

Re: padding mechanism in structures

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2005-06/msg01831.html

- *From:* "Mehta Shailendrakumar" <shailendrakumar.mehta@xxxxxxxxxxxxx>
 - *Date:* Wed, 15 Jun 2005 14:11:40 +0200
-

see if this helps with sample code attachment

```
pack
#pragma pack( [ n ] )
```

Specifies packing alignment for structure and union members. Whereas the packing alignment of structures and unions is set for an entire translation unit by the `/Zp` option, the packing alignment is set at the data-declaration level by the `pack` pragma. The pragma takes effect at the first structure or union declaration after the pragma is seen; the pragma has no effect on definitions.

When you use `#pragma pack(n)`, where `n` is 1, 2, 4, 8, or 16, each structure member after the first is stored on the smaller member type or `n`-byte boundaries. If you use `#pragma pack` without an argument, structure members are packed to the value specified by `/Zp`. The default `/Zp` packing size is `/Zp8`.

The compiler also supports the following enhanced syntax:

```
#pragma pack( [ [ { push | pop }, ] [ identifier, ] ] [ n ] )
```

This syntax allows you to combine program components into a single translation unit if the different components use `pack` pragmas to specify different packing alignments.

Each occurrence of a `pack` pragma with a `push` argument stores the current packing alignment on an internal compiler stack. The pragma's argument list is read from left to right. If you use `push`, the current packing value is stored. If you provide a value for `n`, that value becomes the new packing value. If you specify an identifier, a name of your choosing, the identifier is associated with the new packing value.

Each occurrence of a `pack` pragma with a `pop` argument retrieves the value at the top of an internal compiler stack and makes that value the new packing alignment. If you use `pop` and the internal compiler stack is empty, the alignment value is that set from the command-line and a warning is issued. If you use `pop` and specify a value for `n`, that value becomes the new packing value. If you use `pop` and specify an identifier, all values stored on the stack are removed from the stack until a matching identifier is found. The

Re: padding mechanism in structures

packing value associated with the identifier is also removed from the stack and the packing value that existed just before the identifier was pushed becomes the new packing value. If no matching identifier is found, the packing value set from the command line is used and a level-one warning is issued. The default packing alignment is 8.

The new, enhanced functionality of the pack pragma allows you to write header files that ensure that packing values are the same before and after the header file is encountered:

```
/* File name: include1.h
*/
#pragma pack( push, enter_include1 )
/* Your include-file code ... */
#pragma pack( pop, enter_include1 )
/* End of include1.h */
```

In the previous example, the current pack value is associated with the identifier `enter_include1` and pushed, remembered, on entry to the header file. The pack pragma at the end of the header file removes all intervening pack values that may have occurred in the header file and removes the pack value associated with `enter_include1`. The header file thus ensures that the pack value is the same before and after the header file.

The new functionality also allows you to use code, such as header files, that uses pack pragmas to set packing alignments that differ from the packing value set in your code:

```
#pragma pack( push, before_include1 )
#include "include1.h"
#pragma pack( pop, before_include1 )
```

In the previous example, your code is protected from any changes to the packing value that might occur in `include.h`.

```
begin 666 pack.c
M(VEN8VQU9&4\<W1D:6\N:#X*"@IT>7!E9&5F(-T<G5C= H)"7L*"0EC:&%R
M"6YA;65;,C!=.PH)"6-H87();G5M8F5R.PH)"69L;V%T"7-L87)Y.PH)"7UB
M86YK7V5M<&QO>65E.PH*(W!R86=M82!P86-K*!U<V@L;F]W+#$!"@D)"G1Y
M<&5D968@<W1R=6-T"@D>PH)"6-H87();F%M95LR,%T["@D)8VAA<@EN=6UB
M97(["@D)9FQO870)<VQA<GD["@D)?65M<&QO>65E.PH*(W!R86=M82!P86-K
M*!O<"QN;W<I"@H*"6EN="!M86EN*"D*"7L*"0EE;7!L;WEE92!E;7!L;WEE
M93$["@D)8F%N:U]E;7!L;WEE92!E;7!L;WEE93(["@D)<')I;G1F*"S:7IE
M<R!A<F4@)60@"B5D7&XB+'-I>F5O9BAE;7!L;WEE93$I+'-I>F5O9BAE;7!L
8;WEE93(I*3L*"0ER971U<FX@,#L*"7T*
`
end
```

Re: padding mechanism in structures

- *Follow-Ups:*
 - ◆ *Re: padding mechanism in structures*
 - ◇ *From:* CBFalconer
 - ◆ *Re: padding mechanism in structures*
 - ◇ *From:* Lawrence Kirby

- *References:*
 - ◆ *padding mechanism in structures*
 - ◇ *From:* Roman Mashak

- Prev by Date: *Re: Printing a NULL pointer*
- Next by Date: *Re: pow(2, 1/2) != pow(2, 0.5) problem*
- Previous by thread: *padding mechanism in structures*
- Next by thread: *Re: padding mechanism in structures*
- Index(es):
 - ◆ *Date*
 - ◆ *Thread*