Somewhat merging gobject-introspection into GLib

25 minutes allocated. 20 minutes for talk, 5 minutes for questions.
Hello, this talk is about work we did in the GNOME 46 / GLib 2.80 cycle to move some of gobject-introspection into GLib and stabilise its APIs. Emmanuele Bassi did a talk yesterday about these changes from the point of view of introspection, annotations and library APIs. I'll be talking about it from the point of view of GLib, apps, language bindings and distros.
The work for this merge was funded by the Sovereign Tech Fund (STF) to help with keeping our fundamental infrastructure up to date.
I'll go through what the problems were with the old arrangement, what we changed, and how it impacts on apps/bindings/documentation.

Somewhat merging gobject-introspection into GLib

The problem

The problem

- gobject-introspection was meant to incubate separately from GLib
- Time passed (10+ years)
- gobject-introspection depends on GLib
- But GLib's introspection data is generated by gobject-introspection

When gobject-introspection was started as a project, it was deliberately started outside the glib.git tree, in order to allow it to incubate. The idea was to eventually merge it in. 10+ years later, now we've done that.

Somewhat merging gobject-introspection into GLib

The problem

The problem

- gobject-introspection was meant to incubate separately from GLib
- Time passed (10+ years)
- gobject-introspection depends on GLib
- But GLib's introspection data is generated by gobject-introspection

The previous status quo of having gobject-introspection be out-of-tree worked OK, but it made for slow iteration. The introspection data for GLib was generated in the gobject-introspection build – a separate module – which had to be periodically and manually updated from glib.git. This made for busy work, and also meant that GLib couldn't use any of its introspection data internally. For example, for building documentation.

Somewhat merging gobject-introspection into GLib

What is gobject-introspection?

What is gobject-introspection?

- g-ir-scanner
- g-ir-compiler
- libgirepository
- GIR (XML) format
- Typelib (binary) format

What actually is the gobject-introspection project formed of? Various libraries, formats and utilities. They come together to allow you to take a C library, and scan its source code to extract API definitions using `g-ir-scanner` to produce a GIR (XML) file. You can then use `g-ir-compiler` to 'compile' that GIR file into an architecture-dependent binary typelib file. Language bindings load these typelib files when you import a module. Finally, libgirepository provides programmatic access to read and write typelib files. Most bindings use this to read the typelib files, and `g-ir-compiler` uses it to generate them.

Somewhat merging gobject-introspection into GLib

Merging the two

So, we just move all of gobject-introspection.git into GLib and be done, right? No. `g-ir-scanner`, which is the part of gobject-introspection which parses C source to extract API definitions and generate GIR files, is written in a mixture of CPython and Python, and needs the Python development headers to be built. Adding that as a dependency to GLib would be a big move.

Merging the two

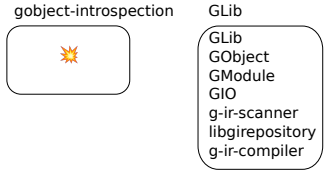

Figure: Merge all of gobject-introspection into GLib ??? Profit

Somewhat merging gobject-introspection into GLib

Merging the two, approach 2

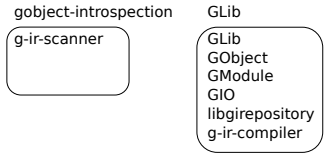

Figure: Merge libgirepository into GLib ??? Profit

So the approach we actually took was to merge half of gobject-introspection into GLib — `libgirepository` and `g-ir-compiler`. `g-ir-scanner` remains in the gobject-introspection.git repository for the moment.

This means we've essentially split the pipeline into the parts which generate GIR files, in gobject-introspection.git — and the parts which consume GIR files and generate and read typelib files, in glib.git.

Somewhat merging gobject-introspection into GLib

API changes

API changes

- `girepository-1.0.pc` → `girepository-2.0.pc`
- `GIRepository-2.0.typelib` → `GIRepository-3.0.typelib`
- `g-ir-scanner` → `gi-compile-repository`
- `g_` prefix → `gi_` prefix
- `GTypeInstance`

There have been no changes to the typelib or GIR format, so all the tools and libraries should be interoperable on formats. We have, however, made a number of API changes to libgirepository and the utilities when moving them into GLib, to tidy things up. In particular, libgirepository's home-grown type system has been replaced with a more-standard `GTypeInstance`.

