GStreamer FAQ (0.10.36)

This is the FAQ for GStreamer, a multimedia framework. Questions and answers range from general information to deep-down-and-dirty compilation issues.

1. Getting Started

So you're eager to get started learning about GStreamer. There's a few ways you can get started.

- If you want to learn by reading about it, start with *General*
- If you'd rather learn by trying it out, start with *Getting GStreamer*
- If you want to live on the bleeding edge and develop and use git, see *Building GStreamer from git*

2. General

Q: Is GStreamer a media player ?

A: No, GStreamer is a development framework for creating applications like media players, video editors, streaming media broadcasters and so on. That said, very good media players can easily be built on top of GStreamer especially when using the high-level object called playbin.

Q: Why is GStreamer written in C ? Why not C++/Objective-C/... ?

A: We like C. Aside from "personal preference", there are a number of technical reasons why C is nice in this project:

- C is extremely portable.
- C is fast.
- It is easy to make language bindings for libraries written in C.
- The GObject object system provided by GLib implements objects in C, in a portable, powerful way. This library provides for introspection and runtime dynamic typing. It is a full OO system, but without the syntactic sugar. If you want sugar, take a look at Vala (http://live.gnome.org/Vala).
- Use of C integrates nicely with Gtk+ and GNOME. Some people like this a lot, but neither Gtk+ nor GNOME are required by GStreamer.

So, in closing, we like C. If you don't, that's fine; if you still want to help out on GStreamer, we always need more language binding people. And if not, don't bother us; we're working :-)

Q: What applications are available for GStreamer ?

A: Many media player applications have chosen GStreamer for their backend. Also a couple of media format conversion tools have been written using the powers of GStreamer. With the advent of GStreamer-0.10 several media editing applications have been started.

For a list of projects, look at the application list (http://gstreamer.freedesktop.org/apps/) on the GStreamer project website.

Q: Does GStreamer support the format of my media files?

A: GStreamer aims to support every format imaginable, but that doesn't mean the developers have managed to achieve that aim yet. If a GStreamer enabled application doesn't play back your files, you can help us solve that problem by filing an enhancement request bug (http://bugzilla.gnome.org) for that format. If you have it, please provide:

- links to other players, preferably Open Source and working on Unix
- links to explanations of the format.
- ways to obtain mediafiles in that format to test.

Q: What are the exact licensing terms for GStreamer and its plugins ?

A: All of GStreamer, including our own plugin code, is licensed under the GNU LGPL (http://www.gnu.org/licenses/lgpl.html) license. Some of the libraries we use for some of the plugins are however under the GPL, which means that those plugins can not be used by a non-GPL-compatible application.

As part of the GStreamer source download you find a file called LICENSE_readme in gst-plugins package. That file contains information in the exact licensing terms of the libraries we use. As a general rule, GStreamer aims at using only LGPL or BSD licensed libraries if available and only use GPL or proprietary libraries where no good LGPL or BSD alternatives are available.

From GStreamer 0.4.2 on, we implemented a license field for all of the plugins, and in the future we might have the application enforce a stricter policy (much like tainting in the kernel).

Q: Is GStreamer a sound server ?

A: No, GStreamer is not a soundserver. GStreamer does however have plugins supporting most of the major soundservers available today, including pulseaudio, ESD, aRTSd, Jack and others.

Q: Will GStreamer be available for platforms other than Unix ?

A: Depends. Our main target is the Unix platform. It also works on Win32 and Mac OS X, but it may still be a bit challenging to get everything up and running. That said, interest has been expressed in

porting GStreamer to other platforms and the GStreamer core team will gladly accept patches to accomplish this.

Q: What is GStreamer's relationship with the GNOME community ?

A: While GStreamer is operated as an independent project, we do have a close relationship with the GNOME community. Many of our hackers consider themselves also to be members of the GNOME community. GStreamer is officialy bundled with the GNOME desktop, as lots of packages (like gnome-media, totem and rhythmbox) are using it. This does not exclude use of GStreamer by other communities at all, of course.

Q: What is GStreamer's relationship with the KDE community ?

A: The GStreamer community wants to have as good a relationship as possible with KDE, and we hope that someday KDE decides to adopt GStreamer as their multimedia API (planned for KDE 4). There have been contacts from time to time between the GStreamer community and KDE and we do already have support for the aRTSd sound server used by KDE. Also, some of the KDE hackers have created Qt bindings of GStreamer, made a simple video player and using it in some audio players (JuK and AmaroK).

