Potential Pitfalls

Accidental use of C-style ellipsis

Inside the function parameters declaration, `...` can be used only in conjunction with a template parameter pack. However, there is an ancient use of `...` in conjunction with C-style variadic functions such as `printf`. That use can cause confusion. Say we set out to declare a simple variadic function, `process`, that takes any number of arguments by pointer:

```
class Widget;                   // declaration of some user-defined type

template <typename... Widgets>  // parameter pack named Widgets
int process(Widget*...);        // meant as a pack expansion, but is it?
```

The author meant to declare `process` as a variadic function taking any number of pointers to objects. However, instead of `Widgets*...`, the author mistakenly typed `Widget*...` (note the missing "s"). This typo took the declaration into a completely different place: It is now a C-style variadic function in the same category as `printf`. Recall the `printf` declaration in the C Standard Library:

```
int printf(const char* format, ...);
```

The comma and the parameter name are optional in C and C++, so omitting both leads to an equivalent declaration:

```
int printf(const char*...);
```

Comparing `process` (with the typo in tow) with `printf` makes it clear that `process` is a C-style variadic function. Runtime errors of any consequence are quite rare because the expansion mechanisms are different across the two kinds of variadics. However, the compile- and link-time diagnostics can be puzzling. In addition, if the variadic function ignores the arguments passed to it, calling it might even compile, but the call will likely use a different calling convention than what was intended or assumed.

As an anecdote, a similar situation occurred during the review stage of this feature section. A simple misunderstanding caused a function to be declared inadvertently as a C-style variadic instead of C++ variadic template, leading to numerous indecipherable compile-time and link-time errors in testing that took many emails to figure out.

Undiagnosed errors

Description — Corner cases of function template argument matching on page 900 shows definitions of variadic template functions that are in error according to the C++ Standard yet pass compilation on contemporary compilers — that is, IFNDR. In certain cases, they can even be called. Such situations are most assuredly latent bugs: