Visibility

Pieter P

ELF symbol visibility and binding can be complex, especially when considering the interaction of different visibility attributes, compiler options, and link types. This page provides an interactive summary of how these factors affect the binding and visibility of various types of symbols in C++, both in the object files and in the final linked binary. Generated using GCC 11.4.0 on Linux x86-64.

ELF symbol binding and visibility summary

Choose a link type, `-fvisibility` setting, and `-fvisibility-inlines-hidden` option to generate the table below. The rows summarize binding and visibility for each function type in the object file, the final target symbol table (`.symtab`), and whether the symbol appears in `.dynsym`. The meaning of the different fields can be found in the ELF specification in the references below.

Link type

(●) Shared library ( ) Executable ( ) Executable with exports

`-fvisibility`

( ) Default ( ) Protected (●) Hidden

`-fvisibility-inlines-hidden`

( ) Off (default) (●) On (hidden)

Compile: `g++ -fPIC -fvisibility=hidden -fvisibility-inlines-hidden`
Link: `g++ -fPIC -shared`

Description	Definition	Object Binding	Object Visibility	Binary Binding	Binary Visibility
Normal function	`void func() { … }`	global	hidden	local	default
Inline function	`inline void func() { … }`	weak	hidden	local	default
Static function	`static void func() { … }`	local	default	local	default

Anonymous namespace function	`namespace {`
	`void func() { … }`
	`}`
		local	default	local	default

Explicit default visibility	`void [[gnu::visibility("default")]] func() { … }`	global	default	global	default
Explicit protected visibility	`void [[gnu::visibility("protected")]] func() { … }`	global	protected	global	protected
Explicit hidden visibility	`void [[gnu::visibility("hidden")]] func() { … }`	global	hidden	local	default
Function template	`template <class T>` `void template_func() { … }`	weak	hidden	local	default
Inline function template	`template <class T>` `inline void inline_template_func() { … }`	weak	hidden	local	default

Description	Definition	Object Binding	Object Visibility	Binary Binding	Binary Visibility

Member function of class template without attribute, implicitly instantiated	`template <class T>`
	`struct Class {`
	`  void func() { … }`
	`};`
	
	`Class<int> obj;  // implicitly instantiated`
	`obj.func();`
		weak	hidden	local	default

Out-of-line member function of class template without attribute, implicitly instantiated	`template <class T>`
	`struct Class {`
	`  void out_of_line_func();`
	`};`
	
	`template <class T>`
	`void Class<T>::out_of_line_func() { … }`
	
	`Class<int> obj;`
	`obj.out_of_line_func();`
		weak	hidden	local	default

Member function of class template without attribute, explicitly instantiated with default visibility	`template <class T>`
	`struct Class {`
	`  void func() { … }`
	`};`
	
	`struct InstDefault;`
	`template struct [[gnu::visibility("default")]]`
	`  Class<InstDefault>;`
	
	`Class<InstDefault> obj;`
	`obj.func();`
		weak	default	weak	default

Out-of-line member function of class template without attribute, explicitly instantiated with default visibility	`template <class T>`
	`struct Class {`
	`  void out_of_line_func();`
	`};`
	
	`template <class T>`
	`void Class<T>::out_of_line_func() { … }`
	
	`struct InstDefault;`
	`template struct [[gnu::visibility("default")]]`
	`  Class<InstDefault>;`
	
	`Class<InstDefault> obj;`
	`obj.out_of_line_func();`
		weak	default	weak	default

Member function of class template with default visibility, implicitly instantiated	`template <class T>`
	`struct [[gnu::visibility("default")]] DefaultClass {`
	`  void func() { … }`
	`};`
	
	`DefaultClass<int> obj;  // implicitly instantiated`
	`obj.func();`
		weak	hidden	local	default

Out-of-line member function of class template with default visibility, implicitly instantiated	`template <class T>`
	`struct [[gnu::visibility("default")]] DefaultClass {`
	`  void out_of_line_func();`
	`};`
	