Q: I'm considering adding GStreamer output to my application...

A: That doesn't really make sense. GStreamer is not a sound server, so you don't output directly to GStreamer, and it's not an intermediate API between audio data and different kinds of audio sinks. It is a fundamental design decision to use GStreamer in your app; there are no easy ways of somehow 'transfering' data from your app to GStreamer. Instead, your app would have to use or implement a number of GStreamer elements, string them together, and tell them to run. In that manner the data would all be internal to the GStreamer pipeline.

That said, it is possible to write a plugin specific to your app that can get at the audio data.

3. Dependencies

Q: Why are there so many dependencies ?

A: Making a full-featured media framework is a huge undertaking in itself. By using the work done by others, we both reduce the amount of redundant work being done and leave ourselves free to work on the architecture itself instead of working on the low-level stuff. We would be stupid not to reuse the code others have written.

However, do realize that in no way you are forced to have all dependencies installed. None of the core developers has all of them installed. GStreamer has only a few obligate dependencies : GLib 2.0, liboil, and very common stuff like glibc, a C compiler, and so on. All of the other dependencies are optional.

So, in closing, let's rephrase the question to "Why are you giving me so many choices and such a rich environment ? "

Q: Does GStreamer use GTK+ 1.2/GLib 1.2 or GLib 2.0 ?

A: Since the 0.3.3 release of GStreamer, we use GLib 2.0 as the core library for GStreamer, which features a move of GObject from GTK+ 2.0 to GLib 2.0. If you want to compile using GTK+ 1.2/GLib 1.2, you need to get the 0.3.1 or earlier release. It is of course not supported.

Q: Does GStreamer offer support for DVD decoder cards like dxr2/3 ?

A: We do have support for the dxr3, although dxr2 support is unknown. GStreamer can easily accommodate hardware acceleration by writing new device-specific elements.

Q: Is GStreamer X independent ?

A: Yes, we have no X dependency in any of our core modules. There are GStreamer applications that run fine without any need for X. However, until our Linux Framebuffer or libsvga plugin is ready, you will not be able to play videos without X. In the future, there will probably be lots of different output plugins for video available.

Q: What is GStreamer's position on efforts such as LADSPA ?

A: GStreamer actively supports such efforts, and in the case of *LADSPA* (http://ladspa.org/), we already have a wrapper plugin. This wrapper plug-in detects the LADSPA plugins present on your system at register time.

Q: Does GStreamer support MIDI ?

A: Not yet. The GStreamer architecture should be able to support the needs of MIDI applications very well however. If you are a developer interested in adding MIDI support to GStreamer we are very interested in getting in touch with you.

Q: Does GStreamer depend on GNOME ?

A: No. But many of the applications developed for GStreamer do, including our sample applications. There is nothing hindering people from developing applications using other toolkits however and we would happily help promote such efforts. A good example of an application using GStreamer, but which is not using GNOME is the *Mozstreamer* (http://mozstreamer.mozdev.org) which uses Mozilla XUL.

4. Getting GStreamer

Q: How do I get GStreamer ?

A: Generally speaking, you have three options, ranging from easy to hard :

- distribution-specific packages
- source tarballs
- git

Q: There seem to be different GStreamer versions, like 0.8 and 0.10? What's up with that?

A: GStreamer-0.8 and GStreamer-0.10 are the main version 'series' currently in use. For all practical purposes you should think of them as two completely different libraries which just happen to have a similar name. They can be installed in parallel and are completely independent.

For the 0.8 version you will need the 0.8 plugins and bindings (gst-plugins 0.8.x, gst-ffmpeg 0.8.x, gst-python 0.8.x etc.), while for the 0.10 version you will need the 0.10 plugins and bindings (ie. gst-plugins-base 0.10.x, gst-plugins-good 0.10.x, gst-plugins-ugly 0.10.x, gst-plugins-bad 0.10.x, gst-ffmpeg 0.10.x, gst-python 0.10.x). The micro version for each main version does not have to match exactly, only the major versions needs to be the same (ie. it may be that the current gst-plugins-good version is 0.10.6 and the current GStreamer core version is 0.10.13). GStreamer-0.10 will not see or use any of the GStreamer-0.8 plugins and vice versa.

