std::format support in Qt

Qt Contributor Summit 2024
Ivan Solovev <ivan.solovev@qt.io>

5 September 2024

Qt Development

Points to discuss

- std::format support for Qt types
- Formatting into QString using std::format-like syntax

Everything is tracked in QTBUG-104651

std::format support for Qt types (1/2)

- Why is that important?
  - The users should not provide their own formatters for Qt types, because that has a risk of ODR violation

- Current situation
  - qfloat16 got support for std::format, it's already in dev (QTBUG-104654)
  - Patches for Qt strings are in gerrit (QTBUG-104652)
  - Patches for QFlags and Qt enums are in gerrit (QTBUG-125325)
  - WIP patch to enable std::format support for all types that support QDebug streaming:
    https://codereview.qt-project.org/c/qt/qtbase/+/587797
    It turns out to be quite tricky, because it immediately causes ambiguities with existing std::formatter specializations that are provided by the standard library.

std::format support for Qt types (2/2)

- Which types should gain support for std::format?
  - All value types?
  - All types in QtCore (QtNetwork/QtGui/other modules...)?
  - All types that have QDebug streaming? Maybe we can provide a convenience macro for that?

- Should we provide custom format arguments?
  - We could start from simply using existing format arguments, because most of Qt types would be formatted into a string anyway
  - Things like QByteArray::toPercentEncoding() might justify custom format arguments
  - The tricky part is that I didn't yet figure out how to handle dynamic format specifiers

- Implementation details
  - For now, I add a separate header for each type or group of types (qfloat16format.h, qstringformat.h, etc...). The idea is that the users do not have an overhead of std::format support if they do not need it. But that does not scale well.
  - Should we have some "central" header for std::format support?
  - How to document std::format support in an understandable way?

Formatting into QString

- An alternative to QString::arg() APIs, but with a std::format-like syntax
- C++20 (as well as C++23, and most probably C++26) does not support formatting into char16_t-based strings, so we need to invent our own way to format into QString
- Thiago suggested (in the SG16 ML) to provide a custom OutputIterator and a qFormat() function that will wrap std::format_to(). That is now tracked in QTBUG-126873
  - The idea is that std::formatter implementation detects the usage of the custom OutputIterator and then can do all the optimizations related to memory allocation
  - We also hope that we would not need to reimplement the std::formatter<T>::parse() method for that

Questions:

- Any other ideas how to implement it?
- Is it useful for us now? The code would be C++20 only, so we would not be able to use it inside Qt until we require that Qt builds with C++20 only

Thank you!