X3J16/96-0095
WG21/N0913
May 28th, 1996
John H. Spicer, Edison Design Group

Fixing Name Injection from Templates

1. Introduction

Name injection from template instantiations has been the subject of a great deal of debate over the past several years. The following example illustrates what is meant by name injection:

```
struct f {};
template <class T> struct A {
        friend void f(A<T>){}
};

int main()
{
        void* fp;
        fp = f; // Error - f is a struct
        A<int> a;
        fp = f; // OK - only one instance of f
        A<char> ac;
        fp = f; // Error - f is now overloaded
}
```

The most serious problems with injection are

1. When the first instantiation of the enclosing class occurs, it is possible that a name that was previously declared to be a type, suddenly becomes a function (as is the case with `f` in the example).

2. The point at which a particular version of a function is declared is difficult to predict. It is possible to construct cases whose meaning changes when the point of instantiation of a class changes from one place to another.

2. The History of Name Injection and X3J16/WG21

Three years ago, the WP was not clear on the subject of name injection. Existing practice was unanimous, however. Injections occurred as part of the instantiation of the enclosing class. Some committee members were concerned, however, because of the fact that a program's behavior can depend on the point of instantiation of a class. This is a concern because it can be very difficult to know exactly where that point is. In San Jose, the Extensions WG decided to clarify that injection was supposed to occur, but this decision was put on hold pending further discussion. The subject was discussed further, and at the next meeting, in San Diego, the Extensions WG proposed that a program attempts to inject something that was not previously declared is ill-formed.

This decision was itself controversial because it affected existing code and disallowed certain coding practices without providing an alternative. More than a 1.5 years later, after numerous discussions of the subject, a motion to restore injection was approved in Valley Forge.

More than 1.5 years has passed since Valley Forge, and the subject of injection has been discussed at virtually every meeting since. There is now a proposal from Bill Gibbons (N0878) to replace name

injection from templates with a special lookup that would find friend functions based on the operand types of the function being called.

3. A Simple Alternative that Retains Injection

A simple change can solve the first serious problem with injection. Simply inject names when the template is defined. Friend classes and friend class templates can be injected when the template is defined, because they can't depend on template parameters of the enclosing template. Functions can, and usually do, depend on the template parameters of the enclosing templates, so the actual function signatures cannot be injected until the enclosing class instantiations are done. However, a "generic" function or function template can be injected. When instantiations of the enclosing class take place, the real functions and function templates would be added to the overload set containing the "generic" entry created when the template definition was scanned.

I believe that this change would be sufficient, and would solve the problems as effectively as the proposal by Bill Gibbons. His proposal implicitly requires that the types of function arguments be fully instantiated, if they had not been previously. If such rules are used in conjunction with injection it makes injection and Bill's alternative equivalent in terms of the second problem described above.

If this is not considered to be sufficient, a "reconsideration" rule could be added. Such a rule would state that the result of any lookup that produced an overload set containing one of the "generic" functions, must produce the same result if repeated at the end of the translation unit.

4. Why we shouldn't get rid of injection

4.1 There is no proposal that provides a complete replacement for injection

The proposed replacement only works when a parameter type of the friend function is an object of the enclosing class type or one of its base classes. In other words, in a class template Derived<T>, it would work with friend `void f(Derived<T>)`, `friend void f(Base<T>)`, but would not work with `friend void f(T)` or `friend void f(Helper<T>)`.

4.2 Eliminating injection from templates would give template classes and nontemplate classes different semantics

Currently, an instantiated template has the same semantics as a defined class. Eliminating injection from templates would give templates and nontemplate different semantics. You then have the problem of choosing which set of rules to apply to explicit specializations.

4.3 Injection has not been a source of problems in practice

Although it is possible to describe situations that could present problems, injection has not been a source of problems so far, despite the fact that virtually all compilers do name injection. One compiler did not do injection (because they were using a WP that prohibited it) and was forced to add it later because of user demand. At EDG, we've received no complaints or questions regarding injection and I've seen no such complaints on Usenet or in literature on C++ programming.

4.4 We understand injection, but have no experience with the alternatives

It isn't perfect, but we know its problems and their impact. Removing injection breaks some amount of existing code without providing an equivalent replacement.