All GStreamer command line tools are suffixed with their main version, e.g. gst-launch-0.8 and gst-launch-0.10, or gst-inspect-0.8 and gst-inspect-0.10. The corresponding GStreamer command line tools without a suffix (e.g. gst-launch) will default to the highest major version.

Applications will use either GStreamer-0.8 or GStreamer-0.10, since the 0.8 and 0.10 API/ABI are not compatible.

Odd-numbered versions such as 0.9.x, 0.11.x, etc. are unstable developer releases that should generally not be used.

Q: So which GStreamer version should I get?

A: You should download GStreamer-0.10. GStreamer-0.8 is not developed any longer and has not been maintained for almost two years (you may still find it packaged for your disto though, but that's most likely for legacy applications).

Q: How can I install GStreamer from source ?

A: We provide tarballs of our releases on our own site, at http://gstreamer.freedesktop.org/src/ (http://gstreamer.freedesktop.org/src/)

When compiling from source, make sure you specify PKG_CONFIG_PATH correctly when building against GStreamer. For example, if you configured GStreamer with the default prefix (which is /usr/local), then you need to

```
export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig
```

before building gst-plugins.

Q: Are there premade binaries available ?

A: Yes, we currently provide precompiled packages for Red Hat, Debian and Linux Mandrake.

We provide RPMS for Red Hat and Fedora Core through our Apt for rpm page. (http://gstreamer.freedesktop.org/pkg/) We usually support the last 2-3 releases of Red Hat. (RH9 and FC1)

GStreamer is already in Debian experimental, so if you are a debian user you should be able to get the Debian packages from there.

GStreamer is also in Mandrake Cooker so if you are using a recent release of Linux Mandrake you should be able to get GStreamer from there. To learn howto get packages from Mandrake cooker the Mandrake cooker page has more info (http://www.mandrakelinux.com/en/cookerdevel.php3)

For other RPM based distributions we recommend getting the source tarball and doing 'rpm -ta gstreamer-0.X.X' to create rpms. We keep our SPEC file constantly up to date so you should always be able to build GStreamer rpms from a tarball. The SPEC file is aimed at Red Hat however, so there might be some need for you to edit the Requirements list if your distribution name these packages differently.

Q: Why don't you provide premade binaries for distribution XY ?

A: GStreamer is run on a volunteer basis. The package that are provided are made by non-paid people who do this on their own time. The distributions we support with binaries are the distributions that we have people who have volunteered to make binaries for. If you are interested in maintaining GStreamer binaries for other distributions or Unices we would be happy to hear from you. Contact us through the GStreamer-devel mailing list.

Q: I am having trouble compiling GStreamer on my LFS installation, why ?

A: If you are running LFS our basic opinion is that you should be knowledgeable enough to solve any build issues you get on your own. Being volunteered based we can't promise support to anyone of course, but are you using LFS consider yourself extra unsupported. We neither can or want to know enough, about how your unique system is configured, to be able to help you. That said, if you come to the #gstreamer channel on irc.openprojects.net we might of course be able to give you some general hints and pointers.

Q: How do I get GStreamer through git ?

A: see this page : http://gstreamer.freedesktop.org/dev/ for git access. (anonymous and developer)

5. Using GStreamer

Q: Ok, I've installed GStreamer. What can I do next ?

A: First of all, verify that you have a working registry and that you can inspect them by typing

```
$ gst-inspect fakesrc
```

This should print out a bunch of information about this particular element. If this tells you that there is "no such element or plugin", you haven't installed GStreamer correctly. Please check how to get GStreamer If this fails with any other message, we would appreciate a bug report.

It's time to try out a few things. Start with gst-launch and two plug-ins that you really should have : fakesrc and fakesink. They do nothing except pass empty buffers. Type this at the command-line :

```
$ gst-launch -v fakesrc num-buffers=3 ! fakesink
```

This will print out output that looks similar to this :

```
RUNNING pipeline ...
fakesrc0: last-message = "get      ******* (fakesrc0:src)gt; (0 bytes, 0) 0x8057510"
fakesink0: last-message = "chain   ******* (fakesink0:sink)lt; (0 bytes, 0) 0x8057510"
fakesrc0: last-message = "get      ******* (fakesrc0:src)gt; (0 bytes, 1) 0x8057510"
fakesink0: last-message = "chain   ******* (fakesink0:sink)lt; (0 bytes, 1) 0x8057510"
fakesrc0: last-message = "get      ******* (fakesrc0:src)gt; (0 bytes, 2) 0x8057510"
fakesink0: last-message = "chain   ******* (fakesink0:sink)lt; (0 bytes, 2) 0x8057510"
execution ended after 5 iterations (sum 301479000 ns, average 60295800 ns, min 3000 ns, r
```

(Some parts of output have been removed for clarity) If it looks similar, then GStreamer itself is running correctly.