The new version of libgirepository, and all the utilities, are parallel installable with the old version. So consumers of the old versions can continue to use them. They continue to be shipped with gobject-introspection for the moment.

Somewhat merging gobject-introspection into GLib

API changes

API changes

https://docs.gtk.org/girepository/migrating-gi.html

The API changes and renames are all documented in a migration guide.

What does this mean for me? Apps

- Nothing!
- Apps still generate GIR files as before
- Just use whatever your build system uses for compiling typelibs (`gnome.generate_gir()`)

If you're an app author, you don't need to do anything at all to adapt to these changes. None of the pipeline for building GIR files has changed. You could move from `g-ir-compiler` to `gi-compile-repository` to turn GIR files into typelibs, but there's no pressing need to. Just use whatever your build system makes it easy to use. (For Meson, at the moment, this is `g-ir-compiler` via `gnome.generate_gir()`.)

What does this mean for me? Bindings

- Think about porting to the new version of libgirepository
- Probably not feasible to make it a compile time option

If you're a binding developer, you don't need to do anything immediately. You may want to put an item on the roadmap for porting to the new version of libgirepository. Because of the API changes in it, this will require numerous small code changes, but many of those changes can be made mechanically. Because of the number of small changes, supporting both versions of libgirepository via a compile time option is probably not feasible. But because the typelib format hasn't changed, you don't need to coordinate porting with other bindings or with distros.

Porting should mean that you end up with cleaner code overall, due to the API improvements in libgirepository. If you find parts of the API which are still awkward or could do with improvement, please let us know by filing an issue against GLib.

Somewhat merging gobject-introspection into GLib

What does this mean for me? Distros

What does this mean for me? Distros

- You'll end up shipping parallel libgirepository and utilities for a while
- No need for a flag day until you want to stop packaging the old versions
- Packaging GLib now needs a bootstrapping build without introspection, then again with introspection

This will be a one-time change for distro packagers, and all the details you should need are included in the GLib 2.79.x and 2.80.0 release notes.

The addition of the new libgirepository library version should be fairly straightforward; it's just a new parallel installable major version of a library.

The changes to GLib's build are a little more involved: in order to generate the GLib introspection data in-tree, GLib now needs to be built once without introspection support, then gobject-introspection needs to be built (to make `g-ir-scanner` available, then GLib needs to be re-built with introspection support so that it can use the scanner to generate the GLib GIR files.

Somewhat merging gobject-introspection into GLib

What does this mean for me? GLib

What does this mean for me? GLib

- Now have a bootstrap cycle between GLib and gobject-introspection
- GLib needs gobject-introspection in order to build its own bindings
- We can (and do) now use gi-docgen in GLib

For us as the people who maintain GLib, this makes our CI and build system somewhat more complex, as we can now have to be careful about providing overrides for tools like `gi-compile-repository`, and potentially building gobject-introspection in CI. On the flipside, it means we have been able to port to gi-docgen, so the GLib documentation on docs.gtk.org matches all the other modern documentation.

Somewhat merging gobject-introspection into GLib

What's next?

What's next?

- More testing and improvements to libgirepository
- Now up to about 70% of functions
- Continue improving gi-docgen docs in GLib (help appreciated)
- Introspection is going to get louder (see Emmanuele's talk)

For now, this is the end of the big rearrangements, and any work we do on this will be stabilisation. For example, adding more unit tests (libgirepository has historically not had many).

Porting the GLib documentation to gi-docgen has been done at a high level, but there are still a lot of individual doc comments which need their link syntax porting to gi-docgen and their formatting polishing. Help is very welcome on that, and it's work which is very suitable for drive-by contributors, or people who want to improve just the bit of documentation they're reading at the moment. Please get in touch.

The scanner side of gobject-introspection is now going to get louder and more opinionated, in an effort to improve the introspectability of things. But you'll have to watch Emmanuele's talk from yesterday to hear more about that.