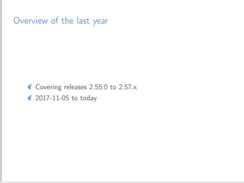
2018-07-06

GLib: What's new and what's next?

25 minutes allocated. 15 minutes for talk, 10 minutes for questions.

Hello. This is going to be a talk in which I present a lot of work done by other people.



Overview of the last year

- Covering releases 2.55.0 to 2.57.x
- 2017-11-05 to today

I'll start by giving a quick overview of what's happened in the year(ish), from the 2.55.0 release (the start of the previous development cycle) to now. We're at the 2.57.2 release now.

Meson migration: feature parity

The Meson port started in the 2.53.4 release. It's now reached rough feature parity with the autotools build. I'll give more details later about the future plans for removing autotools and switching entirely to Meson. Xavier Claessens and Nirbheek Chauhan deserve a lot of credit for doing the work here.

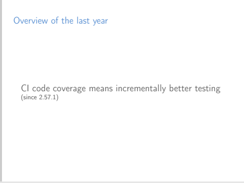
Move to GitLab brings easier contributions
(since May 2018)

As some may have noticed, GNOME has ported to GitLab, which has simplified the contribution process for GLib quite a lot. More on that later. Many thanks to Carlos Soriano for that.

GitLab CI means much better cross-platform support
(since 2.56.0)

One of the big things GitLab brings is easy continuous integration (CI). Since the 2.56.0 release, we've gone from GLib being tested with `make check` whenever the maintainers remember to run it locally — to having most of the tests run on multiple platforms for every merge request (MR), and not allowing merges if the tests fail.

Some tests are still known to fail on some platforms, and there's still a fair amount of work to do to clean things up, but this is a huge improvement. It's already caught platform regressions which would otherwise have been missed for a year.



Overview of the last year

CI code coverage means incrementally better testing
(since 2.57.1)

As part of the CI, we generate code coverage reports, and can use those to request changes to MRs which aren't adequately tested.

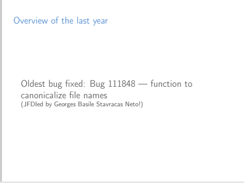
A lot of this stuff is textbook 'how to use GitLab properly', but it makes a proportionally bigger difference to GLib because we need to support platforms which the maintainers don't have regular access to (Windows, macOS, *BSD), and we have a large API surface area to not regress on.

GLib: What's new and what's next?

└─Overview of the last year

75 documentation fixes

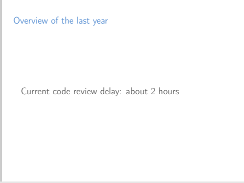
GLib's documentation has never been its strongest point. We've merged a lot of small documentation fixes from a number of contributors recently, a lot of which had been languishing in Bugzilla for years. This doesn't make the documentation perfect, but it's a step in the right direction. Submitting a documentation patch (or filing an issue about some documentation which is missing, unclear, or wrong) is a great way to make an easy and useful contribution to GLib.



Overview of the last year

Oldest bug fixed: Bug 111848 — function to canonicalize file names
(JFDIed by Georges Basile Stavracas Neto!)

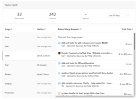
The oldest bug fixed recently dated from 2003, so was well on the way to legally being an adult before Georges squashed it. Thanks Georges!



Current code review delay: about 2 hours

One of the nice features of GitLab is that it gives you stats which you can gamify development with. We don't actually review all MRs in 2 hours: it's more like a bimodal distribution where small MRs get merged within the hour, and larger ones hang around for a few days — but it's an eye-grabbing statistic!

