In comparison, I've never needed to change `MAX_GEN_ATTEMPTS_PER_VALUE` and that one perhaps could be a constexpr.

I'll make the first one a runtime flag (similarly to how `--benchmark_repetitions N` is done) and the second one a constexpr.

```
Userland/Libraries/LibTest/TestCase.h  Outdated  Show resolved
Userland/Libraries/LibTest/TestCase.h  Show resolved
```

`Userland/Libraries/LibTest/Randomized/README.md`

Comment on lines +75 to +119

```
 75 + ## Code organization
 76 +
 77 + - `TestResult.h`
 78 +   - Defines an enum class TestResult.
 79 +     This expands the typical "passed / failed": we also need to care about
 80 +     a generator rejecting a RandomRun (eg. when the user calls the ASSUME(...)
 81 +     macro with a predicate that can't be satisfied).
 82 +
 83 + - `Generator.h`
 84 +   - Contains generators: fns of shape T(), eg. `u32 Gen::unsigned_int(u32 max)`
 85 +     - These implicitly depend on a RandomnessSource held by the singleton
 86 +       TestSuite.
 87 +   - These can be called directly, but the top-level use by the user should always
 88 +     happen via the GEN(...) macro which makes sure the generated value gets
 89 +     logged to the user in case of a failure.
 90 +   - Example:
 91 +     `Gen::vector(1, 4, []() { return Gen::unsigned_int(5); })`
 92 +     generates vectors of length between 1 and 4, of unsigned ints in range 0..5.
 93 +     Eg. `{2,5,3}`, `{0}`, `{1,5,5,2}`.
 94 +
 95 + - `RandomnessSource.h`
 96 +   - A source of random bits.
 97 +   - There are two variants of `RandomnessSource`:
 98 +     - Live: gives `AK/Random` u32 values and remembers them into a `RandomRun`
 99 +     - Recorded: gives (replays) u32 values from a static `RandomRun`
100 +
101 + - `RandomRun.h`
102 +   - A finite sequence of random bits (in practice, `u32`s).
103 +   - Example: `{2,5,0,11,8,0,0,1}`
104 +
105 + - `ShrinkCommand.h`
106 +   - A high-level recipe for how to try and minimize a given `RandomRun`.
107 +   - For example, "zero this contiguous chunk of it" or "minimize the number on
108 +     this index using binary search".
109 +   - These later get interpreted by the PBT runner on a specific `RandomRun`.
110 +
111 + - `Chunk.h`
112 +   - A description of a contiguous `RandomRun` slice.
113 +   - Example: `Chunk{size = 4, index = 2}`:  [_,_,X,X,X,X,...]
114 +
115 + - `Shrink.h`
116 +   - Algorithms for interpreting `ShrinkCommand`s and the main shrinking loop
117 +
118 + - `TestCase.h`
119 +   - The `TestCase::randomized(...)` function contains the main testing loop
```

timschumi on Oct 20, 2023

While we appreciate the effort, to me it looks like the comments within the actual files will be more than sufficient for explaining the structure - duplicating the information here will probably just open up opportunities for the documentation to be outdated, and they are probably closer to internals anyways.

ADKaster on Oct 20, 2023

If we really want the file, it could live in Documentation/ .

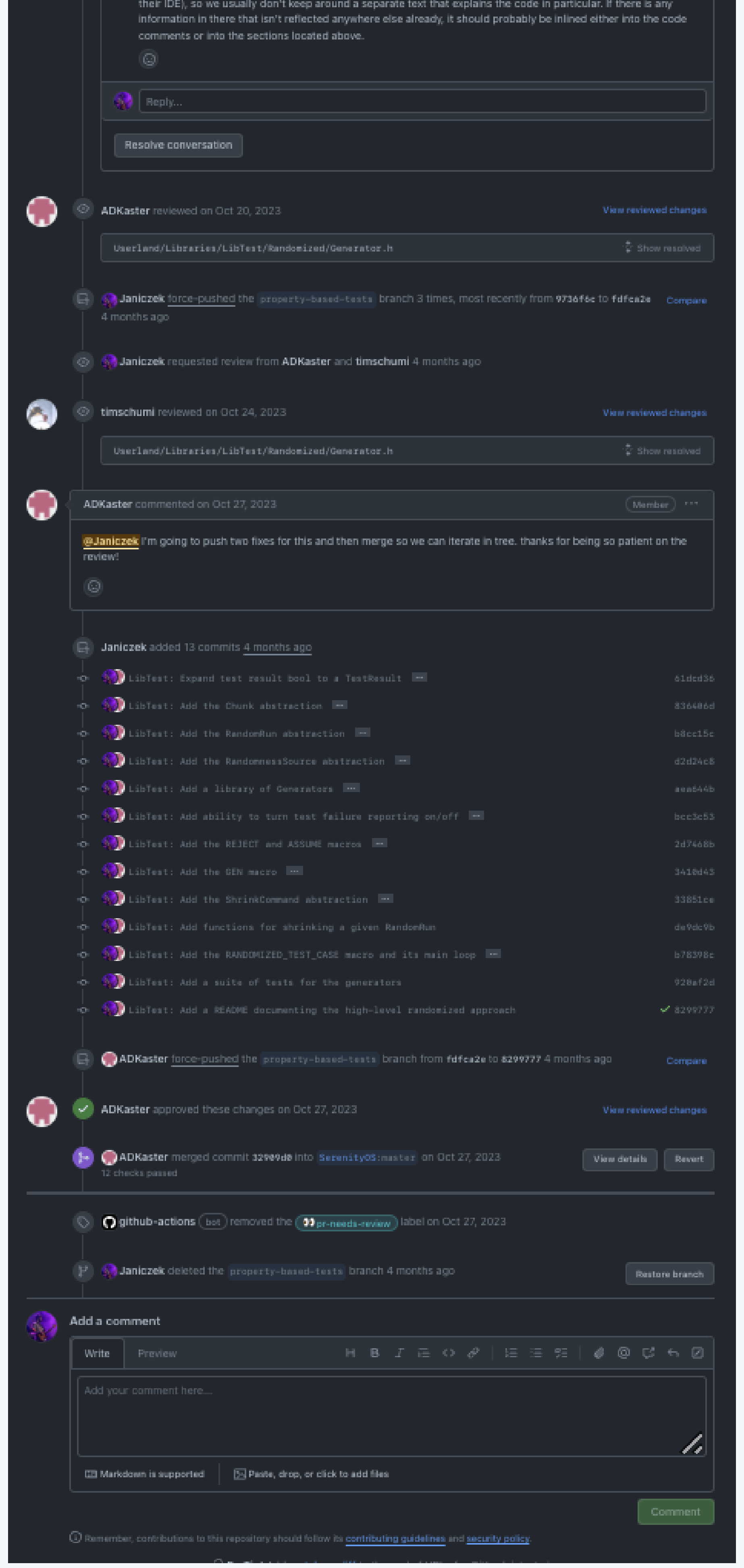
Janiczek on Oct 20, 2023

I felt like a summary "all in one place" would be helpful to get an overall picture, but I can remove it if you don't feel like it helped you much.

I was also contemplating another file with more hands-on examples of using this library, explaining the macros and so on. Examples of writing generators, writing the tests, etc... WDYT?

timschumi on Oct 24, 2023

Personally, I don't mind the "Example", "Property based testing?" and "Implementation" parts, but as Andrew



their IDE), so we usually don't keep around a separate text that explains the code in particular. If there is any information in there that isn't reflected anywhere else already, it should probably be inlined either into the code comments or into the sections located above.

ADKaster reviewed on Oct 20, 2023

```
Userland/Libraries/LibTest/Randomized/Generator.h  Show resolved
```

Janiczek force-pushed the `property-based-tests` branch 3 times, most recently from `9736f6c` to `fdfca2e` 4 months ago

Janiczek requested review from ADKaster and timschumi 4 months ago

timschumi reviewed on Oct 24, 2023

```
Userland/Libraries/LibTest/Randomized/Generator.h  Show resolved
```

ADKaster commented on Oct 27, 2023

@Janiczek I'm going to push two fixes for this and then merge so we can iterate in tree. thanks for being so patient on the review!

Janiczek added 13 commits 4 months ago

`LibTest: Expand test result bool to a TestResult`	61dcd36
`LibTest: Add the Chunk abstraction`	836406d
`LibTest: Add the RandomRun abstraction`	b8cc15c
`LibTest: Add the RandomnessSource abstraction`	d2d24c8
`LibTest: Add a library of Generators`	aea644b
`LibTest: Add ability to turn test failure reporting on/off`	bcc3c53
`LibTest: Add the REJECT and ASSUME macros`	2d7468b
`LibTest: Add the GEN macro`	3410d43
`LibTest: Add the ShrinkCommand abstraction`	33851ce
`LibTest: Add functions for shrinking a given RandomRun`	de9dc9b
`LibTest: Add the RANDOMIZED_TEST_CASE macro and its main loop`	b78398c
`LibTest: Add a suite of tests for the generators`	920af2d
`LibTest: Add a README documenting the high-level randomized approach`	✓ 8299777

ADKaster force-pushed the `property-based-tests` branch from `fdfca2e` to `8299777` 4 months ago

ADKaster approved these changes on Oct 27, 2023

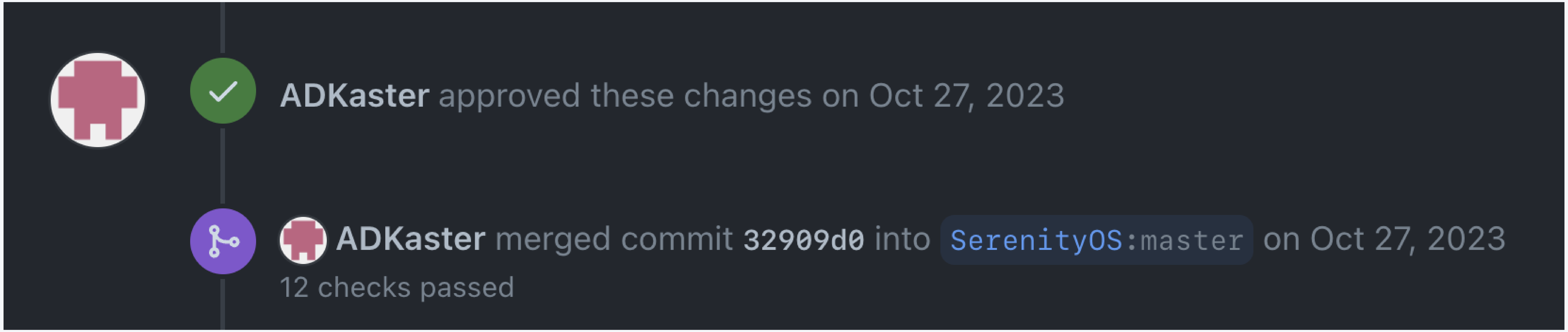
ADKaster merged commit `32909d0` into `SerenityOS:master` on Oct 27, 2023
12 checks passed

github-actions removed the label on Oct 27, 2023

Janiczek deleted the `property-based-tests` branch 4 months ago

Add a comment

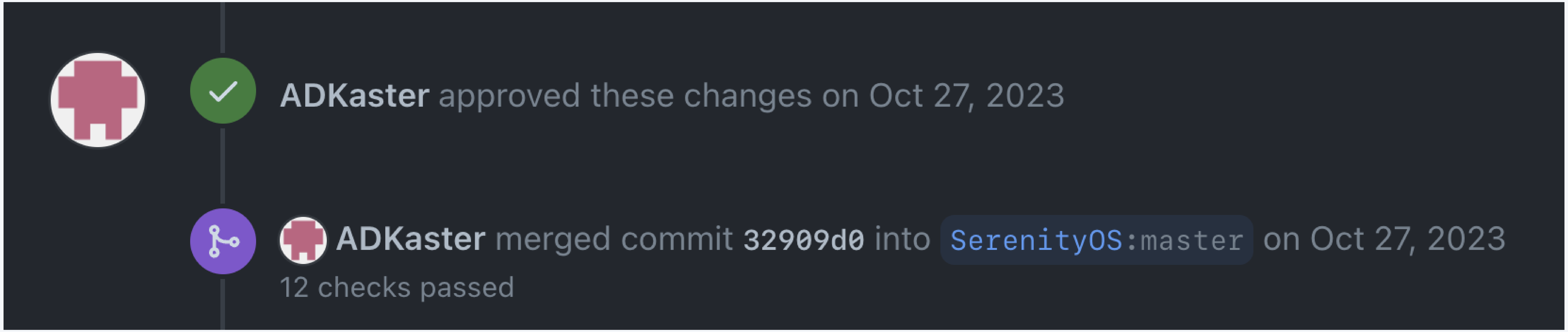
Remember, contributions to this repository should follow its contributing guidelines and security policy.



ADKaster approved these changes on Oct 27, 2023

ADKaster merged commit `32909d0` into `SerenityOS:master` on Oct 27, 2023

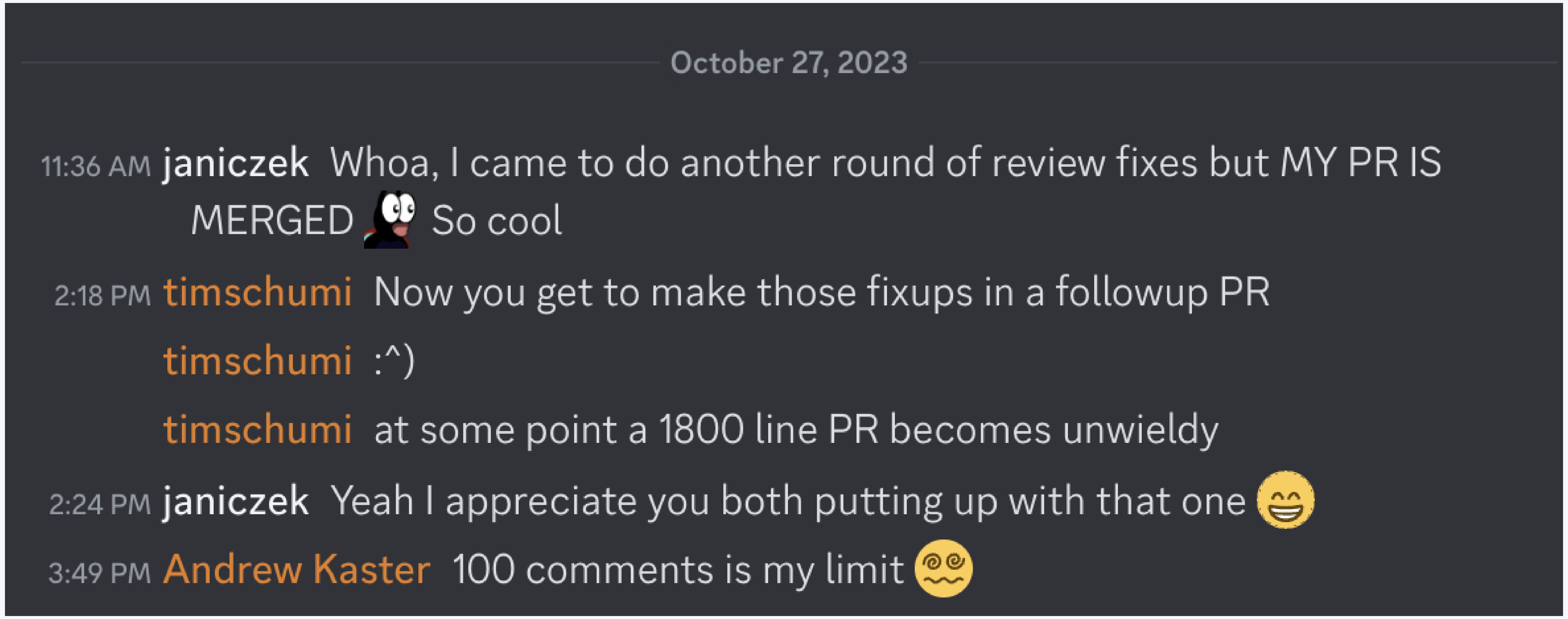
12 checks passed



ADKaster approved these changes on Oct 27, 2023

ADKaster merged commit 32909d0 into SerenityOS:master on Oct 27, 2023

12 checks passed



October 27, 2023

11:36 AM janiczek Whoa, I came to do another round of review fixes but MY PR IS MERGED So cool

