What’s new in C++14, and how you can take advantage of it

Marshall Clow
Qualcomm
mclow.lists@gmail.com

Euro LLVM 2014

C++1y status

- DIS approved in October (Chicago)
- FDIS approved in February (Issaquah)
- Voting in progress
- Voting concludes in August

How did we get here?

- C++98/03
- TR1
- C++11
- C++14
- ... and beyond

C++11 introduced many new features and concepts

- threading
- range-based for loops
- auto
- lambdas
- move semantics
- variadic templates and tuples
- user-defined literals
- regular expressions
- uniform initialization
- unordered containers
- std::chrono
- constexpr

C++14 is much more focused

- Fleshing out the features introduced in C++11
- A few new features
- Fixing bugs

Fleshing out

- constexpr
- tuples
- make_XXX

constexpr

- Now much more full-featured
- No more torturing of the ?: operator
- loops, variables

Tuple enhancements

- find element by type
  - get<string> (tup)
- Compile-time integer sequences

make_XXX

- make_move_iterator (C++11)
- make_shared (C++11)
- make_unique (C++14)
- make_reverse_iterator (C++14)

New features

- Polymorphic lambdas
- Variable templates
- Digit separators
- Binary literals
- Heterogeneous lookup in containers
- Quoted IO of strings

Polymorphic lambdas

- An aid to using lambdas in generic code
- `[=y](auto x) { return x == y; }`

Variable templates

- Before, you could use templated classes, structs, functions
-
  ```
  template<typename T>
  constexpr T pi = T{3.14159265358979 32385};
  ```

Digit separators

- After much debate, the committee settled on single quote
- `unsigned long long x = 123’456’789;`

Binary literals

- Now can use bit patterns directly
- `unsigned int foo = 0b001001010; // 74`

Heterogenous Lookup

- Consider `std::map<string, Foo> x;`
- `x.find (“abc”)`
- What does this do?

Quoted I/O in strings

```
string x{"Hello World"};
strstream ss;

ss << x;
string y;
ss >> y;
assert ( x == y );
```

Quoted I/O in strings (2)

```
string x{"Hello World"};
strstream ss;

ss << quoted(x);
string y;
ss >> quoted(y);
assert ( x == y );
```

Fixing bugs

- Fixing some bad specifications
- Restoring the strong exception guarantee in vector::push_back
- Disallowing temporaries in some places

Disallowing temporaries

- Some parts of the standard library return references into containers that are passed to them
- if the container is a temporary, then these references are “stale” as soon as they are returned.

Temporary example

```
string f() { return "m123.txt"; }



const regex r(R"(m(\d+).*)");
smatch m;
if (regex_match(f(), m, r))
    DoSomethingWith(m[1]);
```

Implementation Status

- C++98/03 took *years* to implement.
- C++11 implementation is ongoing.
- C++14 implementation is also ongoing.

C++11 implementations

- clang & libc++ shipped a complete C++11 implementation in 3.3 (June 2013)
- gcc supported the full language in 4.8.1 (May 2013), and libstdc++ will be complete in 4.9 (real soon now)
- Visual C++ has implemented many of the language and library features, but not all (more on VC++ later)
- Oracle shipped a beta compiler with limited C++11 support last week.

C++14 implementations

- clang & libc++ shipped a complete C++1y implementation in 3.4 (January 2014)
- clang & libc++ will ship a complete C++14 implementation in 3.5 (May/June? 2014)
- gcc & libstdc++ support a few C++14 features in 4.8, more in 4.9
- Visual C++ is implementing C++11 and C++14 together.
    - Rolling out features in “technology previews”

What comes next?

- What the heck is a TS, anyway?
- C++1z?

Technical Specifications

- Filesystem
- Library Extensions
- Array Extensions
- Parallelism
- Concepts
- Modules

Committee Study groups

- Ranges
- Networking
- Reflection
- ... and others

Summary

- For a long time, C++ was a static (unchanging) language.
  - Not any more!
- Lots of people are doing research and experimentation with C++
  - The tools provided by LLVM and clang are enabling this
- The goal is to make C++ a “better” language without sacrificing those things which it excels at (performance, generality, portability, etc).

Questions?