P2665R0

Overloading T with const T&

Bengt Gustafsson

Kona - 2022

Presentation contents

- Rationale
- Proposal contents
- Rules for compilers
- Rules for programmers

Rationale

- There is no one best calling convention.
- This varies by the type, the compiler, the ABI and the platform.
- The compiler knows which is most efficient.
- Allowing the compiler to select allows optimizing each call site.

Proposal contents

- Today it is allowed to create overload sets containing:

```
template<typename T> class vector {
    void push_back(T value);
    void push_back(const T& value);
    void push_back(T&& value);
};
```

- What's new is that **push_back** can still be called.
- The compiler selects any callable function.
- Here: **T** or **const T&** for *lvalues* and **T** or **T&&** for *rvalues*.
- With multiple parameters you can overload even more!

Rules for compilers

- The compiler selects freely between T and const T& for each call site.
- In reality there will be more or less advanced heuristics which can evolve between compiler versions.
- No ABI issues: The only difference is which function gets called.

Rules for programmers

- Overloads must **do the same thing**.
- Don’t rely on the next version of the compiler making the same choices for the same source code.
- Don’t rely on another platform’s compiler making the same choices for the same source code.

Relation to other proposals

- P2667: Rule based parameter passing:

  Provides a way to implement all overloads at once.

- P2666: Last use optimization:

  Simplifies finding rvalues when calling functions.

- Coming up later:

  Labelled types provides named parameters.
  Simplified declarators with all of the type first.