Overview of the last year



Pipeline Health

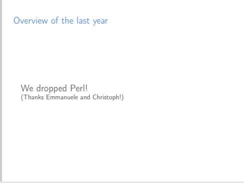
32 New Issues | 242 Commits | - Deploys | Last 30 days

Stage	Median
Issue	Not enough data
Plan	Not enough data
Code	about 22 hours
Test	18 minutes
Review	about 2 hours
Staging	Not enough data
Production	Not enough data

Related Merge Requests	Total Time
Time until first merge request	
Add unit tests for glib-mkenums and resolve #1360 !136 · Opened 6 days ago by Philip Withnall	3 mins
Resolve "g_assert_cmpfloat trips -Wdouble-promotion ... !127 · Opened 12 days ago by Philip Chimento	3 hrs 10 mins
Add unit tests for GMountOperation !124 · Opened 14 days ago by Philip Withnall	1 min
goption: Reject group options specified with three dashes !108 · Opened 13 days ago by Nathan Miller	1 hr 50 mins
glib-compile-resources: Clarify --help output for --sour... !81 · Opened 25 days ago by Philip Withnall	1 day 19 hrs
g_clear_handle_id: don't accept NULL clear_func	10 mins

Figure: GitLab cycle analytics (via GitLab, as of 2018-07-03)

Here are those cycle analytics. As we move through a couple of development cycles with GitLab, we'll get more accurate statistics, and the planning statistics will start to be calculated.

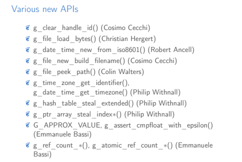


Overview of the last year

We dropped Perl!
(Thanks Emmanuele and Christoph!)

GLib used to use Perl for various of its tools (like `glib-mkenums`) and parts of the build process. Now we don't! We decided to do this because nobody wants to work in Perl any more, and because if we're switching to Meson, we pick up a required Python dependency — so might as well use that instead of Perl.

Thanks to Emmanuele for starting the process, and Christoph Reiter for completing it.

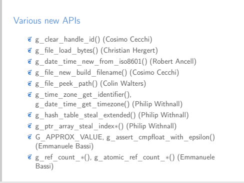


Various new APIs

- g_clear_handle_id() (Cosimo Cecchi)
- g_file_load_bytes() (Christian Hergert)
- g_date_time_new_from_iso8601() (Robert Ancell)
- g_file_new_build_filename() (Cosimo Cecchi)
- g_file_peek_path() (Colin Walters)
- g_time_zone_get_identifier(), g_date_time_get_timezone() (Philip Withnall)
- g_hash_table_steal_extended() (Philip Withnall)
- g_ptr_array_steal_index*() (Philip Withnall)
- G_APPROX_VALUE, g_assert_cmpfloat_with_epsilon() (Emmanuele Bassi)
- g_ref_count_*(), g_atomic_ref_count_*() (Emmanuele Bassi)

There have been over 220 changes since 2.57.1 (it's about time to make a 2.57.2 release). Here highlights of the most interesting of those changes to users of GLib, starting with the new APIs.

- g_clear_handle_id() — A clear function for integer handles, such as GSource IDs or bus name IDs.
- g_file_load_bytes() — Load a GFile into a GBytes.
- g_date_time_new_from_iso8601() — Parse an ISO-8601 string into a GDateTime.
- g_file_new_build_filename() — A combination of g_build_filename() and g_file_new_for_path().
- g_file_peek_path() — A version of g_file_get_path() which returns a **const** value.



Various new APIs

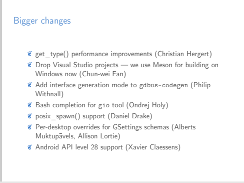
- g_clear_handle_id() (Cosimo Cecchi)
- g_file_load_bytes() (Christian Hergert)
- g_date_time_new_from_iso8601() (Robert Ancell)
- g_file_new_build_filename() (Cosimo Cecchi)
- g_file_peek_path() (Colin Walters)
- g_time_zone_get_identifier(), g_date_time_get_timezone() (Philip Withnall)
- g_hash_table_steal_extended() (Philip Withnall)
- g_ptr_array_steal_index∗() (Philip Withnall)
- G_APPROX_VALUE, g_assert_cmpfloat_with_epsilon() (Emmanuele Bassi)
- g_ref_count_∗(), g_atomic_ref_count_∗() (Emmanuele Bassi)

