

Title

P1112

Pal Balog presenting

Erich: I would like to see a default layout attribute.

Adrian: How does this affect initialization order? Pal: This only changes layout order; initialization order stays the same.

Adrian: Is there a performance impact to that? Pal: Yes; this can affect cache performance.

Adrien: How does this interact with bit-fields? Pal: Bit-fields are treated as a containing type. All adjacent bit-fields are counted as one entity.

Andrzej: Is the compiler allowed to ignore these attributes? Pal: Yes. JF: You can't really ignore the attribute if you have ABI constraints.

JF: To clarify, this doesn't give us `#pragma pack`. Nothing can break alignment requirements.

Attila: Do you plan to have a layout that eliminates false sharing between members? JF: You already have that: `hardware_interference_size`. Pal: It's possible to do that as an extension. The framework aims to allow these sorts of things.

Andrzej: Is the purpose of `declorder` to mandate standard layout? Pal: Precisely. This ensures that layout order matches declaration order regardless of access specifiers.

Clarifying question on ABI stability Pal: "best" is not ABI-stable across compiler versions, but "smallest" should be.

Second Discussion on 17th July 2019

Agustin Berge (AB)

Erich Keane (EK)

Peter Gottschling (PG)

James Touton (JT)

Tobias Loew (TL)

Mihail Mihailove (MM)

Andrzej Krzemienski (AK)

Pal Balog (PB)

Corentin Brauge (CB)

Bruno Lopez (BL)

Corentin Jabot (CJ)

Martin (M)

Michael Park (MP)

Matti (MR)

Jan (JT)

Hana Dusikova (HD)

Sophia (SP)

discussion for P1112r1

PB: describes the paper

BG: Is this specified in the standard?

PB: Yes, it is specified in the standard.

AB: The order of member isn't observable less than operator. Is this going to change that?

JF: You can use offset as well, but you are only allowed to do on trivial layout.

JT: I want to `static_assert` that the layout matches some layout in the binary file.

PB: But if you are interested in that, then you are applying for a strategy that keeps that use case. So, you can do that with this.

JT: As we talked about before, we want smallest to be ABI stable. So, I want to make sure that the property is retained across compiler version/language standard.

PB: That is up to the strategy. There are ideas on how to make the different strategies stable to keep it stable.

HD: I think this is useful for many cases in my work. What about making this strategy in library using say a `constexpr` function? Did you think about using custom provided strategy?

JF: Like through reflection. Like "here's my function to order them."

PB: No I didn't think, because we don't have facilities. but if we have them, then I will be interested.

JF: What Hana is saying that if you give me the ability to pass a function which does the following thing. and then in standard library, add a version of that mechanism that makes it the smallest. And for custom cases there can be user defined function.

HD: Yes

PB: I'm not against that idea. but I like incremental change.

JF: but if you had a general capability as she proposed, and then this is a special case of that.

HD: Instead of giving a list of properties, we can have the name of the `constexpr` function.

SL: Should we consider this struct is POD or not? Question is motivated by C compatibility?

PB: `declorder` should keep standard layout and we propose wording there to make core standard layout as one that keeps the `declorder`. So we change wording of standard layout to mean that.

JF, SL, PB: discussion of standard layout in standard and freedom to do that.

PB: describes the paper

AK: If you do `declorder`, you would have no need for `declorder`, but only for layout smallest. You would still have the problem of attribute.

PB: It is compatible with attribute policy.

AK: I find it surprising.

PB: describes the polls we want to take

MR: There are probably several equally small ordering of the smallest?

JF: I don't want to dive into those details. That question is moot if instead we adopt Hana's approach.

JT: I think this whole thing is subsumed by metaclasses.

MP: Compilers have pack attributes. Are you standardizing existing practice or is this new stuff?

JF: Pragma pack is not in scope for this because it changes alignment.

MP: Is this something you currently writing in your code?

PB: No. I am not aware of this code.

Poll: Spend committee time on this? This being someday to do layout?

SF	F	N	A	SA
0	9	8	0	0

PB: Does reflection gave me the layout which is required here? HD: Yes. PB: Isn't that circular?

JF: If we are voting in this favor, then we want SG7 to look at this. They might say, this makes no sense.

PB: If they say its not doable, then I am not longer blocked.

JF: Yes.

M: This proposal introduce significant changes in C++ since it does not preserve the order of members in definition to memory layout.

JF: His proposal does not change that semantics. Order of construction is still preserved.

M: Do we want to preserve it?

JF: We can discuss that when it comes back from reflection.

TL: If you have base types also being subject to this in a recursive.

MP: We have to discuss this anyway.

MM: The question seems to specific, can we investigate a general approach?

Poll: Investigate a more general approach through reflection, such as a consteval approach which, allows passing a function which dictates a layout. Then provide "smallest" and other consteval functions in the standard library.

SF	F	N	A	SA
2	11	4	0	0

Poll: don't pursue declorder, instead change the standard to address the issue by making the default. AB: This would mean removing the wording about access order.

SF	F	N	A	SA
3	5	10	0	0

PB: describes pros and cons of best layout In the previous meeting, plenty of people said, it would create more trouble than benefits.


HD: `best` is a bad name. `performance` or something.

MM: `best` layout is not so much about the platform but behavior of the application, the only good way to figure it out is Profile-based Optimization and causes lot of backward comptability issues.

PB: I agree

Poll: Remove `best` layout

SF	F	N	A	SA
0	7	10	0	0

--  Wg 21 - 17 Jul 2019

Comments

Topic revision: r2 - 18 Jul 2019 - [Wg21](#)

Copyright © 2008-2019 by the contributing authors. All material on this collaboration platform is the property of the contributing authors.

Ideas, requests, problems regarding TWiki? [Send feedback](#)