Q: Can my system play sound through GStreamer ?

A: You can test this by trying to play a sine tone. For this, you need to link the audiotestsrc element to an output element that matches your hardware. A (non-complete) list of output plug-ins for audio is

- osssink for OSS output
- esdsink for ESound output
- artsdsink for aRTs output (not ported to 0.10 yet)
- alsasink for ALSA output
- alsaspdifsink for ALSA S/PDIF output
- jackaudiosink for JACK output

First of all, run gst-inspect on the output plug-in you want to use to make sure you have it installed. For example, if you use OSS, run

```
$ gst-inspect osssink
```

and see if that prints out a bunch of properties for the plug-in.

Then try to play the sine tone by running

```
$ gst-launch audiotestsrc ! audioconvert ! audioresample ! osssink
```

and see if you hear something. Make sure your volume is turned up, but also make sure it is not too loud and you are not wearing your headphones.

In GNOME, you can configure audio output for most applications by running

```
$ gstreamer-properties
```

which can also be found in the start menu (Applications -> Preferences -> Multimedia Systems Selector). In KDE, there is not yet a shared way of setting audio output for all applications; however, applications such as Amarok allow you to specify an audio output in their preferences dialog.

Q: How can I see what GStreamer plugins I have on my system ?

A: To do this you use the gst-inspect command-line tool, which comes standard with GStreamer. Invoked without any arguments,

```
$ gst-inspect
```

will print out a listing of installed plugins. To learn more about a particular plugin, pass its name on the command line. For example,

```
$ gst-inspect volume
```

will give you information about the volume plugin.

Also, if you install the gst-editor package, you will have a graphical plugin browser available, gst-inspect-gui.

Q: Where should I report bugs ?

A: Bug management is now hosted on GNOME's Bugzilla at http://bugzilla.gnome.org, under the product GStreamer. Using bugzilla you can view past bug history, report new bugs, etc. Bugzilla requires you to make an account here, which might seem cumbersome, but allows us to at least have a chance at contacting you for further information, as we will most likely have to.

Q: How should I report bugs ?

A: When doing a bug report, you should at least describe

- your distribution
- how you installed GStreamer (from git, source, packages, which ?)
- if you installed GStreamer before

It also is useful for us if you attach output of the gst-feedback command to your bug report. If you're having problem with a specific application (either one of ours, somebody else's, or your own), please also provide a log of gst-mask by running

```
myapp --gst-mask=-1 > mask.log 2>&1
gzip mask.log
```

(interrupting the program if it doesn't stop by itself) and attach mask.log.gz to your bug report.

If the application you are having problems with is segfaulting, then provide us with the necessary gdb output. See *The GStreamer application I used stops with a segmentation fault. What can I do ?*

Q: How do I use the GStreamer command line interface ?

A: You access the GStreamer command line interface using the command gst-launch. To decode an mp3 and play it through OSS, you could use

```
gst-launch filesrc location=thesong.mp3 ! mad ! audioconvert !
audioresample ! osssink
```

. More examples can be found in the gst-launch man page.

To automatically detect the right codec in a pipeline, try

```
gst-launch filesrc location=my-random-media-file.mpeg ! decodebin !
 audioconvert ! audioresample ! osssink
```

. or

```
gst-launch filesrc location=my-random-media-file.mpeg ! decodebin !
 ffmpegcolorspace ! xvimagesink
```

Something more complicated:

```
gst-launch filesrc location=my-random-media-file.mpeg ! decodebin name=decoder
   decoder. ! ffmpegcolorspace ! xvimagesink
   decoder. ! audioconvert ! audioresample ! osssink
```

We also have a basic media playing plugin that will take care of most things for you. This plugin is called playbin. Try this:

```
gst-launch playbin uri=file:///home/joe/my-random-media-file.mpeg
```

This should play the file if the format is supported, ie. you have all the necessary demuxing and decoding and some output plugins installed.