2:18 PM timschumi Now you get to make those fixups in a followup PR

timschumi :^)

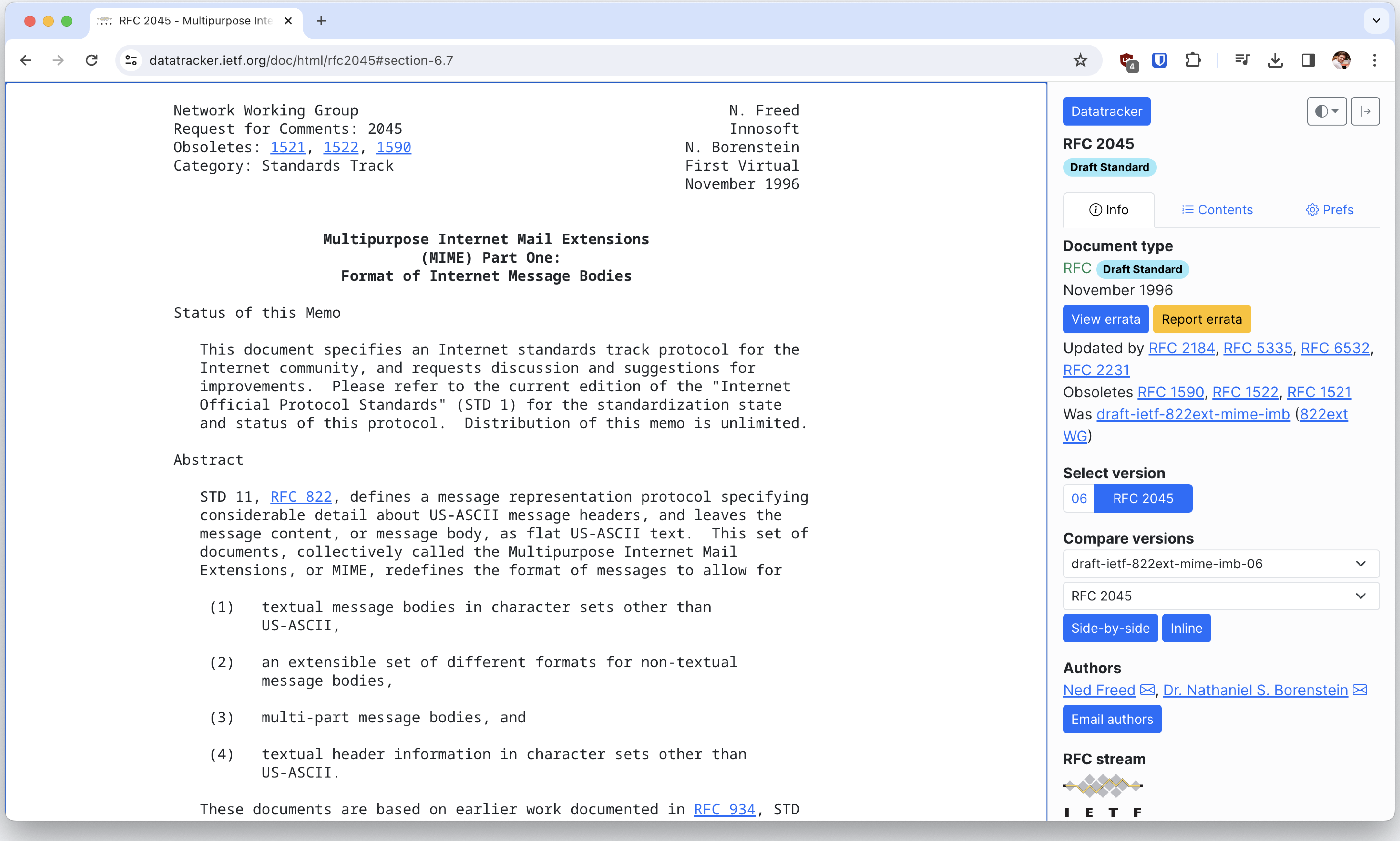
timschumi at some point a 1800 line PR becomes unwieldy

2:24 PM janiczek Yeah I appreciate you both putting up with that one 😁

3:49 PM Andrew Kaster 100 comments is my limit 😵‍💫

Examples:

```cpp
// Tests/LibCompress/TestGzip.cpp
// This test didn't find anything but shows off a very common property we can test with this.
RANDOMIZED_TEST_CASE(roundtrip)
{
    GEN(buffer, Gen::vector(2048,[](){return (u8)Gen::unsigned_int(255);}));
    auto const compressed = MUST(Compress::GzipCompressor::compress_all(buffer));
    auto const decompressed = MUST(Compress::GzipDecompressor::decompress_all(compressed));
    EXPECT(buffer == decompressed);
}
```



RFC 2045 - Multipurpose Inte
datatracker.ietf.org/doc/html/rfc2045#section-6.7

```
Network Working Group                                          N. Freed
Request for Comments: 2045                                     Innosoft
Obsoletes: 1521, 1522, 1590                               N. Borenstein
Category: Standards Track                                 First Virtual
                                                          November 1996


                 Multipurpose Internet Mail Extensions
                            (MIME) Part One:
                   Format of Internet Message Bodies

Status of this Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements.  Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol.  Distribution of this memo is unlimited.

Abstract

   STD 11, RFC 822, defines a message representation protocol specifying
   considerable detail about US-ASCII message headers, and leaves the
   message content, or message body, as flat US-ASCII text.  This set of
   documents, collectively called the Multipurpose Internet Mail
   Extensions, or MIME, redefines the format of messages to allow for

    (1)   textual message bodies in character sets other than
          US-ASCII,

    (2)   an extensible set of different formats for non-textual
          message bodies,

    (3)   multi-part message bodies, and

    (4)   textual header information in character sets other than
          US-ASCII.

   These documents are based on earlier work documented in RFC 934, STD
```

Datatracker

RFC 2045

Draft Standard

Info | Contents | Prefs

Document type

RFC Draft Standard

November 1996

View errata | Report errata

Updated by RFC 2184, RFC 5335, RFC 6532, RFC 2231

Obsoletes RFC 1590, RFC 1522, RFC 1521

Was draft-ietf-822ext-mime-imb (822ext WG)

Select version

06 | RFC 2045

Compare versions

draft-ietf-822ext-mime-imb-06

RFC 2045

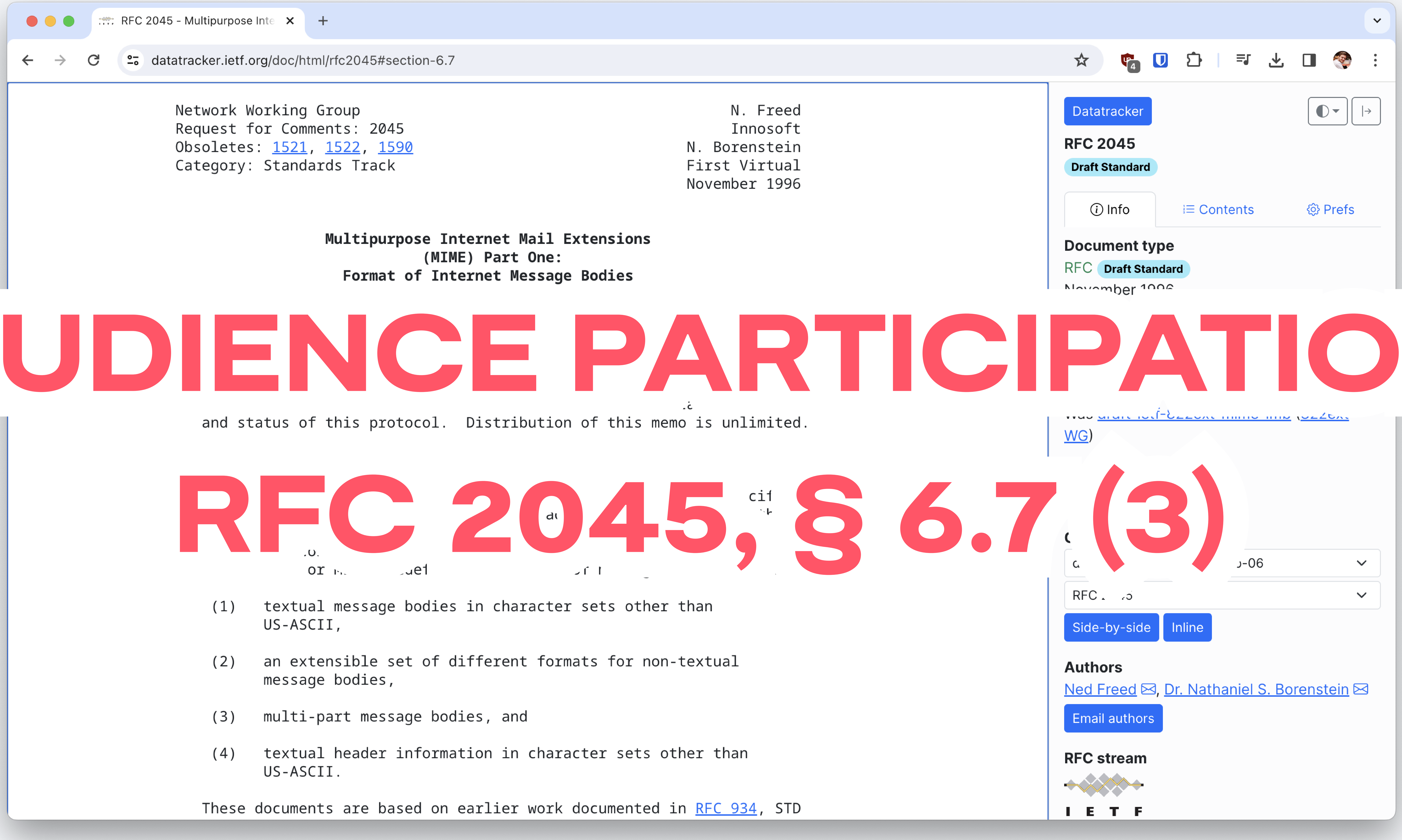
Side-by-side | Inline

Authors

Ned Freed, Dr. Nathaniel S. Borenstein

Email authors

RFC stream


RFC 2045 - Multipurpose Inte
datatracker.ietf.org/doc/html/rfc2045#section-6.7
Datatracker
Info
Contents
Prefs
Side-by-side
Inline
Email authors

Network Working Group N. Freed
Request for Comments: 2045 Innosoft
Obsoletes: 1521, 1522, 1590 N. Borenstein
Category: Standards Track First Virtual
November 1996

Multipurpose Internet Mail Extensions
(MIME) Part One:
Format of Internet Message Bodies

and status of this protocol. Distribution of this memo is unlimited.

(1) textual message bodies in character sets other than US-ASCII,

(2) an extensible set of different formats for non-textual message bodies,

(3) multi-part message bodies, and

(4) textual header information in character sets other than US-ASCII.

These documents are based on earlier work documented in RFC 934, STD

RFC 2045

Draft Standard

Document type

RFC Draft Standard

November 1996

WG)

RFC

Authors

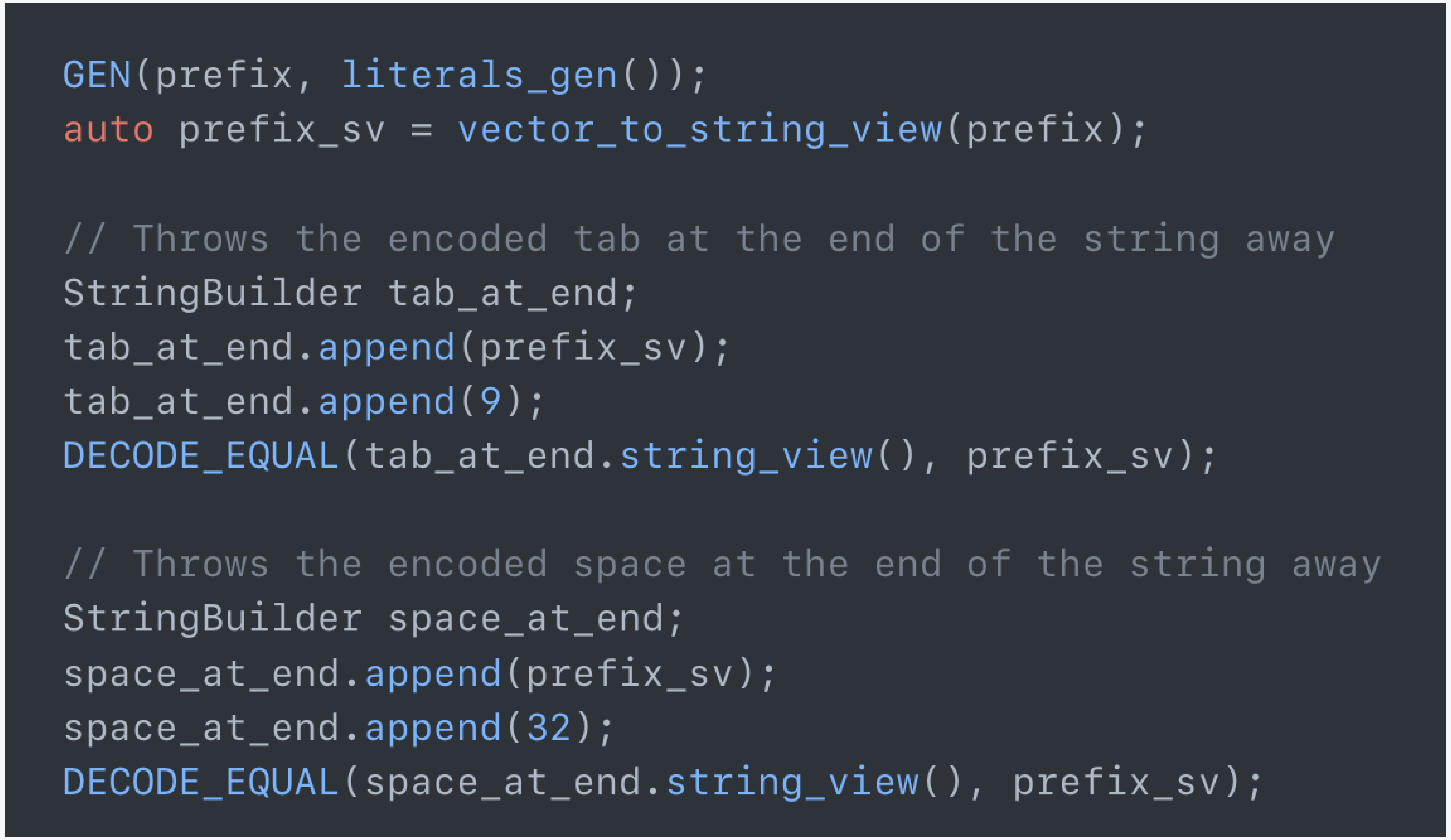
Ned Freed ✉, Dr. Nathaniel S. Borenstein ✉