- g_time_zone_get_identifier(), g_date_time_get_timezone() — Ways to get more information about timezones.
- g_hash_table_steal_extended() — A version of g_hash_table_steal() which actually returns the stolen key and value to the caller.
- g_ptr_array_index∗() — Steal the pointer at the given index from the array and return it.
- G_APPROX_VALUE, g_assert_cmpfloat_with_epsilon() — A version of g_assert_cmpfloat() with an epsilon.
- g_ref_count_∗(), g_atomic_ref_count_∗() — Reference counting primitives for use in your own structs, rather than reinventing your own reference counting.

2018-07-06

GLib: What's new and what's next?

└─Bigger changes



Bigger changes

- get_type() performance improvements (Christian Hergert)
- Drop Visual Studio projects — we use Meson for building on Windows now (Chun-wei Fan)
- Add interface generation mode to `gdbus-codegen` (Philip Withnall)
- Bash completion for `gio` tool (Ondrej Holy)
- posix_spawn() support (Daniel Drake)
- Per-desktop overrides for GSettings schemas (Alberts Muktupāvels, Allison Lortie)
- Android API level 28 support (Xavier Claessens)

Here are some of the bigger changes since 2.55.0 which don't result in new API in GLib:

Content type fixes Various fixes to make GContentType integrate better with macOS.

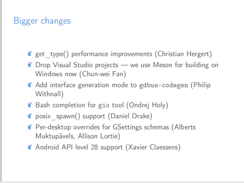
Static linking support GIO modules can now be statically linked into a build.

`typeof()` fixes Improved compile time type checking with g_object_ref().

Windows network monitor An implementation of GNetworkMonitor on Windows.

`GBytes` performance Fewer copies when taking slices of a GBytes.

Nominative case month names Some Slavic languages change the case of month names when they're given together with a day. We now support that when formatting dates.



Bigger changes

- get_type() performance improvements (Christian Hergert)
- Drop Visual Studio projects — we use Meson for building on Windows now (Chun-wei Fan)
- Add interface generation mode to `gdbus-codegen` (Philip Withnall)
- Bash completion for `gio` tool (Ondrej Holy)
- posix_spawn() support (Daniel Drake)
- Per-desktop overrides for GSettings schemas (Alberts Muktupāvels, Allison Lortie)
- Android API level 28 support (Xavier Claessens)

Splice performance Larger buffer sizes give improved splice performance for NFS.

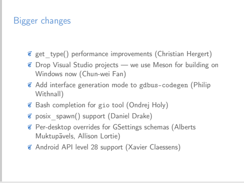
kqueue file monitor A complete rewrite of the kqueue file monitor for *BSD to improve performance and correctness, and eliminate code.

`get_type()` performance Optimisations on the fast path in get_type() functions to reduce overhead.

Visual Studio We've dropped our Visual Studio solution files, as all the GLib Windows developers are now using Meson to generate them.

GDBus interface generation `gdbus-codegen` can now generate just header files containing interface descriptions.

`gio` bash completion Bash completion support for the `gio` utility.


Bigger changes

- get_type() performance improvements (Christian Hergert)
- Drop Visual Studio projects — we use Meson for building on Windows now (Chun-wei Fan)
- Add interface generation mode to `gdbus-codegen` (Philip Withnall)
- Bash completion for `gio` tool (Ondrej Holy)
- posix_spawn() support (Daniel Drake)
- Per-desktop overrides for GSettings schemas (Alberts Muktupāvels, Allison Lortie)
- Android API level 28 support (Xavier Claessens)

`posix_spawn()` support
: The g_spawn_*() functions now support a fast path which uses posix_spawn() rather than fork()+exec(), given some preconditions.