6. Troubleshooting GStreamer

Q: Some application is telling me that I am missing a plug-in. What do I do ?

A: Well, start by checking if you really are missing the plug-in.

```
gst-inspect (plug-in)
```

and replace (plug-in) with the plug-in you think is missing. If this doesn't return any result, then you either don't have it or your registry cannot find it.

If you're not sure either way, then chances are good that you don't have it. You should get the plug-in and run gst-register to register it. How to get the plug-in depends on your distribution.

- if you run GStreamer using packages for your distribution, you should check what packages are available for your distribution and see if any of the available packages contains the plug-in.
- if you run GStreamer from a source install, there's a good chance the plug-in didn't get built because you are missing an external library. When you ran configure, you should have gotten output of what plug-ins are going to be built. You can re-run configure to see if it's there. If it isn't, there is a good reason why it is not getting built. The most likely is that you're missing the library you need for it. Check the README file in gst-plugins to see what library you need. Make sure to remember to re-run configure after installing the supporting library !
- if you run GStreamer from git, the same logic applies as for a source install. Go over the reasons why the plug-in didn't get configured for build. Check output of config.log for a clue as to why it doesn't get built if you're sure you have the library needed installed in a sane place.

Q: I get an error that says something like (process:26626): GLib-GObject-WARNING **: specified instance size for type 'DVDReadSrc' is smaller than the parent type's 'GstElement' instance size What's wrong ?

A: If you run GStreamer from git uninstalled, it means that something changed in the core that requires a recompilation in the plugins. Recompile the plugins by doing "make clean && make".

If you run GStreamer installed, it probably means that you run the plugins against a different (incompatible) version than they were compiled against, which ususally means that you run multiple installations of GStreamer. Remove the old ones and - if needed - recompile again to ensure that it is using the right version.

Note that we strongly recommend using Debian or RPM packages, since you will not get such issues if you use provided packages.

Q: The GStreamer application I used stops with a segmentation fault. What can I do ?

A: There are two things you can do. If you compiled GStreamer with specific optimization compilation flags, you should try recompiling GStreamer, the application and the plug-ins without any optimization flags. This allows you to verify if the problem is due to optimization or due to bad code. Second, it will also allow you to provide a reasonable backtrace in case the segmentation fault still occurs.

The second thing you can do is look at the backtrace to get an idea of where things are going wrong, or give us an idea of what is going wrong. To provide a backtrace, you should

1. run the application in gdb by starting it with

```
gdb (gst-application)
```

(If the application is in a source tree instead of installed on the system, you might want to put "libtool" before "gdb")

2. Pass on the command line arguments to the application by typing

```
set args (the arguments to the application)
```

at the (gdb) prompt

3. Type "run" at the (gdb) prompt and wait for the application to segfault. The application will run a lot slower, however.
4. After the segfault, type "bt" to get a backtrace. This is a stack of function calls detailing the path from main () to where the code is currently at.
5. If the application you're trying to debug contains threads, it is also useful to do

```
info threads
```

and get backtraces of all of the threads involved, by switching to a different thread using "thread (number)" and then again requesting a backtrace using "bt".

6. If you can't or don't want to work out the problem yourself, a copy and paste of all this information should be included in your bug report.

Q: On my system there is no gst-register command.

A: GStreamer version 0.10 does not need this anymore. The registry will be rebuilt automatically. If you suspect the registry is broken, just delete the `registry.*.xml` files under `$HOME/.gstreamer-0.X/` and run

```
gst-inspect
```

to rebuild the registry.

7. Building GStreamer from git

Q: Is there a way to test or develop against GStreamer from git without interfering with my system GStreamer installed from packages?

A: Yes! You have two options: you can either run GStreamer in an uninstalled setup (see How do I develop against an uninstalled GStreamer copy ?), or you can use GNOME's jhbuild.

Q: How do I check out GStreamer from git ?

A: GStreamer is hosted on Freedesktop.org. GStreamer consists of various parts. In the beginning, you will be interested in the "gstreamer" module, containing the core, and "gst-plugins-base" and "gst-plugins-good", containing the basic set of plugins. Finally, you may also be interested in "gst-plugins-ugly", "gst-plugins-bad" and "gst-ffmpeg" for more comprehensive media format support.