	`template <class T>`
	`void DefaultClass<T>::out_of_line_func() { … }`
	
	`DefaultClass<int> obj;`
	`obj.out_of_line_func();`
		weak	hidden	local	default

Member function of class template	`template <class T>`
	`struct [[gnu::visibility("default")]] DefaultClass {`
	`  void func() { … }`
	`};`
		weak	hidden	local	default

Description	Definition	Object Binding	Object Visibility	Binary Binding	Binary Visibility

with default visibility, explicitly instantiated without attribute	`struct Inst;`
	`template struct DefaultClass<Inst>;`
	
	`DefaultClass<Inst> obj;`
	`obj.func();`				

Out-of-line member function of class template with default visibility, explicitly instantiated without attribute	`template <class T>`
	`struct [[gnu::visibility("default")]] DefaultClass {`
	`  void out_of_line_func();`
	`};`
	
	`template <class T>`
	`void DefaultClass<T>::out_of_line_func() { … }`
	
	`struct Inst;`
	`template struct DefaultClass<Inst>;`
	
	`DefaultClass<Inst> obj;`
	`obj.out_of_line_func();`
		weak	hidden	local	default

Member function of class template with default visibility, explicitly instantiated with protected visibility	`template <class T>`
	`struct [[gnu::visibility("default")]] DefaultClass {`
	`  void func() { … }`
	`};`
	
	`struct InstProtected;`
	`template struct [[gnu::visibility("protected"))]]`
	`  DefaultClass<InstProtected>;`
	
	`DefaultClass<InstProtected> obj;`
	`obj.func();`
		weak	default	weak	default

Out-of-line member function of class template with default visibility, explicitly instantiated with protected visibility	`template <class T>`
	`struct [[gnu::visibility("default")]] DefaultClass {`
	`  void out_of_line_func();`
	`};`
	
	`template <class T>`
	`void DefaultClass<T>::out_of_line_func() { … }`
	
	`struct InstProtected;`
	`template struct [[gnu::visibility("protected")]]`
	`  DefaultClass<InstProtected>;`
	
	`DefaultClass<InstProtected> obj;`
	`obj.out_of_line_func();`
		weak	default	weak	default

Caveats

- The visibility attribute of an explicit instantiation of a class template does not override the visibility of the class definition. No warning is emitted by GCC; the attribute is simply ignored. However, if the class template definition is marked with default visibility, but the explicit instantiation does not specify a visibility, the visibility of the explicit instantiation is determined by the `-fvisibility` option.
  Contrary to the implicit instatiation case, such an explicit instantiation without a visibility attribute does preserve default visibility if `-fvisibility=default` and `-fvisibility-inlines-hidden` are enabled.
- If an `extern template` declaration for an explicit class template instantiation exists, the visibility of the `extern template` declaration is used, even if the explicit instantiation does not specify a visibility. GCC warns if the visibility of the `extern template` declaration does not match the visibility of the explicit instantiation (`warning: type attributes ignored after type is already defined [-Wattributes]`).
- In the case of an implicit instantiation, the `-fvisibility=hidden` and `-fvisibility-inlines-hidden` options hide member functions of class templates, even if the class template itself is marked with default visibility. The `-fvisibility-inlines-hidden` option only hides inline member functions of class templates, without affecting member functions with out-of-line definitions.
  To force visibility of all member functions, the class template must be explicitly instantiated with default

visibility. This should be done before any implicit instantiations occur, which typically means it must be done in the header file, using an `extern template`.
- Even though function templates and inline function templates both emit weak symbols (similar to ordinary inline functions), the visibility of the emitted symbols is different when `-fvisibility-inlines-hidden` is enabled.

Methodology

Test program built with a matrix of different visibility settings and link types, then parsing the output of `readelf` to generate the table.

▶ Test program source code ...

External references

- GCC Visibility Wiki
- GCC -fvisibility Option
- GCC -fvisibility-inlines-hidden Option
- Linux Foundation ELF Specification