RFC stream

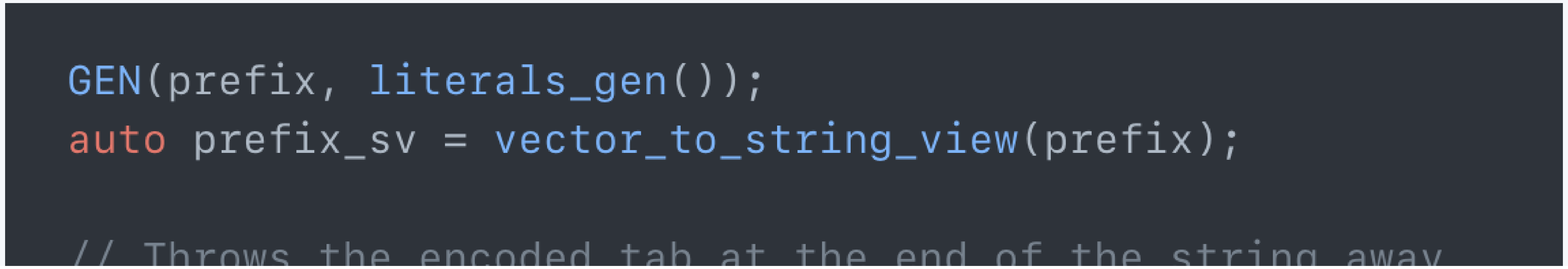
I E T F

AUDIENCE PARTICIPATION

RFC 2045, § 6.7 (3)

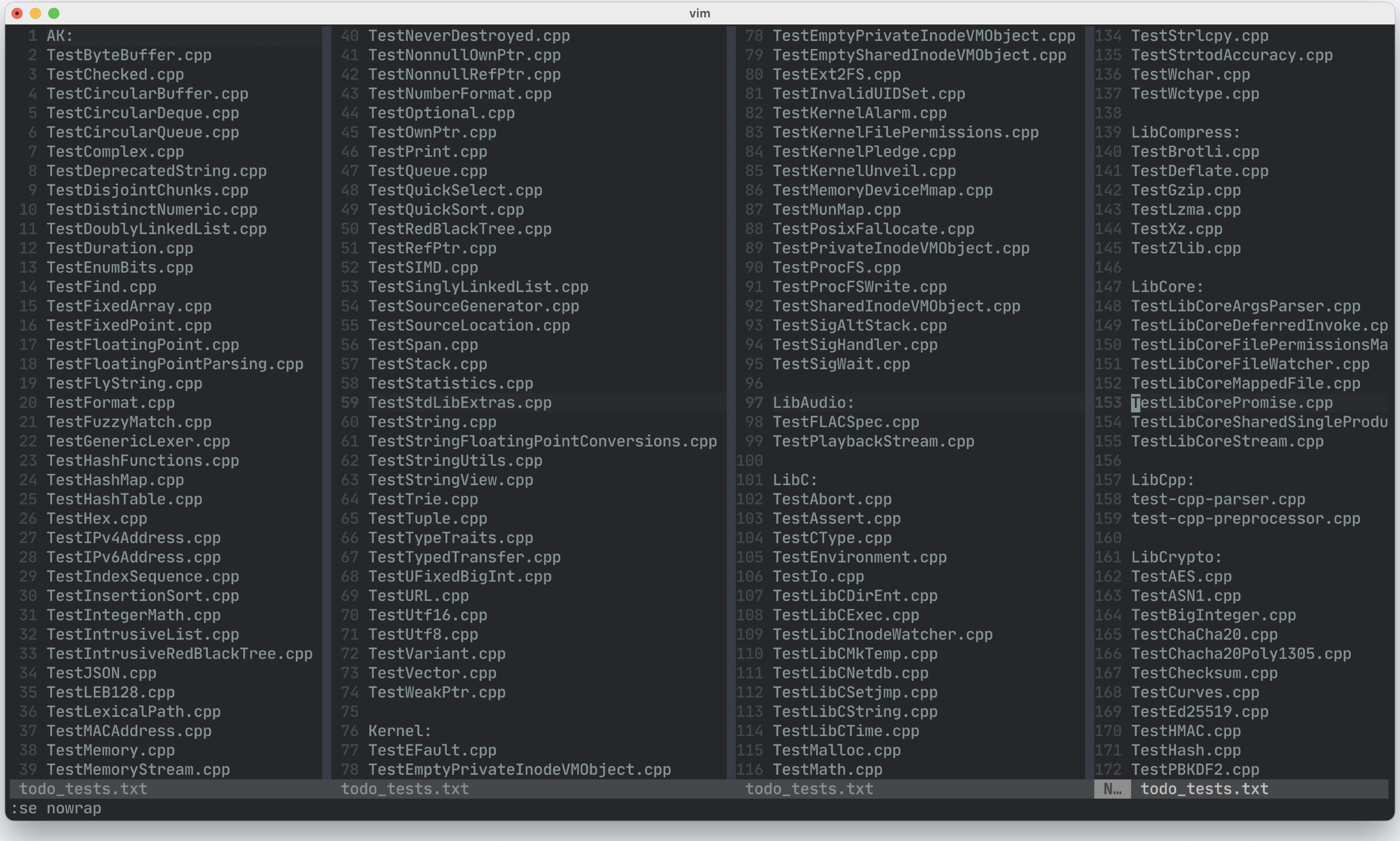


```
GEN(prefix, literals_gen());
auto prefix_sv = vector_to_string_view(prefix);

// Throws the encoded tab at the end of the string away
StringBuilder tab_at_end;
tab_at_end.append(prefix_sv);
tab_at_end.append(9);
DECODE_EQUAL(tab_at_end.string_view(), prefix_sv);

// Throws the encoded space at the end of the string away
StringBuilder space_at_end;
space_at_end.append(prefix_sv);
space_at_end.append(32);
DECODE_EQUAL(space_at_end.string_view(), prefix_sv);
```



```
GEN(prefix, literals_gen());
auto prefix_sv = vector_to_string_view(prefix);

// Throws the encoded tab at the end of the string away
```

```
// Tests/LibIMAP/TestQuotedPrintable.cpp
// The randomized test below found a deviation from the spec!

TEST_CASE(section_6_7_3_white_space_regressions)
{
    // Found by the randomized test below.

    // Throws the encoded tab/space at the end of the string away
    DECODE_EQUAL("!\t"sv,  "!"sv);
    DECODE_EQUAL("! "sv,   "!"sv);

    // Doesn't throw the encoded tab/space in the middle of the string away
    DECODE_EQUAL("!\t!"sv, "!\t!"sv);
    DECODE_EQUAL("! !"sv,  "! !"sv);
}
```



```
AK:
TestByteBuffer.cpp
TestChecked.cpp
TestCircularBuffer.cpp
TestCircularDeque.cpp
TestCircularQueue.cpp
TestComplex.cpp
TestDeprecatedString.cpp
TestDisjointChunks.cpp
TestDistinctNumeric.cpp
TestDoublyLinkedList.cpp
TestDuration.cpp
TestEnumBits.cpp
TestFind.cpp
TestFixedArray.cpp
TestFixedPoint.cpp
TestFloatingPoint.cpp
TestFloatingPointParsing.cpp
TestFlyString.cpp
TestFormat.cpp
TestFuzzyMatch.cpp
TestGenericLexer.cpp
TestHashFunctions.cpp
TestHashMap.cpp
TestHashTable.cpp
TestHex.cpp
TestIPv4Address.cpp
TestIPv6Address.cpp
TestIndexSequence.cpp
TestInsertionSort.cpp
TestIntegerMath.cpp
TestIntrusiveList.cpp
TestIntrusiveRedBlackTree.cpp
TestJSON.cpp
TestLEB128.cpp
TestLexicalPath.cpp
TestMACAddress.cpp
TestMemory.cpp
TestMemoryStream.cpp
```

```
TestNeverDestroyed.cpp
TestNonnullOwnPtr.cpp
TestNonnullRefPtr.cpp
TestNumberFormat.cpp
TestOptional.cpp
TestOwnPtr.cpp
TestPrint.cpp
TestQueue.cpp
TestQuickSelect.cpp
TestQuickSort.cpp
TestRedBlackTree.cpp
TestRefPtr.cpp
TestSIMD.cpp
TestSinglyLinkedList.cpp
TestSourceGenerator.cpp
TestSourceLocation.cpp
TestSpan.cpp
TestStack.cpp
TestStatistics.cpp
TestStdLibExtras.cpp
TestString.cpp
TestStringFloatingPointConversions.cpp
TestStringUtils.cpp
TestStringView.cpp
TestTrie.cpp
TestTuple.cpp
TestTypeTraits.cpp
TestTypedTransfer.cpp
TestUFixedBigInt.cpp
TestURL.cpp
TestUtf16.cpp
TestUtf8.cpp
TestVariant.cpp
TestVector.cpp
TestWeakPtr.cpp

Kernel:
TestEFault.cpp
TestEmptyPrivateInodeVMObject.cpp
```

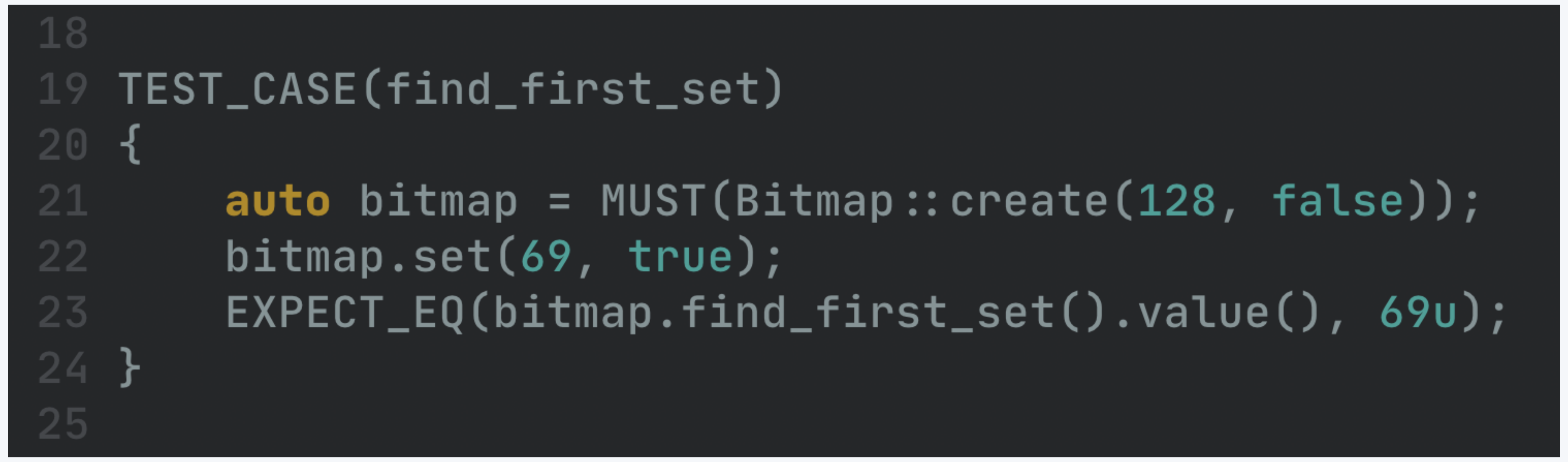
```
TestEmptyPrivateInodeVMObject.cpp
TestEmptySharedInodeVMObject.cpp
TestExt2FS.cpp
TestInvalidUIDSet.cpp
TestKernelAlarm.cpp
TestKernelFilePermissions.cpp
TestKernelPledge.cpp
TestKernelUnveil.cpp
TestMemoryDeviceMmap.cpp
TestMunMap.cpp
TestPosixFallocate.cpp
TestPrivateInodeVMObject.cpp
TestProcFS.cpp
TestProcFSWrite.cpp
TestSharedInodeVMObject.cpp
TestSigAltStack.cpp
TestSigHandler.cpp
TestSigWait.cpp

LibAudio:
TestFLACSpec.cpp
TestPlaybackStream.cpp

LibC:
TestAbort.cpp
TestAssert.cpp
TestCType.cpp
TestEnvironment.cpp
TestIo.cpp
TestLibCDirEnt.cpp
TestLibCExec.cpp
TestLibCInodeWatcher.cpp
TestLibCMkTemp.cpp
TestLibCNetdb.cpp
TestLibCSetjmp.cpp
TestLibCString.cpp
TestLibCTime.cpp
TestMalloc.cpp
TestMath.cpp
```

```
TestStrlcpy.cpp
TestStrtodAccuracy.cpp
TestWchar.cpp
TestWctype.cpp

LibCompress:
TestBrotli.cpp
TestDeflate.cpp
TestGzip.cpp
TestLzma.cpp
TestXz.cpp
TestZlib.cpp

LibCore:
TestLibCoreArgsParser.cpp
TestLibCoreDeferredInvoke.cp
TestLibCoreFilePermissionsMa
TestLibCoreFileWatcher.cpp
TestLibCoreMappedFile.cpp
TestLibCorePromise.cpp
TestLibCoreSharedSingleProdu
TestLibCoreStream.cpp

LibCpp:
test-cpp-parser.cpp
test-cpp-preprocessor.cpp

LibCrypto:
TestAES.cpp
TestASN1.cpp
TestBigInteger.cpp
TestChaCha20.cpp
TestChacha20Poly1305.cpp
TestChecksum.cpp
TestCurves.cpp
TestEd25519.cpp
TestHMAC.cpp
TestHash.cpp
TestPBKDF2.cpp
```

```
todo_tests.txt          todo_tests.txt          todo_tests.txt          N… todo_tests.txt
:se nowrap
```