To check out the latest git version of the core and the basic modules, use

```
for module in gstreamer gst-plugins-base gst-plugins-good; do
  git clone git://anongit.freedesktop.org/git/gstreamer/$module ;
done
```

This will create three directories in your current directory: "gstreamer", "gst-plugins-base", and "gst-plugins-good". If you want to get another module, use the above git clone command line and replace $module with the name of the module. Once you have checked out these modules, you will need to change into each directory and run ./autogen.sh, which will among other things checkout the common module underneath each module checkout.

The modules page (http://gstreamer.freedesktop.org/modules/) has a list of active ones together with a short description.

Q: How do I get developer access to GStreamer git ?

A: If you want to gain developer access to GStreamer git, you should ask for it on the development lists, or ask one of the maintainers directly. We will usually only consider requests by developers who have been active and competent GStreamer contributors for some time already. If you are not already a registered developer with a user account on Freedesktop.org, you will then have to provide them with:

1. your desired unix username
2. your full name
3. your e-mail address
4. a copy of your public sshv2 identity. If you do not have this yet, you can generate it by running "ssh-keygen -t rsa -f ~/.ssh/id_rsa.pub-fdo". The resulting public key will be in `~/.ssh/id_rsa.pub-fdo`
5. your GPG fingerprint. This would allow you to add and remove ssh keys to your account.

Once you have all these items, see http://freedesktop.org/wiki/AccountRequests for what to do with them.

Q: I ran autogen.sh, but it fails with aclocal errors. What's wrong ?

```
+ running aclocal -I m4 -I common/m4 ...
aclocal: configure.ac: 8: macro 'AM_DISABLE_STATIC' not found in library
aclocal: configure.ac: 17: macro 'AM_PROG_LIBTOOL' not found in library
aclocal failed
```

What's wrong ?

A: aclocal is unable to find two macros installed by libtool in a file called libtool.m4. Normally this would indicate that you don't have libtool, but that would mean autogen.sh would have failed on not finding libtool.

It is more likely that you installed automake (which provides aclocal) in a different prefix than libtool. You can check this by examining in what prefix both aclocal and libtool are installed.

You can do three things to fix this :

1. install automake in the same prefix as libtool
2. force use of the automake installed in the same prefix as libtool by using the --with-automake option
3. figure out what prefix libtool has been installed to and point aclocal to the right location by running
   `export ACLOCAL_FLAGS="-I $(prefix)/share/aclocal"`
   where you replace prefix with the prefix where libtool was installed.

Q: Why is "-Wall -Werror" being used ?

A: "-Wall" is being used because it finds a lot of possible problems with code. Not all of them are necessarily a problem, but it's better to have the compiler report some false positives and find a work-around than to spend time chasing a bug for days that the compiler was giving you hints about.

"-Werror" is turned off for actual releases. It's turned on by default for git and prereleases so that people actually notice and fix problems found by "-Wall". We want people to actively hit and report or fix them.

If for any reason you want to bypass these flags and you are certain it's the right thing to do, you can run

```
make ERROR_CFLAGS=""
```

to clear the CFLAGS for error checking.

8. Developing applications with GStreamer

Q: How do I compile programs that use GStreamer ?

A: GStreamer uses pkg-config to assist applications with compilation and linking flags. pkg-config is already used by GTK+, GNOME, SDL, and others; so if you are familiar with using it for any of those, you're set.

If you're not familiar with pkg-config to compile and link a small one-file program, pass the --cflags and --libs arguments to pkg-config. For example:

```
$ libtool --mode=link gcc `pkg-config --cflags --libs gstreamer-0.10` -o myprog myprog.c
```

would be sufficient for a gstreamer-only program. If (for example) your app also used GTK+ 2.0, you could use

```
$ libtool --mode=link gcc `pkg-config --cflags --libs gstreamer-0.10 gtk+-2.0` -o myprog
```

Those are back-ticks (on the same key with the tilde on US keyboards), not single quotes.

For bigger projects, you should integrate pkg-config use in your Makefile, or integrate with autoconf using the pkg.m4 macro (providing PKG_CONFIG_CHECK).

Q: How do I develop against an uninstalled GStreamer copy ?

