]pexip[

The pexLGPL bundle

GStreamer Conference 2025

Tulio Beloqui <tulio@pexip.com>

Everything started with gst-full..

- Our goal was to have a single dependency that contains and provides all the LGPL code and symbols that we build.
- As a technical benefit, this will ease the shipping of our product -> we *just* pack our binaries plus this bundle.
- It allows us to comply with the license, we can opensource the way we create the bundle, and since all of our LGPL forks are already public on github, win win.
- Everything started with gst-full.. we have been using it for a while, so starting there was the first step.
- We defined a "lgpl dep" in a meson project, and started using it as the single dep that which provide all the lgpl symbols our binaries needed to run.
- gst-full was almost enough!
  - Conflicting double instances of glib! (json-glib and our pexrtmpserver depends on glib)
  - We have our own **os packages** for pango and cairo with dependencies which we couldn't use the wrapper system
  - We have more dependencies than gstreamer

The pexLGPL bundle

- The idea started very simple, we build all of the LGPL code we use, as *static libraries* and merge them on a single dynamic one.

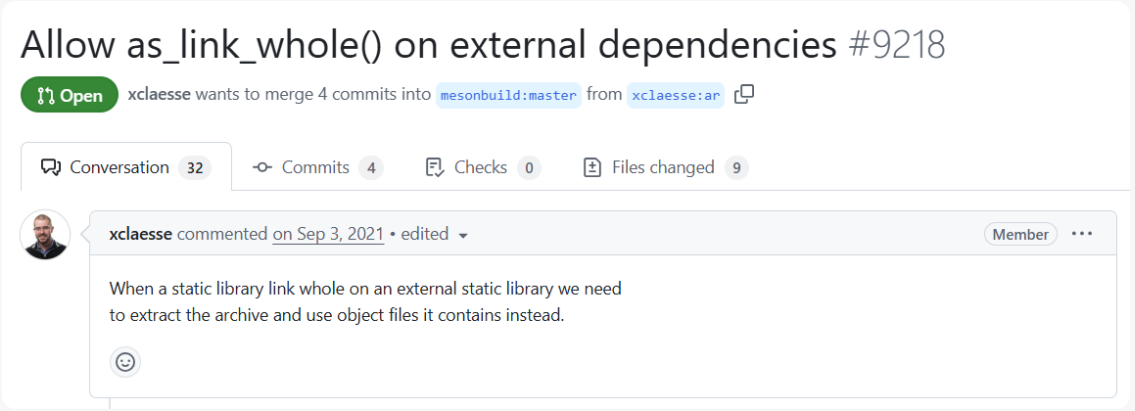
- This brought us two major challenges:
  - doing the merge of the libraries
  - get the symbols to be exported!

- For the **first** challenge, we had to rewrite a bit an existing patch in meson that provides us to link static libraries "as-whole", which meant every single static library we put in, will pull every symbol from it, into our bundle.

  - pull in **only** the library we are linking as-whole (not its own dependencies! But we had to reference them later!)
  - **pex-hack:** static and link-as-whole only for libraries inside the **prefix folder!** (to avoid pulling in system libraries!)

  - NOTE: Meson has the link_whole() functionality but for meson projects you have built yourself! (internal dependencies) It doesn't work on external dependencies pulled in from pkg-config.

- For the **second**, we had to deal with compiler and linker flags... a lot of them...



https://github.com/mesonbuild/meson/pull/9218

Writing a meson project

- The meson project started to get in shape; by defining a list of the libraries we will include in this bundle:
  - glib
  - gobject
  - gio
  - gstreamer-1.0, gstreamer-base, etc etc
- Then a list of **gstreamer plugins** we build for the selected platform.
- Create pkg-config file...
- Includeing all the libraries include dirs, their compiler and linker flags!
- Use the "pexlgpl_dep" as the dependency for all

Compilers & Linkers

- We had to create a definition map file for the supported compilers, to make sure we understand what we are exporting and from *where* (what library?!)

- But.. MSVC linker and its constraints made us go back into the libraries we were dealing with and revisit their export macros.
- In msvc, you tag a symbol to be exported at build-time (extern vs dllexport/dllimport), so later the linker knows if it has to export it or not.

- In other compilers, (gcc and clang) you can get away with it by using the "extern" keyword, which works for both static and dynamic linking <3

Whats next?

- Our patches are here
  - https://github.com/pexip/meson
  - https://github.com/pexip/gstreamer

- We plan to open source the meson project to build the bundle !

Thanks!

- Contact
  - tulio@pexip.com
  - tulio @ GStreamer Community in Matrix

Come to talk to us for more details :)