GSettings overrides
: Override files for GSettings keys can now be customised for different desktops.

Android API
: We support Android API level 28; more generally, we now have more testing against Android.

Who's contributed?

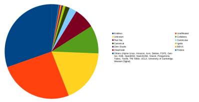


Figure: GLib contribution affiliation since 2.55.0 (29 affiliands, 936 commits, via `git shortlog`, as of 2018-07-03)

I think the affiliation of the contributors to a project is a quick way of assessing the health of the project, in terms of bus factor. I tried to work out the affiliation of each contributor since 2.55.0. 'Unaffiliated' means it's most likely a contribution in someone's personal time. 'Unknown' means it could be a corporate or personal contribution. About 25% of our contributions are unaffiliated, and about 15% are unknown. 90% of contributions are either from Endless, unaffiliated, unknown, Collabora or Red Hat.

The maintainers (Matthias, Emmanuele and me) are from Red Hat, unaffiliated, and Endless; which I think is a fairly safe state to be in.

GLib: What's new and what's next?

└─Development activity

Development activity

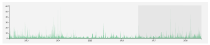


Figure: Commit count since 2013 (via GitLab, as of 2018-07-05)

I thought it would be interesting to look at how development activity has changed over time. Here we have a graph of commits per year. I don't think it's relevant to look at the older years (since people didn't split commits up properly, etc.). What's interesting is that we've got almost as many commits so far in 2018 as we had in all of 2017, and already more than in 2016 or 2015.

This second graph is from GitLab, and while it's a little harder to see the difference, it does look like the commit density in 2018 has increased compared to 2017.

2018-07-06

GLib: What's new and what's next?

└─Bug count before GitLab

Bug count before GitLab

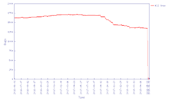


Figure: Open bug count on Bugzilla, as of 2018-07-05

Here's a chart of the number of open bugs, from Bugzilla, up to the present. The massive drop off at the end is our transition to GitLab. There was a fairly big effort to triage and close old bugs in the second half of 2017, in preparation for migrating to GitLab. Since then, things have calmed down a bit.

Unfortunately, GitLab doesn't provide a similar chart (as far as I know).

We're currently at around 1300 issues.

CI charts

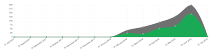
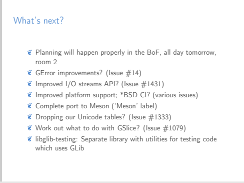


Figure: Successful (green) vs total CI pipelines in the last year (via GitLab, as of 2018-07-05)

GitLab does provide lots of charts of the CI system, though. Here we've got a look at how long each of the last 30 commits has taken to build and test in CI. We average around 15 minutes, which isn't great, but there are probably some easy improvements to make about not downloading dependencies on each build.

The second chart shows the successful CI builds (green) against the total, so the difference is the number of failed builds. We have had a lot of builds fail due to the infamous GitLab rackattack ban, but that's recently (hopefully?) been fixed, so hopefully this graph will improve a little.

In any case, CI failures have not been a major obstacle to development.

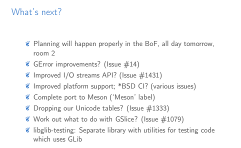


What's next?

- Planning will happen properly in the BoF, all day tomorrow, room 2
- GError improvements? (Issue #14)
- Improved I/O streams API? (Issue #1431)
- Improved platform support; *BSD CI? (various issues)
- Complete port to Meson ('Meson' label)
- Dropping our Unicode tables? (Issue #1333)
- Work out what to do with GSlice? (Issue #1079)
- libglib-testing: Separate library with utilities for testing code which uses GLib

So what's planned next? Nothing is planned concretely — that's going to happen in the birds of a feather (BoF) session all day tomorrow. Anyone with a stake in GLib is welcome to pop in, or to talk to me about it beforehand.
The list of ideas here are what I am personally thinking about putting forward for the 2.60 release.