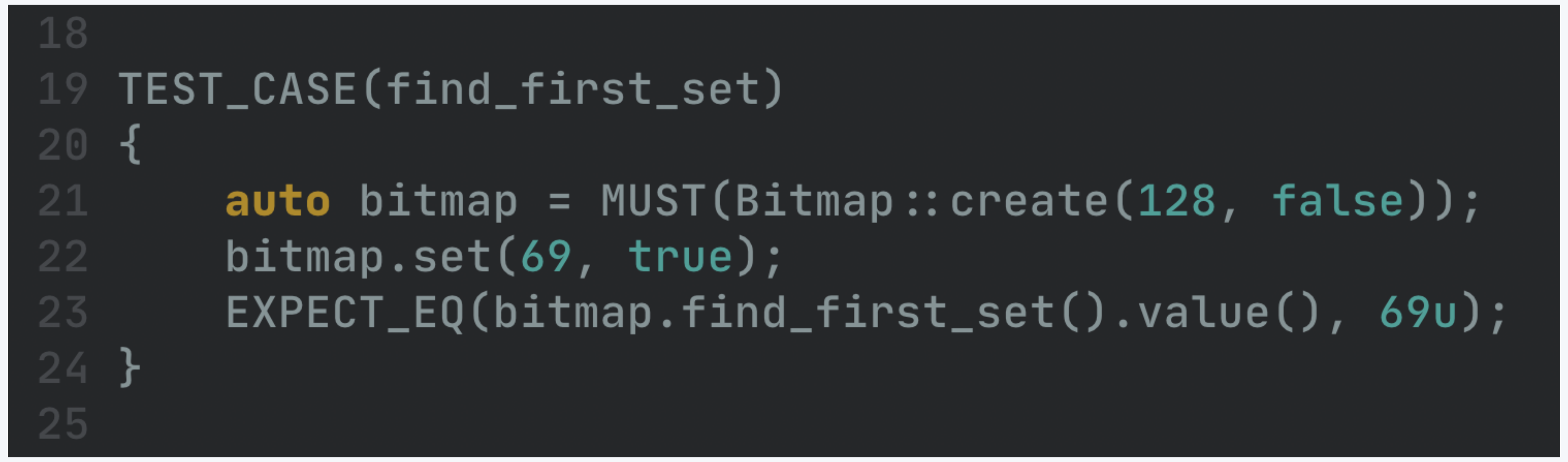
Bitmap

Bitmap

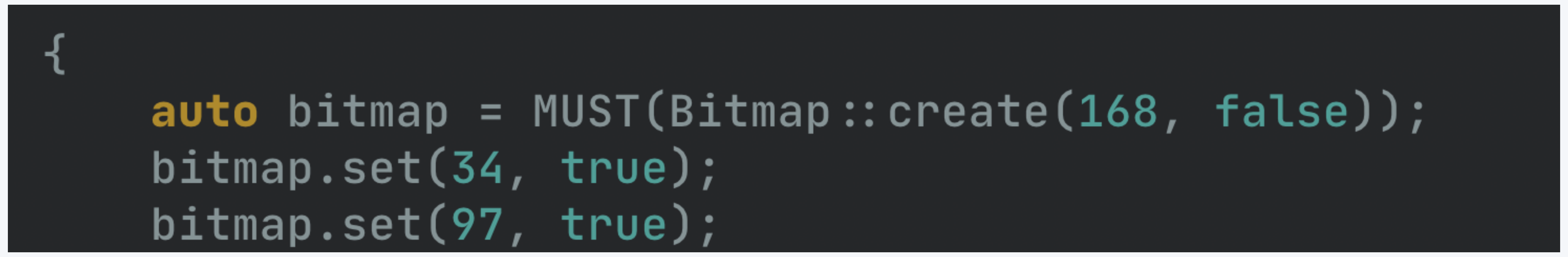


```

TEST_CASE(find_first_set)
{
    auto bitmap = MUST(Bitmap::create(128, false));
    bitmap.set(69, true);
    EXPECT_EQ(bitmap.find_first_set().value(), 69u);
}

```

Bitmap

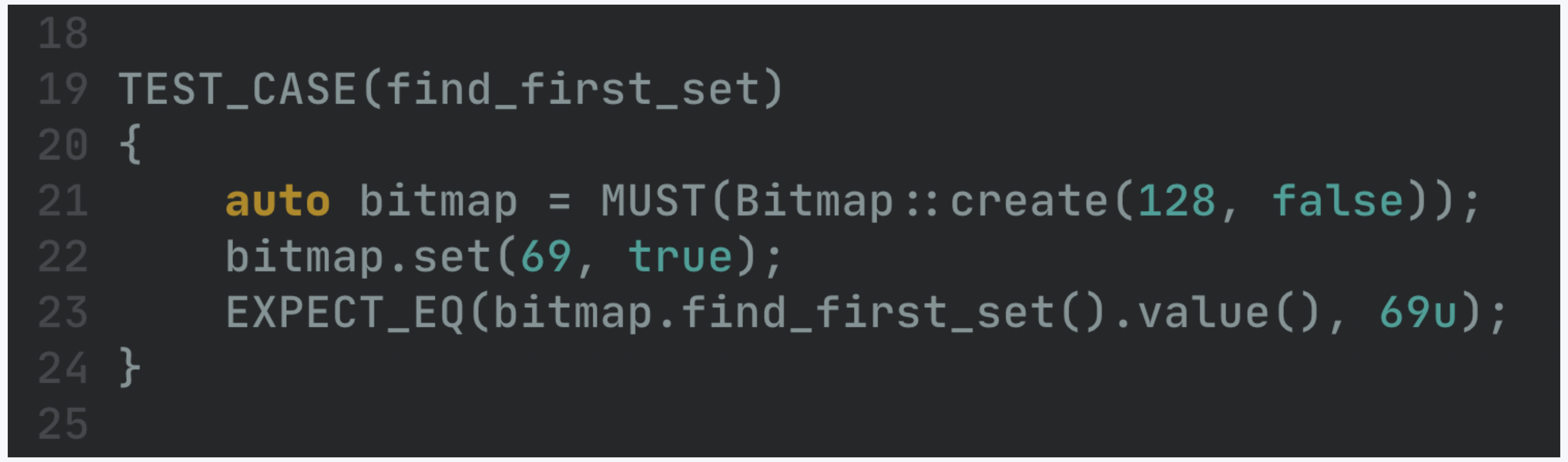


```

TEST_CASE(find_first_set)
{
    auto bitmap = MUST(Bitmap::create(128, false));
    bitmap.set(69, true);
    EXPECT_EQ(bitmap.find_first_set().value(), 69u);
}

```

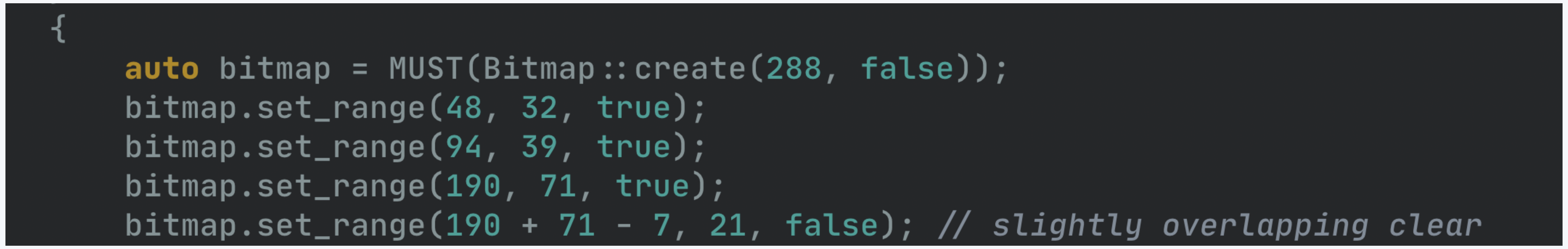


```
{
    auto bitmap = MUST(Bitmap::create(168, false));
    bitmap.set(34, true);
    bitmap.set(97, true);
```

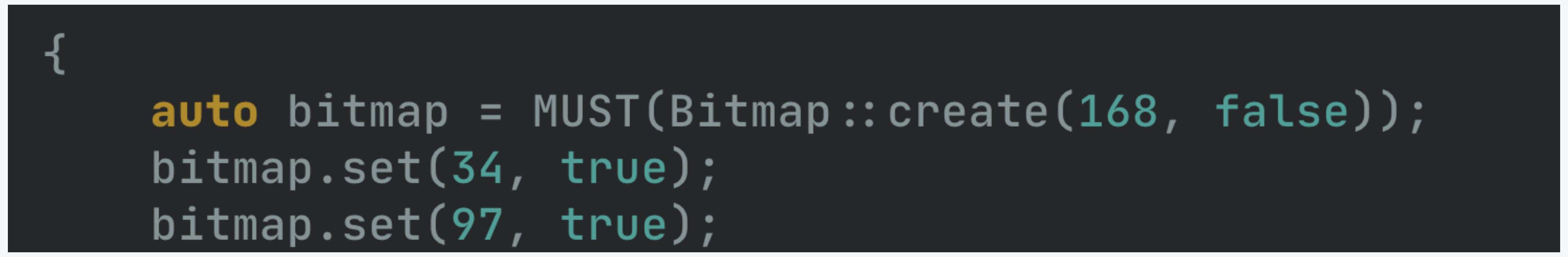
Bitmap



```
TEST_CASE(find_first_set)
{
    auto bitmap = MUST(Bitmap::create(128, false));
    bitmap.set(69, true);
    EXPECT_EQ(bitmap.find_first_set().value(), 69u);
}
```

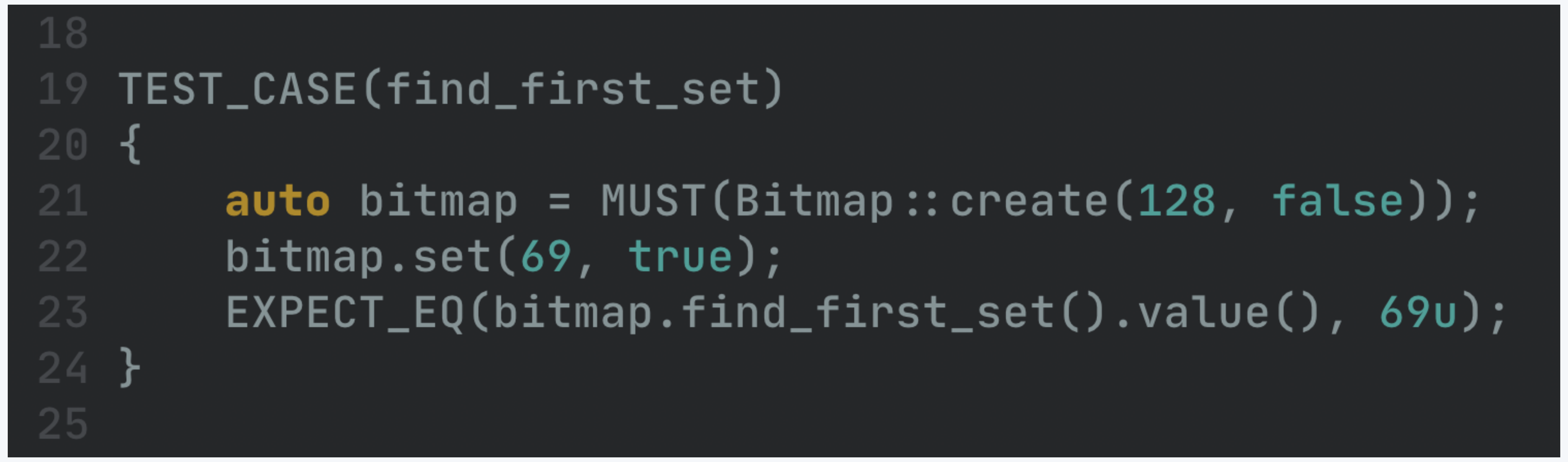


```
{
    auto bitmap = MUST(Bitmap::create(288, false));
    bitmap.set_range(48, 32, true);
    bitmap.set_range(94, 39, true);
    bitmap.set_range(190, 71, true);
    bitmap.set_range(190 + 71 - 7, 21, false); // slightly overlapping clear
```

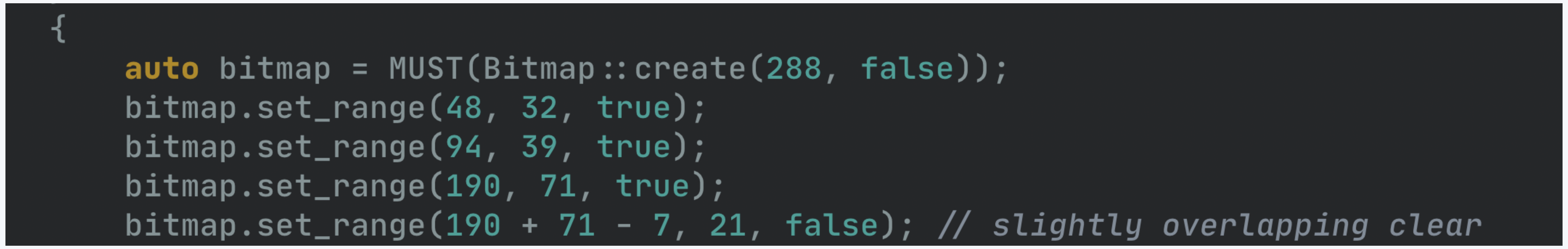


```
{
    auto bitmap = MUST(Bitmap::create(168, false));
    bitmap.set(34, true);
    bitmap.set(97, true);
```

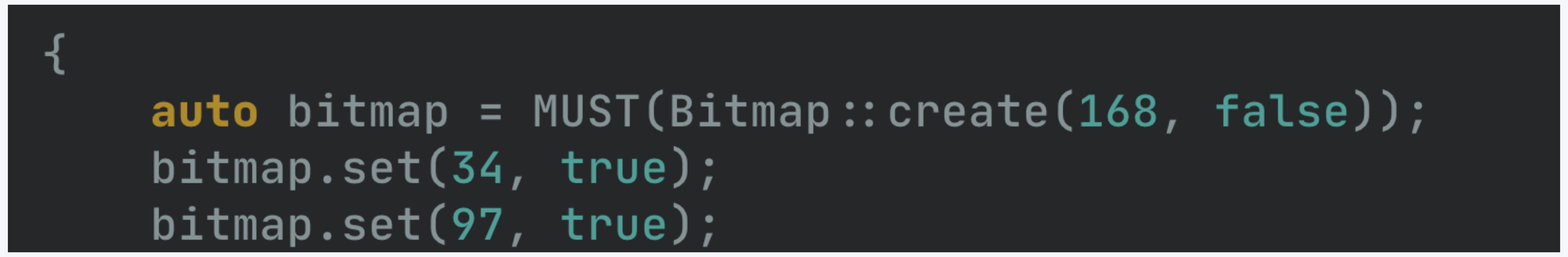
Bitmap



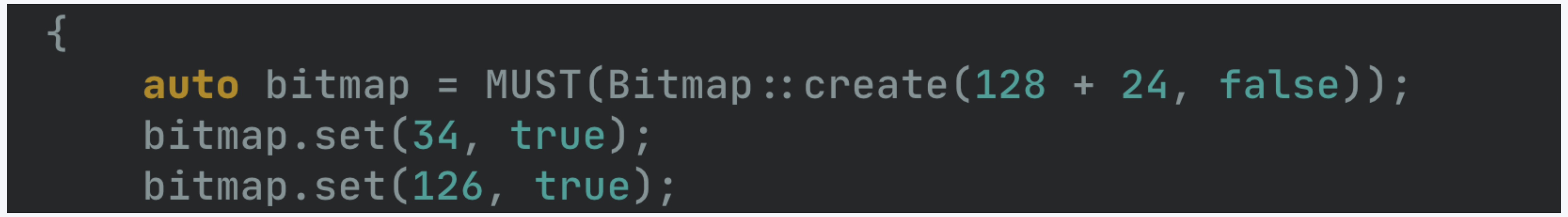
```
TEST_CASE(find_first_set)
{
    auto bitmap = MUST(Bitmap::create(128, false));
    bitmap.set(69, true);
    EXPECT_EQ(bitmap.find_first_set().value(), 69u);
}
```



