Feedback from downstreams

Philip Withnall

2015-01-21

Client 1: Medical devices

- Training
- GTK+ customisation
- Using: GLib, GTK+ on Wayland, GStreamer, gtkmm

Source: *Jay Reed, CC-BY-SA 2.0*

Client 2: In-vehicle infotainment (IVI)

- Code review
- Licencing and open sourcing
- Poking upstreams
- Using: GLib, Clutter, D-Bus, gtk-doc

Source: *Land Rover Our Planet, CC-BY-ND 2.0*

Client 1: Problems

- SVGs in `GtkImages` don't scale with the widget
- Some properties are confusing, such as `GtkCellRenderText:editable[-set]`
- Really want a way to make `GtkBuilder` behave like a factory
- Gtkmm's documentation, especially Devhelp integration, is poor
- Documentation of widget event masks and event propagation is opaque
- Want a horizontal accordion revealer widget
- No legal or tooling issues

Client 1: Summary

- Various specific problems or misunderstandings of the API
- Deprecations should always be documented with a rationale and replacement API — preferably also a porting guide
- New functionality needs documenting: `GtkBuilder` templates were blogged about extensively, but are given a single sentence in the manual
- New widget: Accordion revealer
- No legal or tooling issues

Client 2: Legal

- Choosing a licence to release under
- Checking for proprietary code
- Licence hygiene
- Project naming!

Client 2: Infrastructure

- git
- Bugzilla
- Website and wiki
- Code review system
- Packaging system

Client 2: Information

- Is the project even interesting?
- Marketing the project
- Producing documentation
- Limitations and known issues

Client 2: Interaction with upstreams

- Insulated by Collabora
- Lacking documentation
- Worry about API deprecations

Client 2: Module setup problems

- Project bootstrapping and directory layout best practices
- Bootstrapping problems with gettext, intltool and gtk-doc
- Indicating licencing in files
- Knowing which files to put in git
- Parallel installability of libraries
- Total confusion over LT versioning

Client 2: Coding conventions problems

- Namespacing and GObject modularity
- Little confusion over use of GError
- Repeated, inefficient use of GList
- Confusion between sync and async; over-use of threads

Client 2: Memory management problems

- Massive over-use of `g_strdup()`
- Persistent confusion over `g_free(NULL)`

Client 2: Tooling problems

- No compiler, GIR or gtk-doc warnings enabled
- No gtk-doc tests enabled
- Confusion over documentation generation with `gdbus-codegen`

Client 2: Unit testing problems

- Tests were not automatable
- Tests were not hooked up to CI or other tooling
- APIs were untestable by design

Client 2: API-specific problems

- JSON reader function pairing on different code paths
- No validation of resource files (XML, JSON schemas)
- Confusion over GSettings schema internationalisation
- Confusion over relocatable GSettings schemas
- Confusion over GDBus vs. libdbus and libdbus-glib
- Poor D-Bus API design in places
- Problems with D-Bus build system integration
- Problems building SQL and SPARQL queries
- Use of POSIX APIs incompatible with GLib (e.g. `sigaction()`, `system()`, `sleep()`)

Client 2: Threading problems

- Threads were used instead of async calls
- Reinvented thread pools
- Implications on main contexts ignored

Client 2: File system access problems

- Repeated usage of sync APIs
- Problems building file paths
- No validation of file sandboxing

Client 2: Summary

- Various bootstrapping and build system integration problems
- Project layout, namespacing, versioning and version control conventions
- Confusion between sync, async and threading
- Various specific problems or misunderstandings of the API
- No tooling warnings enabled — should be on by default
- APIs which could be abused, were abused (SPARQL queries)
- Problems with incompatible POSIX APIs
- Need to think about file sandboxing as part of the API

Miscellany

CC-BY-SA 4.0