A: It is possible to develop and compile against an uninstalled copy of gstreamer and gst-plugins-* (for example, against gits checkouts). The easiest way to do this is to use a bash script like this: latest version of gst-uninstalled (http://cgit.freedesktop.org/gstreamer/gstreamer/tree/scripts/gst-uninstalled). If you put this script in your path, and symlink it to gst-git (if you want to develop against git master) or to gst-released (if you want to develop against the lastest release of each module), it will automatically use the uninstalled version from that directory (ie. gst-git will look for a directory called 'git', and gst-released will expect the uninstalled modules to be in the 'released' directory; you are free to use any name or identifier you like here).

This requires you to have put your checkouts of gstreamer and gst-plugins under ~/gst/git (for the master version). The program is easily modifiable if this isn't the case.

After running this script, you'll be in an environment where the uninstalled tools and plugins will be used by default. Also, pkg-config will detect the uninstalled copies before (and prefer them to) any installed copies.

Q: How can I use GConf to get the system-wide defaults ?

A: For GNOME applications it's a good idea to use GConf to find the default ways of outputting audio and video. You can do this by using the 'gconfaudiosink' and 'gconfvideosink' elements for audio and video output. They will take care of everything GConf-related for you and automatically use the outputs that the user configured. If you are using gconfaudiosink, your application should set the 'profile' property.

Q: How do I debug these funny shell scripts that libtool makes ?

A: When you link a program against uninstalled GStreamer using libtool, funny shell scripts are made to modify your shared object search path and then run your program. For instance, to debug gst-launch, try

```
libtool --mode=execute gdb /path/to/gst-launch
```

. If this does not work, you're probably using a broken version of libtool.

Q: Why is mail traffic so low on gstreamer-devel ?

A: Our main arena for coordination and discussion is IRC, not email. Join us in #gstreamer on irc.freenode.net (irc://irc.freenode.net/#gstreamer) For larger picture questions or getting more input from more persons, a mail to gstreamer-devel is never a bad idea.

Q: What kind of versioning scheme does GStreamer use ?

A: For public releases, GStreamer uses a standard MAJOR.MINOR.MICRO version scheme. If the release consists of mostly bug fixes or incremental changes, the MICRO version is incremented. If the release contains big changes, the MINOR version is incremented. If we're particularly giddy, we might even increase the MAJOR number. Don't hold your breath for that though.

During the development cycle, GStreamer also uses a fourth or NANO number. If this number is 1, then it's a git development version. Any tarball or package that has a nano number of 1 is made from git and thus not supported. Additionally, if you didn't get this package or tarball from the GStreamer team, don't have high hopes on it doing whatever you want it to do.

If the number is 2 or higher, it's an official pre-release in preparation of an actual complete release. Your help in testing these tarballs and packages is very much appreciated.

Q: What is the coding style for GStreamer code?

A: The core and almost all plugin modules are basically coded in K&R with 2-space indenting. Just follow what's already there and you'll be fine.

Individual plugins in gst-plugins-* or plugins that you want considered for addition to one of the gst-plugins-* modules should be coded in the same style. It's easier if everything is consistent. Consistency is, of course, the goal.

Simply run your code (only the *.c files, not the header files) through

```
indent \
  --braces-on-if-line \
  --case-brace-indentation0 \
  --case-indentation2 \
  --braces-after-struct-decl-line \
  --line-length80 \
  --no-tabs \
  --cuddle-else \
  --dont-line-up-parentheses \
  --continuation-indentation4 \
  --honour-newlines \
  --tab-size8 \
  --indent-level2
```

before submitting a patch. (This is using GNU indent.) There is also a gst-indent script in the GStreamer core source tree in the tools directory which wraps this and contains the latest option. The easiest way to get the indenting right is probably to develop against a git checkout. The local git commit hook will ensure correct indentation. We only require code files to be indented, header files may be indented manually for better readability (however, please use spaces for indenting, not tabs, even in header files).

As for the code itself, the GNOME coding guidelines (http://developer.gnome.org/doc/guides/programming-guidelines/book1.html) is a good read. Where possible, we try to adhere to the spirit of GObject and use similar coding idioms.

Patches should be made against git master or the latest release and should be in 'unified context' format (use diff -u -p). They should be attached to a bug report (or feature request) in bugzilla (http://bugzilla.gnome.org) rather than sent to the mailing list. Also see SubmittingPatches (http://gstreamer.freedesktop.org/wiki/SubmittingPatches) in the GStreamer wiki.

