The dangers of C-style casts

Joshua Gerrard

Example 1

```
float x = 1.0f;

double* y = (double*) &x;
```

Who thinks this will compile?

- GCC 5.2.0 and MSVC 19 don't even emit a warning ...

```
float x = 1.0f;

double* y = static_cast<double*>(&x);
```

Who thinks this will compile?

```
GCC 5.2.0 invalid static_cast from type 'float*' to type 'double*'

MSVC 19 C2440 'static_cast': cannot convert from 'float *' to 'double *'
```

Example 2

```
struct T;

const float x = 1.0f;

T* y = (T*) &x;
```

Who thinks this will compile?

- GCC 5.2.0 and MSVC 19 don't even emit a warning …

```
T* y = static_cast<T*>(&x);
```

Who thinks this will compile?

```
GCC 5.2.0 invalid static_cast from type 'const float*' to type 'T*'

MSVC 19 C2440: 'static_cast': cannot convert from 'const float *' to 'T *'
```

What's going on here?

N4296 - 5.4 Explicit type conversion (cast notation) [expr.cast]

The conversions performed by [a C-style cast are, in this order, until one succeeds]:

- const_cast
- static_cast*
- static_cast* followed by a const_cast
- reinterpret_cast
- reinterpret_cast follow by a const_cast

Eek.

Why the * on the static_cast?

[…] when performing a static_cast in the [above] situations the conversion is valid even if the base class is inaccessible:

- a pointer to an object of derived class type or an lvalue or rvalue of derived class type may be explicitly converted to a pointer or reference to an unambiguous base class type, respectively;
- a pointer to member of derived class type may be explicitly converted to a pointer to member of an unambiguous non-virtual base class type;
- a pointer to an object of an unambiguous non-virtual base class type, a glvalue of an unambiguous non-virtual base class type, or a pointer to member of an unambiguous non-virtual base class type may be explicitly converted to a pointer, a reference, or a pointer to member of a derived class type, respectively.