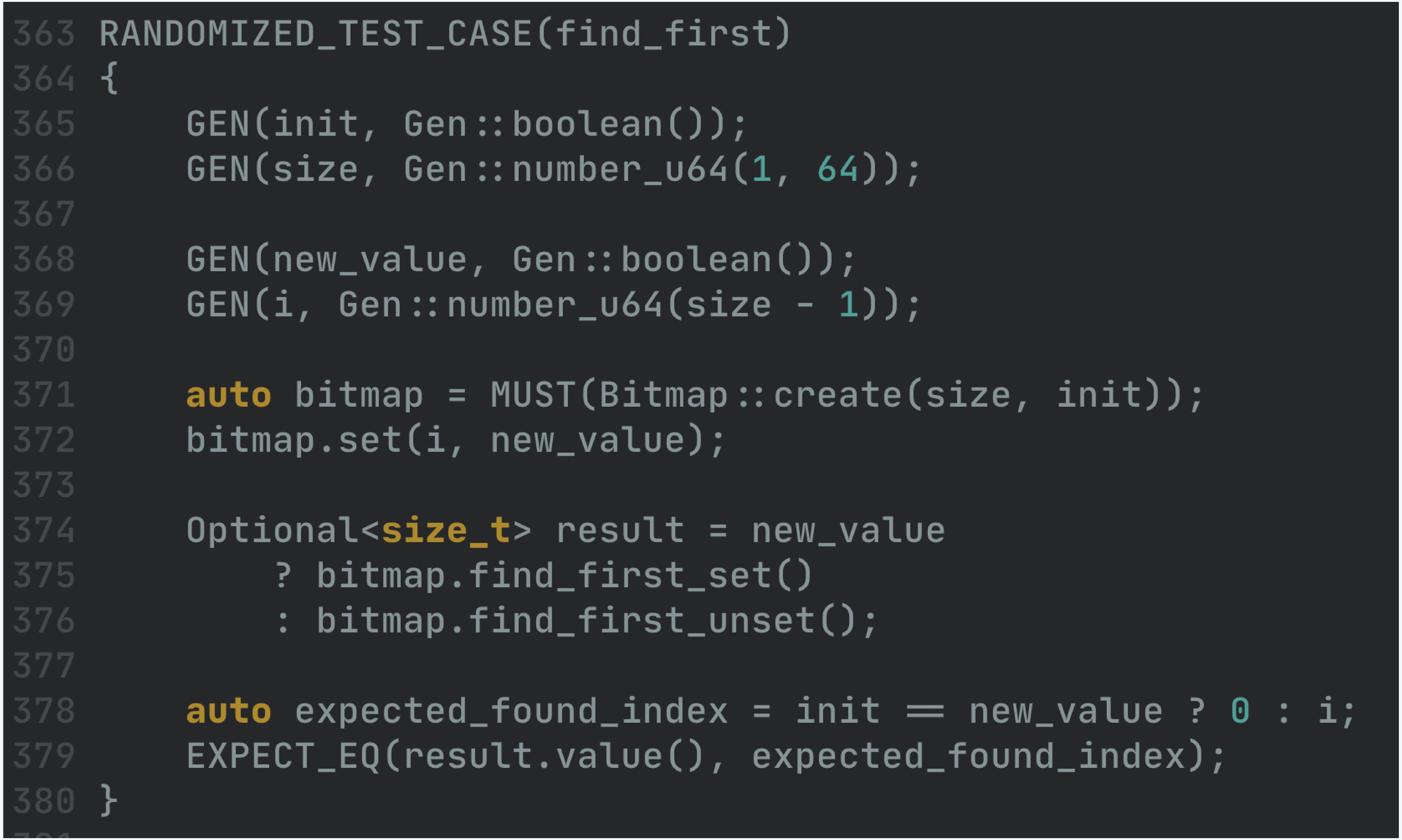
```
{
    auto bitmap = MUST(Bitmap::create(288, false));
    bitmap.set_range(48, 32, true);
    bitmap.set_range(94, 39, true);
    bitmap.set_range(190, 71, true);
    bitmap.set_range(190 + 71 - 7, 21, false); // slightly overlapping clear
```



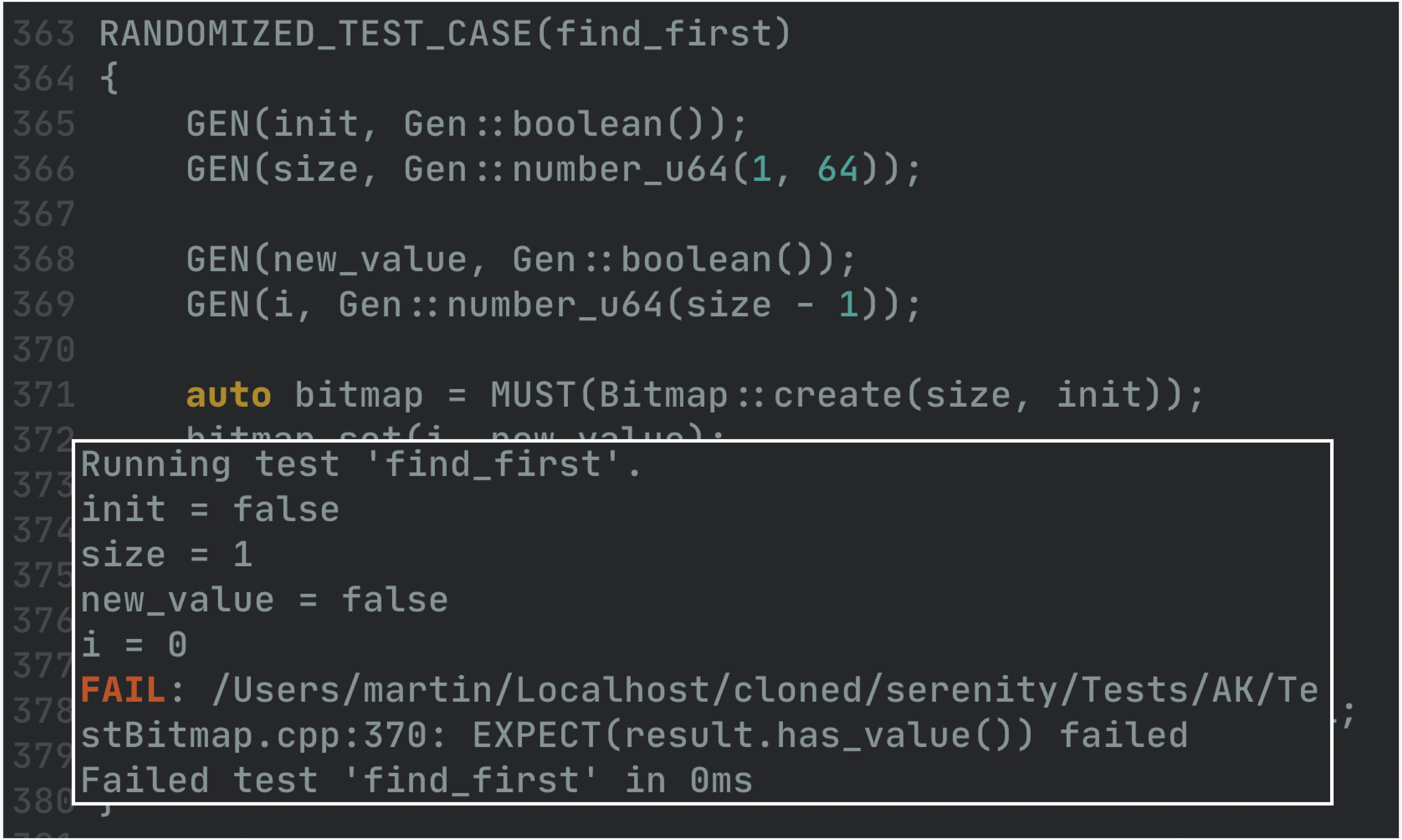
```
{
    auto bitmap = MUST(Bitmap::create(168, false));
    bitmap.set(34, true);
    bitmap.set(97, true);
```



```
{
    auto bitmap = MUST(Bitmap::create(128 + 24, false));
    bitmap.set(34, true);
    bitmap.set(126, true);
```

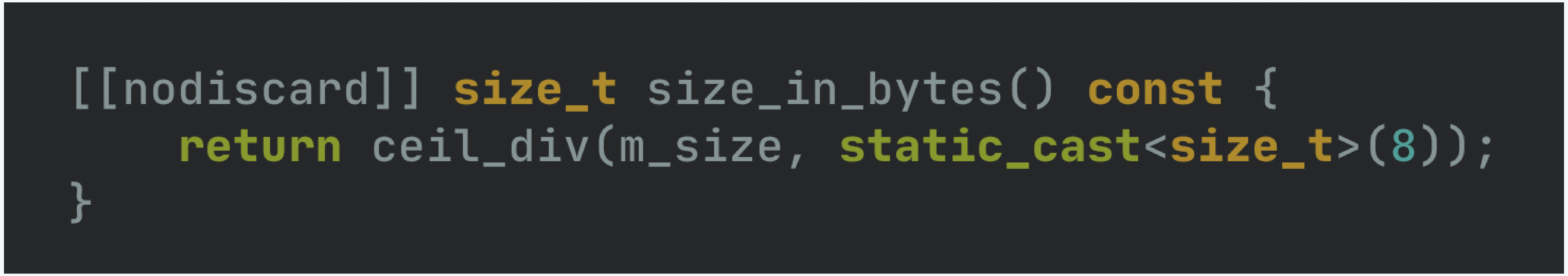


```
RANDOMIZED_TEST_CASE(find_first)
{
    GEN(init, Gen::boolean());
    GEN(size, Gen::number_u64(1, 64));

    GEN(new_value, Gen::boolean());
    GEN(i, Gen::number_u64(size - 1));

    auto bitmap = MUST(Bitmap::create(size, init));
    bitmap.set(i, new_value);

    Optional<size_t> result = new_value
        ? bitmap.find_first_set()
        : bitmap.find_first_unset();

    auto expected_found_index = init == new_value ? 0 : i;
    EXPECT_EQ(result.value(), expected_found_index);
}
```

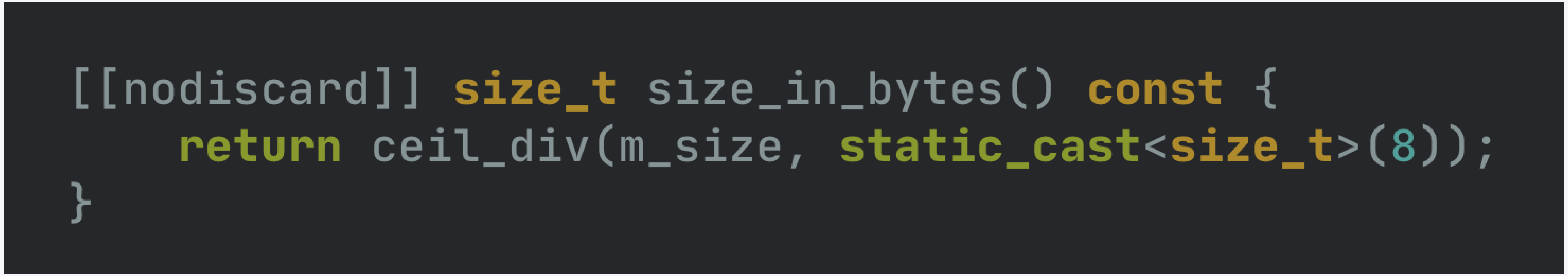


```
RANDOMIZED_TEST_CASE(find_first)
{
    GEN(init, Gen::boolean());
    GEN(size, Gen::number_u64(1, 64));

    GEN(new_value, Gen::boolean());
    GEN(i, Gen::number_u64(size - 1));

    auto bitmap = MUST(Bitmap::create(size, init));
    bitmap.set(i, new_value);
```

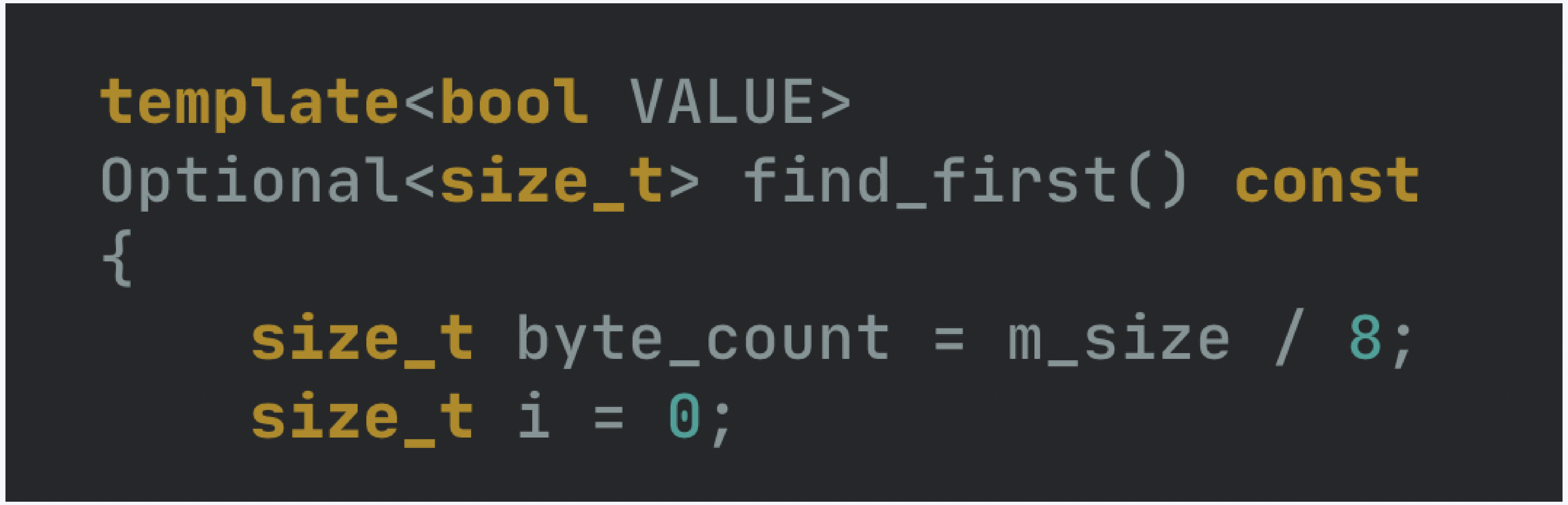
```
Running test 'find_first'.
init = false
size = 1
new_value = false
i = 0
FAIL: /Users/martin/Localhost/cloned/serenity/Tests/AK/Te
stBitmap.cpp:370: EXPECT(result.has_value()) failed
Failed test 'find_first' in 0ms
```



