I SPY with my little eye

Boost.Predef in a constexpr world

A docudrama by Joël FALCOU

@joel_f

The downfall of the C++ Preprocessor

- ✗ **Preprocessing is dying everywhere**
  - ✗ Modules = no more #includes
  - ✗ Templates functions & variables = 0 #define
  - ✗ Variadics = no more PP blood magic
- ✗ **Everywhere ? Not in a tiny corner of C++**
  - ✗ #ifdef nests for platform checks
  - ✗ #ifdef nests for compiler versions
  - ✗ Basically all Boost.Predef use cases

Introducing SPY

- SPY is a C++17 library providing:
  - Constexpr compatible detection of platform specifics informations
  - Easy syntax for comparing versions
  - Easy to extend

- Early implementation at: https://github.com/jfalcou/spy

Checks with if constexpr

- ✗ Checks for exact match on:
  - ✗ Compilers
  - ✗ OS
- ✗ Checks for version ordering:
  - ✗ Compilers
  - ✗ OS
  - ✗ libc
  - ✗ stdlib

```cpp
#include <spy/compiler.hpp>
#include <spy/os.hpp>

// Check for a given compiler
if constexpr(spy::os == spy::linux_)
  std::cout << "LINUX\n";

// Check for a compiler version
if constexpr(spy::compiler > 8'2_gcc)
  std::cout << "G++ 8.2 or sup.\n";
```

Interaction with concepts

- ✗ Why?
  - ✗ If constexpr still ODR-checks its branch
  - ✗ You can't call a non-existing function from a non-evaluated branch of if constexpr
- ✗ Solution :
  - ✗ Use Concepts
  - ✗ Still requires template :(

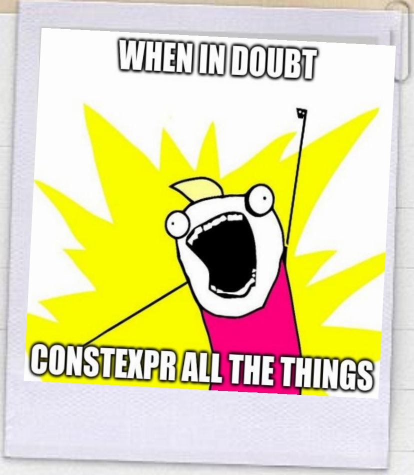
```
#include <spy/compiler.hpp>

template<typename T>
auto f(T t) requires( spy::clang )
{
  return __builtin_bitreverse32(t);
}

template<typename T>
auto f(T t) requires( spy::gcc )
{
  return __builtin_bswap32(t);
}
```

What's left to do ?

- ✗ Hardware detection
  - ✗ Memory model
  - ✗ SIMD extension
  - ✗ GPGPU related info
- ✗ Extension to runtime
  - ✗ Portable wrapper for CPUID ?
  - ✗ Cache infos ?
- ✗ Worth standardizing ?



Thanks for your attention !!