Subject: Make THISFN simpler and more powerful
Posted by Lance on Tue, 18 Oct 2022 19:08:20 GMT
View Forum Message <> Reply to Message

in Core/Function.h, we have THISFN defined like this

```
template <class Ptr, class Class, class Res, class... ArgTypes>
Function<Res (ArgTypes...)> MemFn(Ptr object, Res (Class::*method)(ArgTypes...))
{
 return [=](ArgTypes... args) { return (object->*method)(args...); };
}

#define THISFN(x)   MemFn(this, &CLASSNAME::x)
```

This can be done in C++20 with the following

```
#define THISFN(memfun) std::bind_front(&std::remove_cvref_t<decltype(*this)>::memfun, this)
```

Of course this involves only library feature (no core language features), so it should be able to be rewritten to make available in current versons of c++ compilers/libraries that UPP aims to conform to atm.

An additional benefit of above macro definition is that it relieves the library user of the burden of an otherwise useless typedef

```
class MyClass
{
    typedef MyClass CLASSNAME;
//....
}
```

to be able to use thisfn

BTW, thisfn is provided in the UPP library for a good reason: there are occasions where using thisfn is so much easier than using an lambda.

eg.

```
 menu.Set( THISFN(MenuMain) );
```

vs

```
 menu.Set ( [=, this] ( Bar & bar )
  {
   MenuMain ( bar );
  }
```

);

And imagine when the callback accept a few more parameters.
[/code]

U++ is currently aiming at compliance to C++14. What if you want to use it in your project and you know your compiler/c++library are modern enough?

One way is to put the following in a header file and make sure you include it after <Core/Core.h> is included.

```
#ifdef THISFN
#    undef THISFN
#    define THISFN(memfun) ....
#endif
```

Reference:
Why use `std::bind_front` over lambdas in C++20?