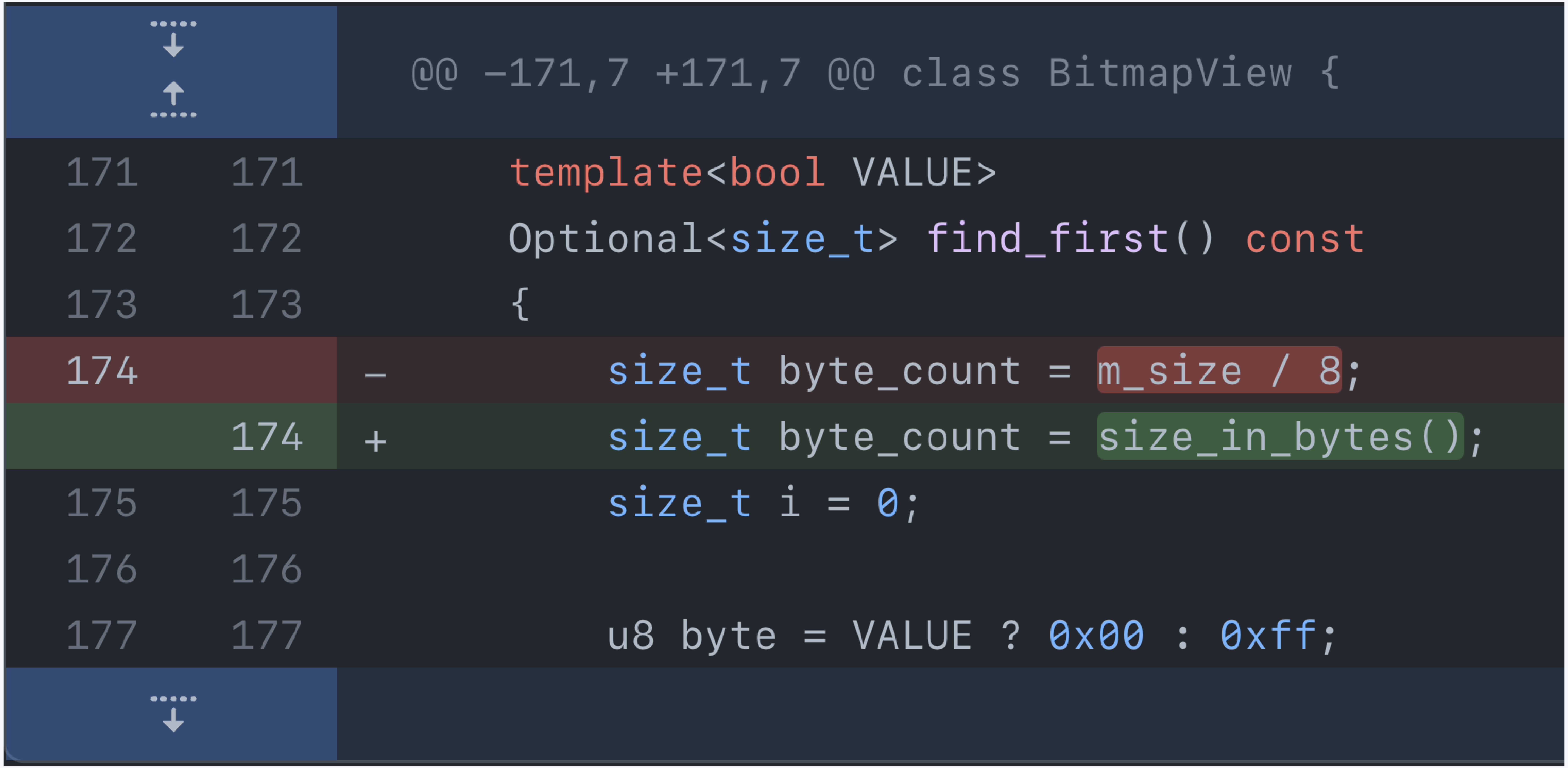
```cpp
[[nodiscard]] size_t size_in_bytes() const {
    return ceil_div(m_size, static_cast<size_t>(8));
}
```



```cpp
[[nodiscard]] size_t size_in_bytes() const {
    return ceil_div(m_size, static_cast<size_t>(8));
}
```



```cpp
template<bool VALUE>
Optional<size_t> find_first() const
{
    size_t byte_count = m_size / 8;
    size_t i = 0;
```



```
@@ -171,7 +171,7 @@ class BitmapView {
     template<bool VALUE>
     Optional<size_t> find_first() const
     {
-        size_t byte_count = m_size / 8;
+        size_t byte_count = size_in_bytes();
         size_t i = 0;

         u8 byte = VALUE ? 0x00 : 0xff;
```

Edit
<> Code
Merged
Conversation 0
Commits 1
Checks 15
Files changed 2
+21 −3
Contributor
Reviewers
Assignees
Labels
Projects

AK: Fix one-off error in BitmapView::find_first and find_one_anywhere #21409

timschumi merged 1 commit into `SerenityOS:master` from `Janiczek:fix-bitmap` on Oct 11, 2023

Janiczek commented on Oct 11, 2023 • edited

The mentioned functions used `m_size / 8` instead of `size_in_bytes()` (division with ceiling rounding mode), which resulted in an off-by-one error such that the functions didn't search in the last not-fully-8-bits byte.

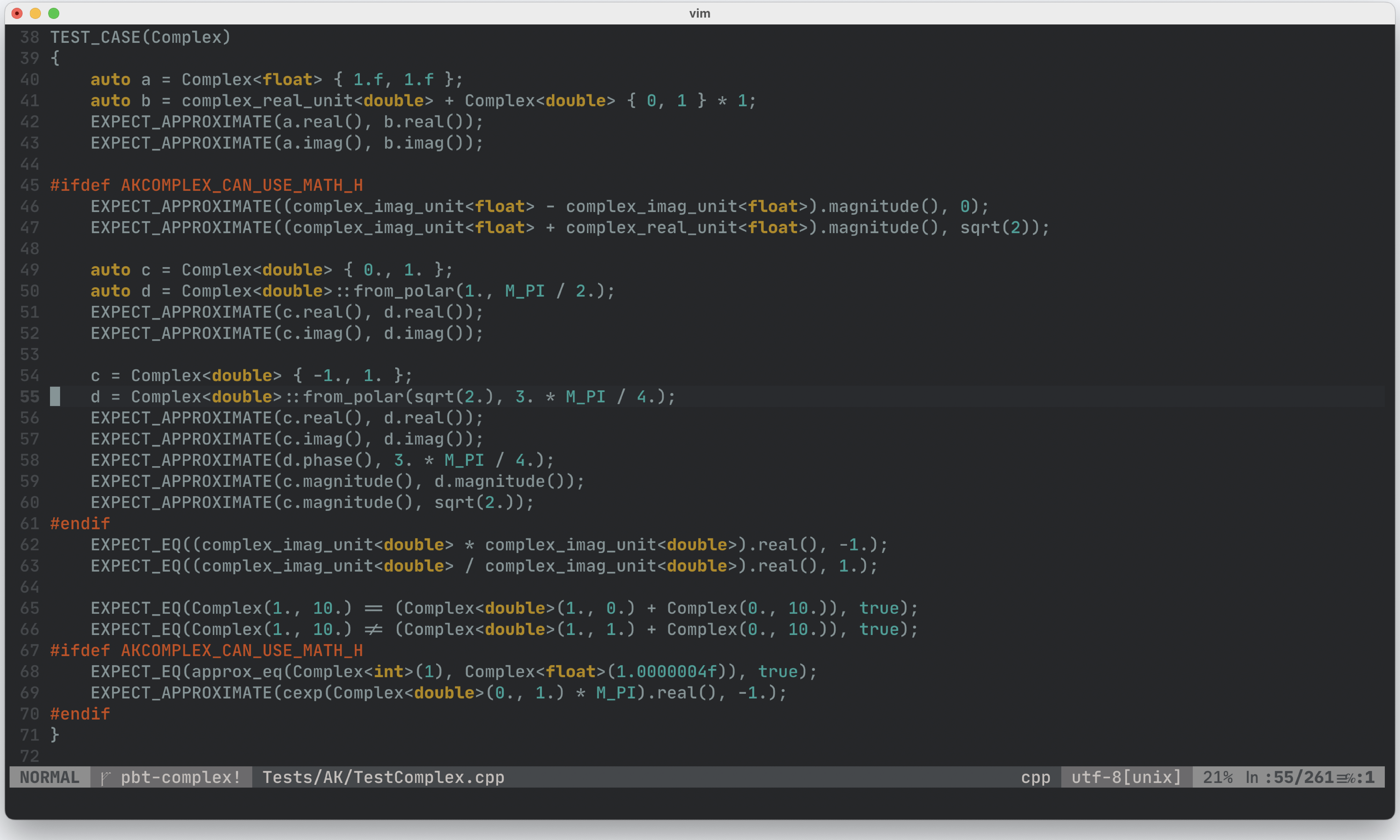
Using `size_in_bytes()` instead of `m_size / 8` fixes this.

> **Note**
>
> This was found using `RANDOMIZED_TEST_CASE` (PR ready for review). I'd appreciate reviews there as well, so that I could commit the randomized tests that found the issue as well!

Reviewers: gmta ✓, timschumi ✓

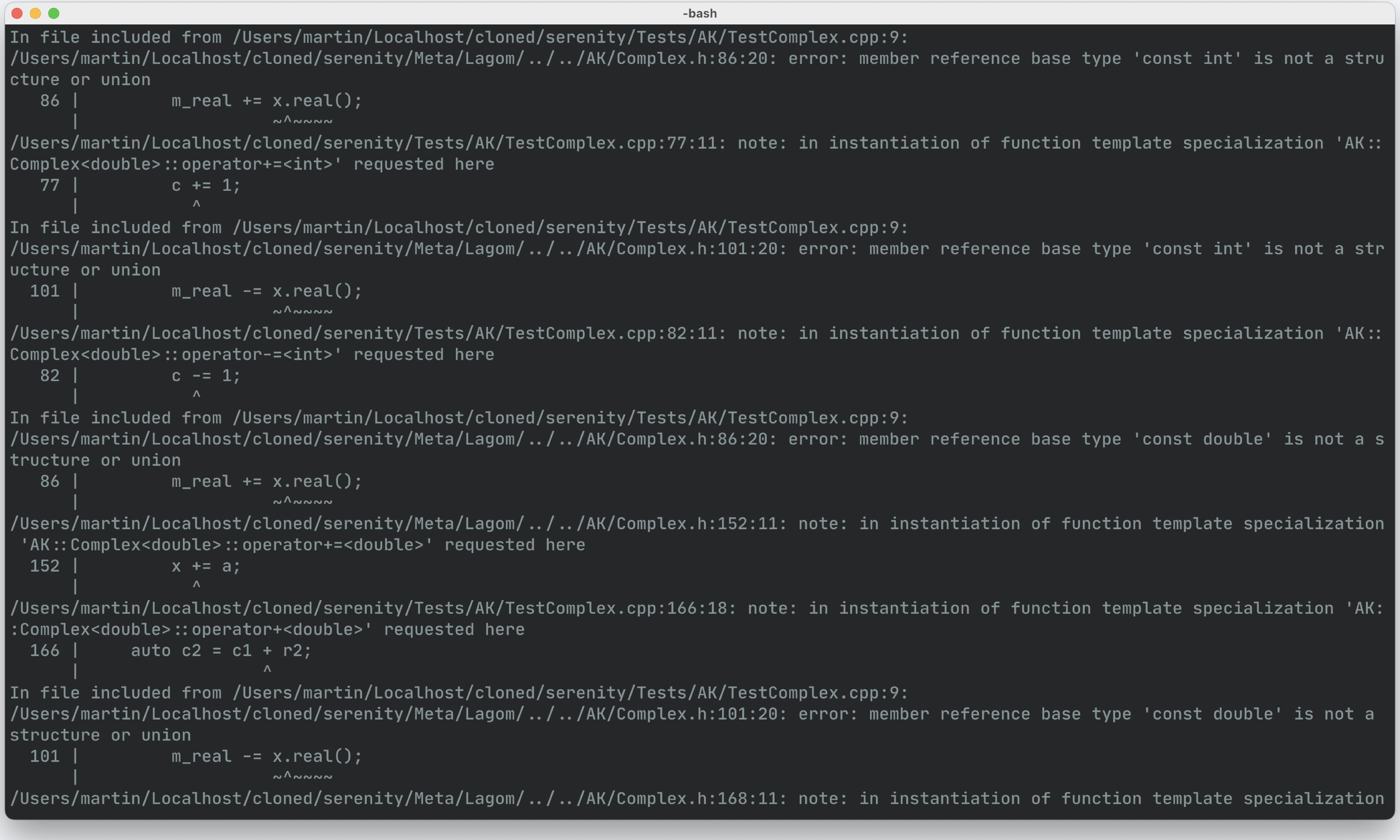
Assignees: No one assigned

Labels: None yet

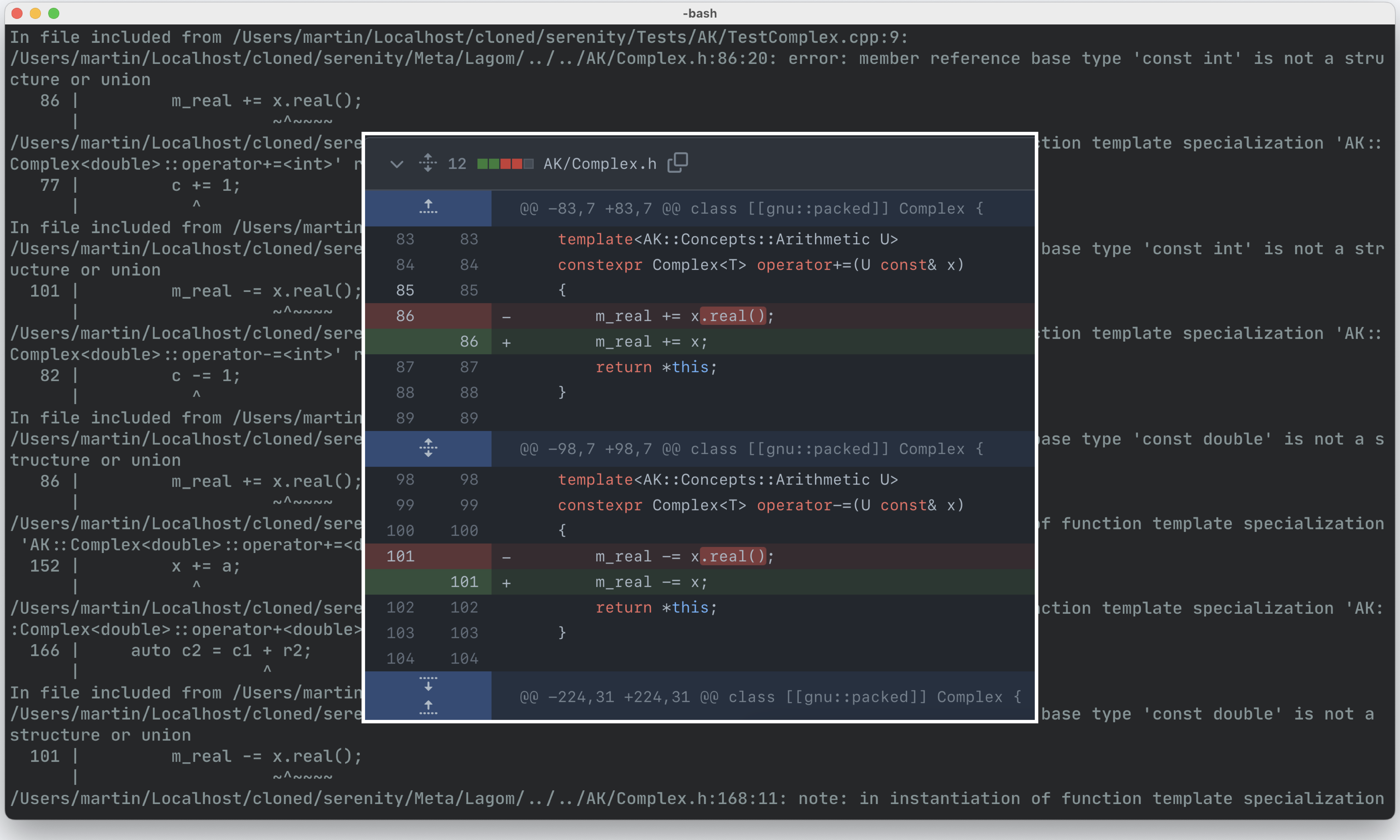


```cpp
TEST_CASE(Complex)
{
    auto a = Complex<float> { 1.f, 1.f };
    auto b = complex_real_unit<double> + Complex<double> { 0, 1 } * 1;
    EXPECT_APPROXIMATE(a.real(), b.real());
    EXPECT_APPROXIMATE(a.imag(), b.imag());

#ifdef AKCOMPLEX_CAN_USE_MATH_H
    EXPECT_APPROXIMATE((complex_imag_unit<float> - complex_imag_unit<float>).magnitude(), 0);
    EXPECT_APPROXIMATE((complex_imag_unit<float> + complex_real_unit<float>).magnitude(), sqrt(2));

    auto c = Complex<double> { 0., 1. };
    auto d = Complex<double>::from_polar(1., M_PI / 2.);
    EXPECT_APPROXIMATE(c.real(), d.real());
    EXPECT_APPROXIMATE(c.imag(), d.imag());

    c = Complex<double> { -1., 1. };
    d = Complex<double>::from_polar(sqrt(2.), 3. * M_PI / 4.);
    EXPECT_APPROXIMATE(c.real(), d.real());
    EXPECT_APPROXIMATE(c.imag(), d.imag());
    EXPECT_APPROXIMATE(d.phase(), 3. * M_PI / 4.);
    EXPECT_APPROXIMATE(c.magnitude(), d.magnitude());
    EXPECT_APPROXIMATE(c.magnitude(), sqrt(2.));
#endif
    EXPECT_EQ((complex_imag_unit<double> * complex_imag_unit<double>).real(), -1.);
    EXPECT_EQ((complex_imag_unit<double> / complex_imag_unit<double>).real(), 1.);

    EXPECT_EQ(Complex(1., 10.) == (Complex<double>(1., 0.) + Complex(0., 10.)), true);
    EXPECT_EQ(Complex(1., 10.) != (Complex<double>(1., 1.) + Complex(0., 10.)), true);
#ifdef AKCOMPLEX_CAN_USE_MATH_H
    EXPECT_EQ(approx_eq(Complex<int>(1), Complex<float>(1.0000004f)), true);
    EXPECT_APPROXIMATE(cexp(Complex<double>(0., 1.) * M_PI).real(), -1.);
#endif
}