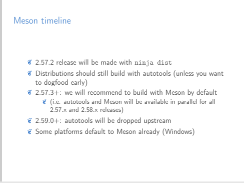
`GError`	People have been asking for ways to extend GError for a while; this deserves some thought.
I/O streams	Improvements to the I/O stream APIs to support zero-copy transfers where possible, and improve end-of-file handling and blocking notification.
Platform support	Additions and improvements to our CI builders; fixing various tests on non-Linux platforms and enabling tracking test failures instead of ignoring them.



What's next?

- Planning will happen properly in the BoF, all day tomorrow, room 2
- GError improvements? (Issue #14)
- Improved I/O streams API? (Issue #1431)
- Improved platform support; *BSD CI? (various issues)
- Complete port to Meson ('Meson' label)
- Dropping our Unicode tables? (Issue #1333)
- Work out what to do with GSlice? (Issue #1079)
- libglib-testing: Separate library with utilities for testing code which uses GLib

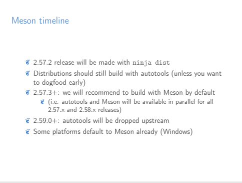
Meson port	Finishing off the Meson port and dropping autotools (see the following slide).
Unicode tables	Potentially dropping our internal Unicode tables in favour of platform-wide ones which are almost universally installed and loaded already.
`GSlice`	Potentially dropping GSlice, since other allocators have improved since it was written.
libglib-testing	Providing more standard test harnesses for code written using GLib. For example, a standard way of testing object property implementations, or I/O streams.



Meson timeline

- 2.57.2 release will be made with `ninja dist`
- Distributions should still build with autotools (unless you want to dogfood early)
- 2.57.3+: we will recommend to build with Meson by default
  - (i.e. autotools and Meson will be available in parallel for all 2.57.x and 2.58.x releases)
- 2.59.0+: autotools will be dropped upstream
- Some platforms default to Meson already (Windows)

1. Starting from 2.57.2, create release tarballs with ‘ninja dist‘, but recommend that distributions still build with autotools (unless they want to dogfood with Meson early).
2. From 2.57.3, switch to recommending that distributions build with Meson.
3. Starting from 2.59.0 (the actual start of next dev cycle), drop autotools completely; assuming that steps 1 and 2 have gone OK.

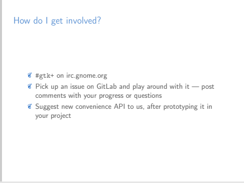


Meson timeline

- 2.57.2 release will be made with `ninja dist`
- Distributions should still build with autotools (unless you want to dogfood early)
- 2.57.3+: we will recommend to build with Meson by default
  - (i.e. autotools and Meson will be available in parallel for all 2.57.x and 2.58.x releases)
- 2.59.0+: autotools will be dropped upstream
- Some platforms default to Meson already (Windows)

I want to make sure that distributions only start building GLib using Meson for their unstable/development releases, rather than for stable releases. There have only recently been bugs about code which was compiled with autotools not being built with Meson (the FAM file monitor comes to mind), which doesn't give me enough confidence to jump to recommending building with Meson right yet.
Note that on some platforms, we may drop support for autotools early. Notably the maintainers of the GLib builds on Windows have already fully switched to Meson, and there is a merge request open about making configure error out on Windows.

I could imagine the same happening for other platforms where we're in direct contact with the small set of packagers for those platforms (for example, MacPorts and the *BSDs).



How do I get involved?

- #gtk+ on irc.gnome.org
- Pick up an issue on GitLab and play around with it — post comments with your progress or questions
- Suggest new convenience API to us, after prototyping it in your project

Talk to us! GLib apparently has a reputation for being big and scary. It might be big, but it's just code. We'd be happy to see your contributions and get them accepted.