Autotools Mythbuster

Prev Next

3. Exposing and Hiding Symbols

For shared objects to be helpful, both in case of dynamic libraries and plugins, they have to expose (or export) an interface to the other programs. This interface is composed of named functions and data symbols, that can be accessed either by linking to the library or by using `dlsym()` and similar interfaces provided by the runtime loader.

Not all symbols defined within a shared object need to be exported, though. In the vast majority of cases, a dynamic library will provide a set of symbols corresponding to its public API, while for plugins, the interface to be exposed is usually mandated by the host application or library.

Exposing more symbols than necessary can have negative effects on the object in many ways: it almost always increases the time necessary for the dynamic loader to completely load the object, and – if the internal symbols are not properly guarded – it can cause collisions between different objects, on operating systems using flat namespaces, as it is the case for Linux and most Unix-based systems.

Most, if not all, link editors allow to avoid this problem by defining a list of symbols to export; any symbol not in such lists will be hidden and thus not be part of the public interface of the object. Since the options used by the link editors to provide this function are not standard, libtool leverages it via three main options: `-export-dynamic`, `-export-symbols` and `-export-symbols-regex`.

3.1. `-export-dynamic`

The `-export-dynamic` option is used to declare that the interface exposed by the current object is to be used with the `dlopen()` and `dlsym()` functions (or their equivalent on non-Unix operating systems). This is the case for instance of all plugins, as seen in Section 2, “Building plugins”.

This option is not commonly used for projects whose main target is Linux or other operating systems using ELF for their objects, as any symbol exposed by an ELF object is available to be accessed through the `dlsym()` function. It is a requirement, though, of other operating system that make difference whether the symbol should be resolved at build time or during execution, such as Windows.

3.2. `-export-symbols` and `-export-symbols-regex`

As the title implies, the `-export-symbols` and `-export-symbols-regex` are tightly related. They both are used to provide libtool with a list of symbols that should be exposed (the interface of the object).

The first option takes as a single parameter the path to a file, containing the list of symbols to expose, one per line. The second instead takes as a parameter a regular expression: symbols whose name matches the expression will be exposed by the object; libtool takes care of producing the list in that case.

Once libtool knows the list of symbols to expose, it then uses the link editor's own interface to complete the task; this is done through either *linker scripts* for Unix-derived link editors, or through *definition lists* for link editors for Windows, as they both serve similar purposes.

Example 3.7. Exposing only the public interface of a library via **`-export-symbols-regex`**

```
lib_LTLIBRARIES = libfoo.la

libfoo_la_SOURCES = foo1.c foo2.c foo3.c
libfoo_la_LDFLAGS = -export-symbols-regex '^foo_'
```

Using the `-export-symbols-regex` option makes it very easy to hide unnecessary symbols from a library's interface, but relies on the library being designed to use a regular pattern for naming of non-static functions and data symbols. In the earlier example, for instance, libtool will export all the symbols whose name start with `foo_`, assuming that the internal symbols use instead a prefix like `x_foo` or something along those lines.

When this assumption cannot be applied, you have instead to use the other option, `-export-symbols`, providing it with a complete list of the interfaces to export. The main downside to this method is, obviously, that you have to either manually compile it (which is prone to errors) or find a different, automated way to produce it, similarly to what libtool does when provided with a regular expression.