```

```
NORMAL  pbt-complex!  Tests/AK/TestComplex.cpp                cpp  utf-8[unix]  21%  ln :55/261≡%:1
```



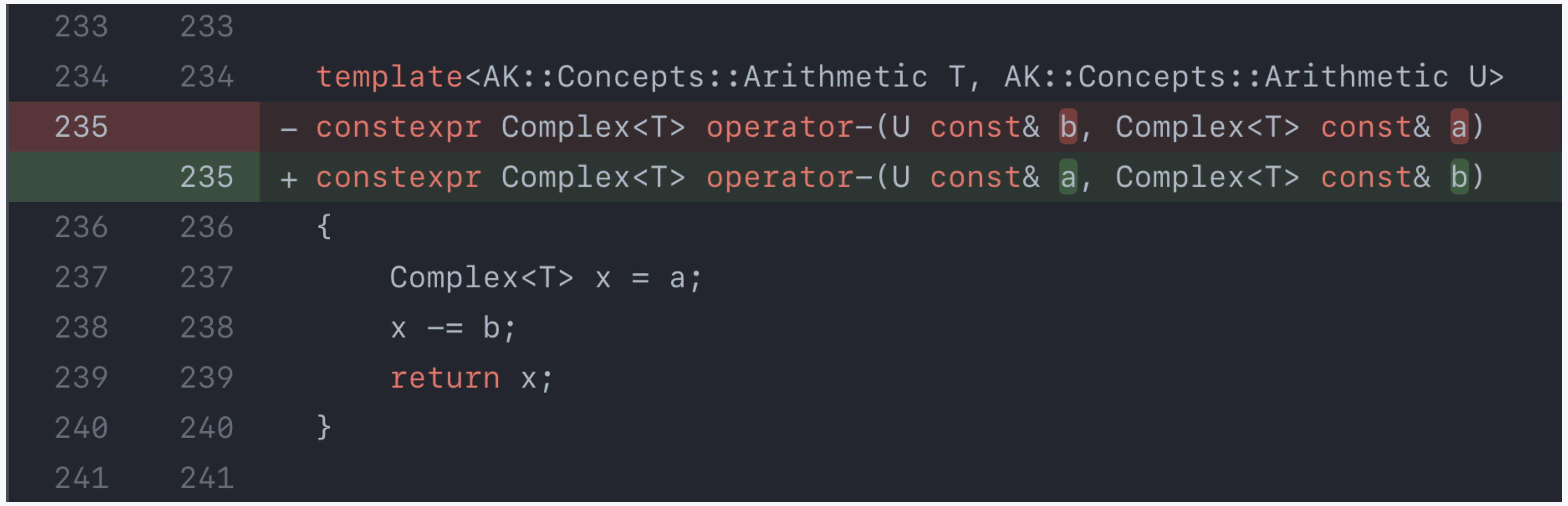
```
In file included from /Users/martin/Localhost/cloned/serenity/Tests/AK/TestComplex.cpp:9:
/Users/martin/Localhost/cloned/serenity/Meta/Lagom/../../AK/Complex.h:86:20: error: member reference base type 'const int' is not a stru
cture or union
   86 |         m_real += x.real();
      |                   ~^~~~~
/Users/martin/Localhost/cloned/serenity/Tests/AK/TestComplex.cpp:77:11: note: in instantiation of function template specialization 'AK::
Complex<double>::operator+=<int>' requested here
   77 |         c += 1;
      |           ^
In file included from /Users/martin/Localhost/cloned/serenity/Tests/AK/TestComplex.cpp:9:
/Users/martin/Localhost/cloned/serenity/Meta/Lagom/../../AK/Complex.h:101:20: error: member reference base type 'const int' is not a str
ucture or union
  101 |         m_real -= x.real();
      |                   ~^~~~~
/Users/martin/Localhost/cloned/serenity/Tests/AK/TestComplex.cpp:82:11: note: in instantiation of function template specialization 'AK::
Complex<double>::operator-=<int>' requested here
   82 |         c -= 1;
      |           ^
In file included from /Users/martin/Localhost/cloned/serenity/Tests/AK/TestComplex.cpp:9:
/Users/martin/Localhost/cloned/serenity/Meta/Lagom/../../AK/Complex.h:86:20: error: member reference base type 'const double' is not a s
tructure or union
   86 |         m_real += x.real();
      |                   ~^~~~~
/Users/martin/Localhost/cloned/serenity/Meta/Lagom/../../AK/Complex.h:152:11: note: in instantiation of function template specialization
 'AK::Complex<double>::operator+=<double>' requested here
  152 |         x += a;
      |           ^
/Users/martin/Localhost/cloned/serenity/Tests/AK/TestComplex.cpp:166:18: note: in instantiation of function template specialization 'AK:
:Complex<double>::operator+<double>' requested here
  166 |     auto c2 = c1 + r2;
      |                  ^
In file included from /Users/martin/Localhost/cloned/serenity/Tests/AK/TestComplex.cpp:9:
/Users/martin/Localhost/cloned/serenity/Meta/Lagom/../../AK/Complex.h:101:20: error: member reference base type 'const double' is not a
structure or union
  101 |         m_real -= x.real();
      |                   ~^~~~~
/Users/martin/Localhost/cloned/serenity/Meta/Lagom/../../AK/Complex.h:168:11: note: in instantiation of function template specialization
```



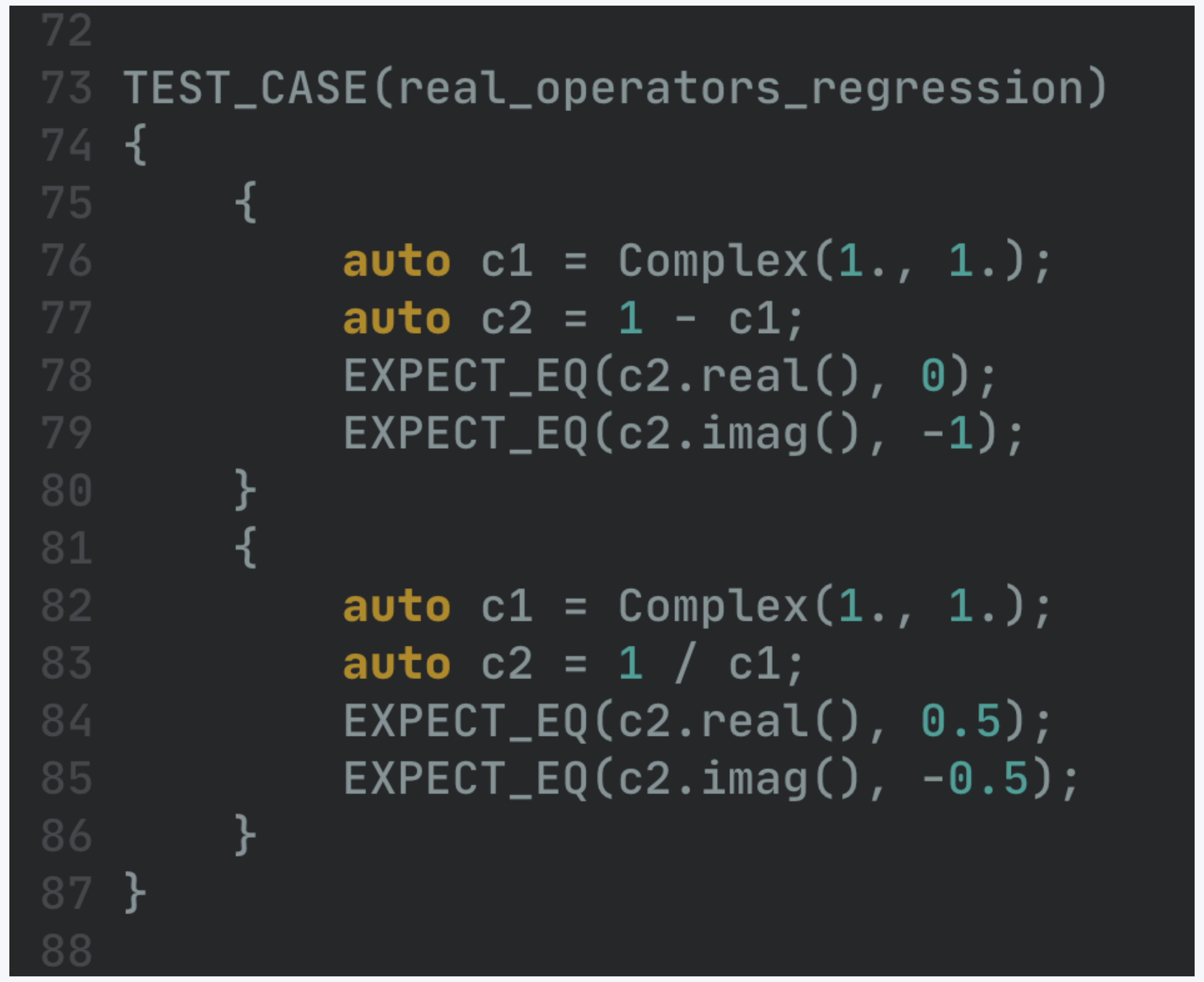
```
In file included from /Users/martin/Localhost/cloned/serenity/Tests/AK/TestComplex.cpp:9:
/Users/martin/Localhost/cloned/serenity/Meta/Lagom/../../AK/Complex.h:86:20: error: member reference base type 'const int' is not a stru
cture or union
   86 |         m_real += x.real();
      |                   ~^~~~~
/Users/martin/Localhost/cloned/sere                                                  tion template specialization 'AK::
Complex<double>::operator+=<int>' r
   77 |         c += 1;
      |           ^
In file included from /Users/martin
/Users/martin/Localhost/cloned/sere                                                  base type 'const int' is not a str
ucture or union
  101 |         m_real -= x.real();
      |                   ~^~~~~
/Users/martin/Localhost/cloned/sere                                                  tion template specialization 'AK::
Complex<double>::operator-=<int>' r
   82 |         c -= 1;
      |           ^
In file included from /Users/martin
/Users/martin/Localhost/cloned/sere                                                  ase type 'const double' is not a s
tructure or union
   86 |         m_real += x.real();
      |                   ~^~~~~
/Users/martin/Localhost/cloned/sere                                                  f function template specialization
 'AK::Complex<double>::operator+=<d
  152 |         x += a;
      |           ^
/Users/martin/Localhost/cloned/sere                                                  ction template specialization 'AK:
:Complex<double>::operator+<double>
  166 |     auto c2 = c1 + r2;
      |                  ^
In file included from /Users/martin
/Users/martin/Localhost/cloned/sere                                                  base type 'const double' is not a
structure or union
  101 |         m_real -= x.real();
      |                   ~^~~~~
/Users/martin/Localhost/cloned/serenity/Meta/Lagom/../../AK/Complex.h:168:11: note: in instantiation of function template specialization
```

12 AK/Complex.h

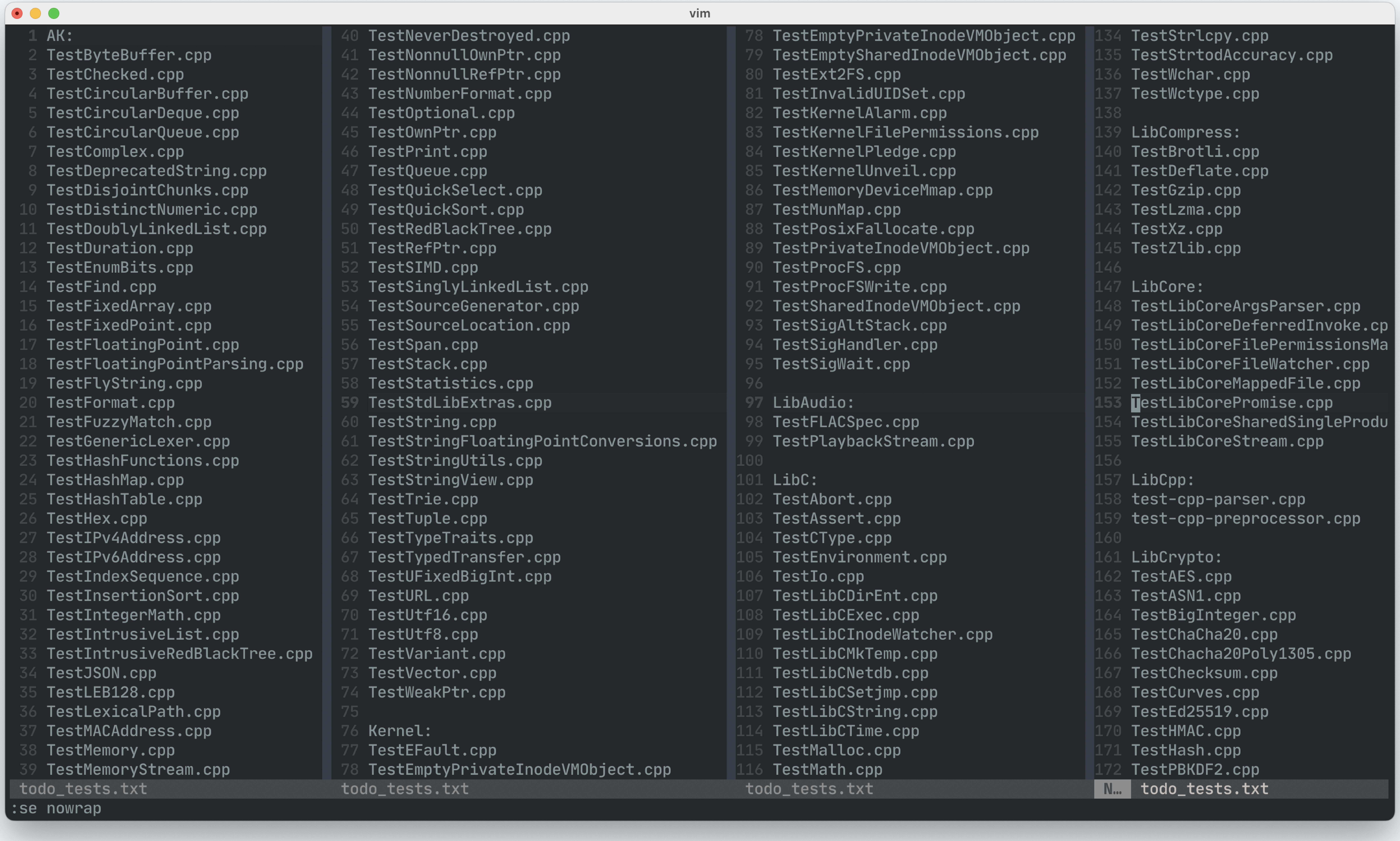
```diff
@@ -83,7 +83,7 @@ class [[gnu::packed]] Complex {
 83   83        template<AK::Concepts::Arithmetic U>
 84   84        constexpr Complex<T> operator+=(U const& x)
 85   85        {
 86      -         m_real += x.real();
      86 +         m_real += x;
 87   87            return *this;
 88   88        }
 89   89
@@ -98,7 +98,7 @@ class [[gnu::packed]] Complex {
 98   98        template<AK::Concepts::Arithmetic U>
 99   99        constexpr Complex<T> operator-=(U const& x)
100  100        {
101      -         m_real -= x.real();
     101 +         m_real -= x;
102  102            return *this;
103  103        }
104  104
@@ -224,31 +224,31 @@ class [[gnu::packed]] Complex {
```



```diff
  
  template<AK::Concepts::Arithmetic T, AK::Concepts::Arithmetic U>
- constexpr Complex<T> operator-(U const& b, Complex<T> const& a)
+ constexpr Complex<T> operator-(U const& a, Complex<T> const& b)
  {
      Complex<T> x = a;
      x -= b;
      return x;
  }
  
