Crash with empty parameter pack on variadic concept as return type constraint #48617

Closed **llvmbot** opened this issue on 29 Dec 2020 · 7 comments

llvmbot commented on 29 Dec 2020

Bugzilla Link	48617
Resolution	FIXED
Resolved on	Oct 11, 2021 20:29
Version	trunk
OS	Linux
Blocks	#52147
Reporter	LLVM Bugzilla Contributor
CC	@LebedevRI,@mizvekov,@zygoloid,@tstellar
Fixed by commit(s)	68b9d8ed7abe4046992ae1557990edfbb3a772bc 02dece03f93d

Assignees

mizvekov

Labels

bugzilla c++20

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

Extended Description

https://godbolt.org/z/6KE9ed

If a constraint on a return type ends up being given zero parameters for its parameter pack, clang blows up with a very confusing crash and no useful diagnostic. In addition, in certain situations if you explicitly specialize the function template before calling it again without explicit template arguments, the second inferred call will compile but only if preceded by the first call with explicit args. Unfortunately I do not have a minimal repro of this case yet but it is the way I initially encountered this bug.

2 participants

llvmbot commented on 29 Dec 2020 Author

assigned to **@mizvekov**

llvmbot commented on 29 Dec 2020 Author

Nevermind about the comment about explicit template args; that has to do with things relating to overloads and conversions in the specific code I was working with that would take a lot more code to demonstrate here and aren't core to the crash at hand.

tstellar commented on 22 Feb

This bug was not resolved in time for the 12.0.0 release, so it will have to wait for 12.0.1.
If you feel this is a high-priority bug that should be fixed for 12.0.0, please re-add release-12.0.0 to the Blocks field and add a comment explaining why this is high-priority.

zygoloid mannequin commented on 8 Sep

The behavior has nearly reversed since the 12 release. Now:

The lines labeled `// 3...` and `// 2...` both crash (with different backtraces!) and the lines labeled `// 1...` and `// BANG!` both work properly.

mizvekov commented on 8 Sep

Reduced example, showing some extra problems:

```
template <typename...> class a {};
template <typename...> concept g = true;
template <typename... d> g<d...> auto e(a<d...>) { return 0; }

#if   BUG == 0

  template g<int> auto e<int>(a<int>); // Clang fails to instantiate this

#elif BUG == 1

  template g<> auto e<>(a<>); // Clang crashes

#elif BUG == 2

  // Same as above, but it instantiates 'e' when it shouldn't.
  template auto e<>(a<>);

#elif BUG == 3

  // Get around BUG 0, since we can't just instantiate this.
  // Clang has a different crash this time.
  template <typename... b> a<b...> c(b...);
  int f = e(c(1, 2));

#endif
```

CE Workspace: https://godbolt.org/z/zK7E9nMfb

tstellar commented on 9 Sep

Merged: 02dece03f93d

tstellar commented 5 days ago

mentioned in issue #52147

tstellar mentioned this issue on 12 Oct

[meta] 13.0.1 Release Blockers #52147 Open

This issue was **closed**.