Q: I have translated one of the module .po files into a new language. How do I get it included?

A: GStreamer translations are uniformly managed through the Translation Project (http://translationproject.org). There are some instructions on how to join the Translation Project team and submit new translations at http://translationproject.org/html/translators.html.

New translations submitted via the Translation Project are merged periodically into git by the maintainers by running 'make download-po' in the various modules.

9. GStreamer Legal Issues

This part of the FAQ is based on a series of questions we asked the FSF to understand how the GPL works and how patents affects the GPL. These questions were answered by the FSF lawyers (http://www.fsf.org/), so we view them as the final interpretation on how the GPL and LGPL interact with patents in our opinion. This consultancy was paid for by Fluendo (http://www.fluendo.com/) in order to obtain clear and quotable answers. These answers were certified by the FSF lawyer team and verified by FSF lawyer and law professor Eben Moglen.

Q: Can someone distribute the combination of

- GStreamer, the LGPL library
- MyPlayer, a GPL playback application
- The binary-only Sorenson decoder

together in one distribution/operating system ? If not, what needs to be changed to make this possible ?

A: This would be a problem, because the GStreamer and MyPlayer licenses would forbid it. In order to link GStreamer to MyPlayer, you need to use section 3 of the LGPL to convert GStreamer to GPL. The GPL version of GStreamer forbids linking to the Sorenson decoder. Anyway, the MyPlayer GPL license forbids this.

If the authors of MyPlayer want to permit this, we have an exception for them: the controlled interface exception from the FAQ. The idea of this is that you can't get around the GPL just by including a LGPL bit in the middle.

Note: MyPlayer is a completely fictituous application at the time of writing.

Q: Suppose Apple wants to write a binary-only proprietary plugin for GStreamer to decode Sorenson video, which will be shipped stand-alone, not part of a package like in the question above. Can Apple distribute this binary-only plugin ?

A: Yes, modulo certain reverse engineering requirements in section 6 of the LGPL.

Q: If a program released under the GPL uses a library that is LGPL, and this library can dlopen plug-ins at runtime, what are the requirements for the license of the plug-in ?

A: You may not distribute the plug-in with the GPL application. Distributing the plug-in alone, with the knowledge that it will be used primarily by GPL software is a bit of an edge case. We will not advise you that it would be safe to do so, but we also will not advise you that it would be absolutely forbidden.

Q: Can someone in a country that does not have software patents distribute code covered by US patents under the GPL to people in, for example, Norway ? If he/she visits the US, can he/she be arrested ?

A: Yes, he can. No, there are no criminal penalties for patent infringement in the US.

Q: Can someone from the US distribute software covered by US patents under the GPL to people in Norway ? To people in the US ?

A: This might infringe some patents, but the GPL would not forbid it absent some actual restriction, such as a court judgement or agreement. The US government is empowered to refuse importation of patent infringing devices, including software.

Q: There are a lot of GPL- or LGPL-licensed libraries that handle media codecs which have patents. Take mad, an mp3 decoding library, as an example. It is licensed under the GPL. In countries where patents are valid, does this invalidate the GPL license for this project ?

A: The mere existence of a patent which might read on the program does not change anything. However, if a court judgement or other agreement prevents you from distributing libmad under GPL terms, you can not distribute it at all.

The GPL and LGPL say (sections 7 and 11): "If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all."

Q: So let's say there is a court judgement. Does this mean that the GPL license is invalid for the project everywhere, or only in the countries where it conflicts with the applicable patents ?

A: The GPL operates on a per-action, not per-program basis. That is, if you are in a country which has software patents, and a court tells you that you cannot distribute (say) libmad in source code form, then you cannot distribute libmad at all. This doesn't affect anyone else.

Q: Patented decoding can be implemented in GStreamer either by having a binary-only plugin do the decoding, or by writing a plugin (with any applicable license) that links to a binary-only library. Does this affect the licensing issues involved in regards to GPL/LGPL?

A: No.

Q: Is it correct that you cannot distribute the GPL mad library to decode mp3's, *even* in the case where you have obtained a valid license for decoding mp3 ?

A: The only GPL-compatible patent licenses are those which are open to all parties posessing copies of GPL software which practices the teachings of the patent.

If you take a license which doesn't allow others to distribute original or modified versions of libmad practicing the same patent claims as the version you distribute, then you may not distribute at all.