```



```cpp

TEST_CASE(real_operators_regression)
{
    {
        auto c1 = Complex(1., 1.);
        auto c2 = 1 - c1;
        EXPECT_EQ(c2.real(), 0);
        EXPECT_EQ(c2.imag(), -1);
    }
    {
        auto c1 = Complex(1., 1.);
        auto c2 = 1 / c1;
        EXPECT_EQ(c2.real(), 0.5);
        EXPECT_EQ(c2.imag(), -0.5);
    }
}

```



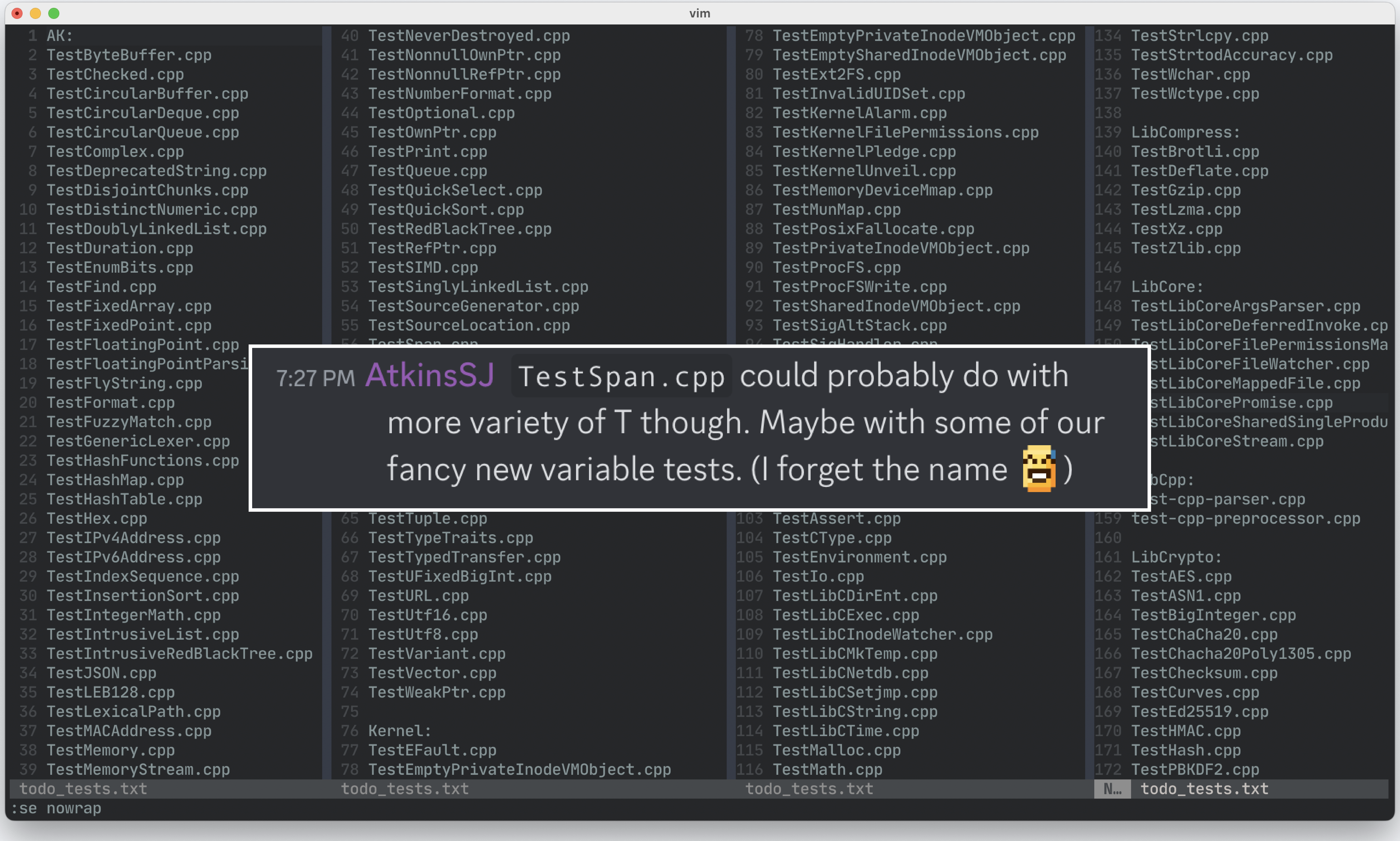
```
AK:
TestByteBuffer.cpp
TestChecked.cpp
TestCircularBuffer.cpp
TestCircularDeque.cpp
TestCircularQueue.cpp
TestComplex.cpp
TestDeprecatedString.cpp
TestDisjointChunks.cpp
TestDistinctNumeric.cpp
TestDoublyLinkedList.cpp
TestDuration.cpp
TestEnumBits.cpp
TestFind.cpp
TestFixedArray.cpp
TestFixedPoint.cpp
TestFloatingPoint.cpp
TestFloatingPointParsing.cpp
TestFlyString.cpp
TestFormat.cpp
TestFuzzyMatch.cpp
TestGenericLexer.cpp
TestHashFunctions.cpp
TestHashMap.cpp
TestHashTable.cpp
TestHex.cpp
TestIPv4Address.cpp
TestIPv6Address.cpp
TestIndexSequence.cpp
TestInsertionSort.cpp
TestIntegerMath.cpp
TestIntrusiveList.cpp
TestIntrusiveRedBlackTree.cpp
TestJSON.cpp
TestLEB128.cpp
TestLexicalPath.cpp
TestMACAddress.cpp
TestMemory.cpp
TestMemoryStream.cpp
todo_tests.txt
```

```
TestNeverDestroyed.cpp
TestNonnullOwnPtr.cpp
TestNonnullRefPtr.cpp
TestNumberFormat.cpp
TestOptional.cpp
TestOwnPtr.cpp
TestPrint.cpp
TestQueue.cpp
TestQuickSelect.cpp
TestQuickSort.cpp
TestRedBlackTree.cpp
TestRefPtr.cpp
TestSIMD.cpp
TestSinglyLinkedList.cpp
TestSourceGenerator.cpp
TestSourceLocation.cpp
TestSpan.cpp
TestStack.cpp
TestStatistics.cpp
TestStdLibExtras.cpp
TestString.cpp
TestStringFloatingPointConversions.cpp
TestStringUtils.cpp
TestStringView.cpp
TestTrie.cpp
TestTuple.cpp
TestTypeTraits.cpp
TestTypedTransfer.cpp
TestUFixedBigInt.cpp
TestURL.cpp
TestUtf16.cpp
TestUtf8.cpp
TestVariant.cpp
TestVector.cpp
TestWeakPtr.cpp

Kernel:
TestEFault.cpp
TestEmptyPrivateInodeVMObject.cpp
todo_tests.txt
```

```
TestEmptyPrivateInodeVMObject.cpp
TestEmptySharedInodeVMObject.cpp
TestExt2FS.cpp
TestInvalidUIDSet.cpp
TestKernelAlarm.cpp
TestKernelFilePermissions.cpp
TestKernelPledge.cpp
TestKernelUnveil.cpp
TestMemoryDeviceMmap.cpp
TestMunMap.cpp
TestPosixFallocate.cpp
TestPrivateInodeVMObject.cpp
TestProcFS.cpp
TestProcFSWrite.cpp
TestSharedInodeVMObject.cpp
TestSigAltStack.cpp
TestSigHandler.cpp
TestSigWait.cpp

LibAudio:
TestFLACSpec.cpp
TestPlaybackStream.cpp

LibC:
TestAbort.cpp
TestAssert.cpp
TestCType.cpp
TestEnvironment.cpp
TestIo.cpp
TestLibCDirEnt.cpp
TestLibCExec.cpp
TestLibCInodeWatcher.cpp
TestLibCMkTemp.cpp
TestLibCNetdb.cpp
TestLibCSetjmp.cpp
TestLibCString.cpp
TestLibCTime.cpp
TestMalloc.cpp
TestMath.cpp
todo_tests.txt
```

```
TestStrlcpy.cpp
TestStrtodAccuracy.cpp
TestWchar.cpp
TestWctype.cpp

LibCompress:
TestBrotli.cpp
TestDeflate.cpp
TestGzip.cpp
TestLzma.cpp
TestXz.cpp
TestZlib.cpp

LibCore:
TestLibCoreArgsParser.cpp
TestLibCoreDeferredInvoke.cp
TestLibCoreFilePermissionsMa
TestLibCoreFileWatcher.cpp
TestLibCoreMappedFile.cpp
TestLibCorePromise.cpp
TestLibCoreSharedSingleProdu
TestLibCoreStream.cpp

LibCpp:
test-cpp-parser.cpp
test-cpp-preprocessor.cpp

LibCrypto:
TestAES.cpp
TestASN1.cpp
TestBigInteger.cpp
TestChaCha20.cpp
TestChacha20Poly1305.cpp
TestChecksum.cpp
TestCurves.cpp
TestEd25519.cpp
TestHMAC.cpp
TestHash.cpp
TestPBKDF2.cpp
N… todo_tests.txt
```

```
:se nowrap
```



```
AK:
TestByteBuffer.cpp
TestChecked.cpp
TestCircularBuffer.cpp
TestCircularDeque.cpp
TestCircularQueue.cpp
TestComplex.cpp
TestDeprecatedString.cpp
TestDisjointChunks.cpp
TestDistinctNumeric.cpp
TestDoublyLinkedList.cpp
TestDuration.cpp
TestEnumBits.cpp
TestFind.cpp
TestFixedArray.cpp
TestFixedPoint.cpp
TestFloatingPoint.cpp
TestFloatingPointParsi
TestFlyString.cpp
TestFormat.cpp
TestFuzzyMatch.cpp
TestGenericLexer.cpp
TestHashFunctions.cpp
TestHashMap.cpp
TestHashTable.cpp
TestHex.cpp
TestIPv4Address.cpp
TestIPv6Address.cpp
TestIndexSequence.cpp
TestInsertionSort.cpp
TestIntegerMath.cpp
TestIntrusiveList.cpp
TestIntrusiveRedBlackTree.cpp
TestJSON.cpp
TestLEB128.cpp
TestLexicalPath.cpp
TestMACAddress.cpp
TestMemory.cpp
TestMemoryStream.cpp
```

```
TestNeverDestroyed.cpp
TestNonnullOwnPtr.cpp
TestNonnullRefPtr.cpp
TestNumberFormat.cpp
TestOptional.cpp
TestOwnPtr.cpp
TestPrint.cpp
TestQueue.cpp
TestQuickSelect.cpp
TestQuickSort.cpp
TestRedBlackTree.cpp
TestRefPtr.cpp
TestSIMD.cpp
TestSinglyLinkedList.cpp
TestSourceGenerator.cpp
TestSourceLocation.cpp
TestSpan.cpp
TestTuple.cpp
TestTypeTraits.cpp
TestTypedTransfer.cpp
TestUFixedBigInt.cpp
TestURL.cpp
TestUtf16.cpp
TestUtf8.cpp
TestVariant.cpp
TestVector.cpp
TestWeakPtr.cpp

Kernel:
TestEFault.cpp
TestEmptyPrivateInodeVMObject.cpp
```

```
TestEmptyPrivateInodeVMObject.cpp
TestEmptySharedInodeVMObject.cpp
TestExt2FS.cpp
TestInvalidUIDSet.cpp
TestKernelAlarm.cpp
TestKernelFilePermissions.cpp
TestKernelPledge.cpp
TestKernelUnveil.cpp
TestMemoryDeviceMmap.cpp
TestMunMap.cpp
TestPosixFallocate.cpp
TestPrivateInodeVMObject.cpp
TestProcFS.cpp
TestProcFSWrite.cpp
TestSharedInodeVMObject.cpp
TestSigAltStack.cpp
TestSigHandler.cpp
TestAssert.cpp
TestCType.cpp
TestEnvironment.cpp
TestIo.cpp
TestLibCDirEnt.cpp
TestLibCExec.cpp
TestLibCInodeWatcher.cpp
TestLibCMkTemp.cpp
TestLibCNetdb.cpp
TestLibCSetjmp.cpp
TestLibCString.cpp
TestLibCTime.cpp
TestMalloc.cpp
TestMath.cpp
```

```
TestStrlcpy.cpp
TestStrtodAccuracy.cpp
TestWchar.cpp
TestWctype.cpp

LibCompress:
TestBrotli.cpp
TestDeflate.cpp
TestGzip.cpp
TestLzma.cpp
TestXz.cpp
TestZlib.cpp

LibCore:
TestLibCoreArgsParser.cpp
TestLibCoreDeferredInvoke.cp
TestLibCoreFilePermissionsMa
stLibCoreFileWatcher.cpp
stLibCoreMappedFile.cpp
stLibCorePromise.cpp
stLibCoreSharedSingleProdu
stLibCoreStream.cpp

bCpp:
st-cpp-parser.cpp
test-cpp-preprocessor.cpp

LibCrypto:
TestAES.cpp
TestASN1.cpp
TestBigInteger.cpp
TestChaCha20.cpp
TestChacha20Poly1305.cpp
TestChecksum.cpp
TestCurves.cpp
TestEd25519.cpp
TestHMAC.cpp
TestHash.cpp
TestPBKDF2.cpp
```

```
todo_tests.txt    todo_tests.txt    todo_tests.txt    N… todo_tests.txt
:se nowrap
```

7:27 PM **AtkinsSJ** `TestSpan.cpp` could probably do with more variety of T though. Maybe with some of our fancy new variable tests. (I forget the name 😅)

Summary

Summary

- Property tests are great!

Summary

- Property tests are great!
- For the love of God, steal from Hypothesis

Summary

- Property tests are great!
- For the love of God, steal from Hypothesis
- Functional C++ is hard; don’t go against the grain

Summary

- Property tests are great!
- For the love of God, steal from Hypothesis
- Functional C++ is hard; don’t go against the grain
- SerenityOS+PBT is a learning opportunity

Thank you!

@janiczek