Enable Modules on Windows

Vaibhav Garg
Google Summer of Code' 20 | CERN-HSF | Project ROOT

Mentors:
Vassil Vassilev & Bertrand Bellenot

Modules

What are they?

- Software is built using libraries (system + third-party).
- In C, we access the libraries using `#include <SomeLib.h>`
- Modules provide an alternate, simple way to access libraries.

Benefits?

- Better compile-time scalability.
- Elements problems inherent to using the C preprocessor to access the API of a library.

Why switch?

Problems with #include

- Compile-time scalability.
  - Every time a header is included, it's contents need to be parsed.
  - For example, say, you have a project with M translation units each having N header files in each unit, the compiler needs to perform M x N level of work.
  - This is worse in C++, as support for templates forces a large amount of code in each header.

Why switch?

Problems with #include

- Compile-time scalability.
- Fragility
  - #include directive is treated as textual inclusion.
  - They are, therefore, subject to any active macro definitions at the time of inclusion.
  - If any of the active macro definitions happens to collide with a name in the library, it causes failures.
  - Workarounds possible, ex: Include guards.

How do Modules solve these issues?

Semantic Import

```
import std.io ;                    // pseudo-code
```

Modules improve access to the API of software libraries by replacing the textual preprocessor inclusion model with a more robust, more efficient semantic model.

There is only a minor change in the user's perspective, but the import declarations behave quite differently.

C++ Modules in ROOT

- The ROOT v6.16 release came with a preview of the module technology.
- C++ Modules are default in ROOT, starting from v6.20 in UNIX and v6.22 in OS X.
- My project aimed to extend the support of C++ Modules of ROOT to Windows.

Why is Windows different?

- Windows uses MSVC (Microsoft Visual C++), instead of commonly used GCC in Unix based systems.
- MSVC has an absence of C99 support.
- Also, several GCC specific headers are not present in MSVC. Ex: `bits/allocator.h` and likewise.
- MSVC allows many invalid constructs in class templates that Clang has historically rejected.

Major Changes

- New Modulemap files for Standard Library of Windows
- Merge Two Decl's successfully when Inheritable attributes are present. (Clang)
- Teach DynamicLibraryManager to recognise symbols in COFF Object Files
- A lot of other fixes here and there.

Results

- We are now able to successfully build ROOT on Windows with C++ modules enabled.
- 65% of the tests are currently passing.
- Some more minor issues are needed to be fixed in order to make Modules default on Windows.

Thank you!