Subject: Re: Make THISFN simpler and more powerful
Posted by Lance on Wed, 09 Oct 2024 14:09:28 GMT
View Forum Message <> Reply to Message

I don't like that fact that compilers are allowed to stuff random padding bits in a bitfield as they like, but it's actually standard compliant.

In the above example, change unsigned to byte (Upp::byte of course) actually removes the extra cost on total storage usage. But the padding MSVC inserts vs GCC's sequential packed bits will result in binary incompatibilities. Worse, some old c tricks no longer work with MSVC.

A somewhat more realistic though simplified example.

```
class SomeFormat{
...
private:
   Font font;
   Color paper, ink, highlight;
   union{
     int32 dummy;
     struct{
        byte info1:3;
        byte info2:5;

        // allow individual font properties
        // to be Null for multi-tier composition
        bool faceNotNull:1;
        bool heightNotNull:1;
        bool widthNotNull:1;
        bool boldNotNull:1;
        bool strikeoutNotNull:1;
        bool underlineNotNull:1;
        bool italicNotNull:1;

     };
   };


};
```

In old c days, if we want to check if all Font properties are set, we can simply

```
bool SomeFormat::AllFontPropertiesSet()const
{
#define SOMEFORMAT_MASK (((1<<7)-1)<<8)
   return (dummy & SOMEFORMAT_MASK) == SOMEFORMAT_MASK;
#define SOMEFORMAT_MASK
```

```
}
```

And to mark all font properties as set(non-Null)

```
SomeFormat::SetAllFontProperties()
{
#define SOMEFORMAT_MASK (((1<<7)-1)<<8)
   return dummy |= SOMEFORMAT_MASK;
#define SOMEFORMAT_MASK
}
```

etc. With GCC, you can still do things like that. Total predictability. Fully appreciated.

End of the day, what benefits MSVC is going to achieve by padding random bits? I can see if a bitfield crosses a machine's fast-integer boundary (a few bits in previous FAST-INTEGER and a few in the following), there will be extra cpu cost involved. Other than that, what's going to be saved?

Thumbs down for MSVC on this regard.

---