PulseAudio

PulseAudio is a sound server running on top of some other sound system, usually ALSA. The original purpose was to get software mixing and transmit sound over network. Now PulseAudio is more than that, but it came at cost of increased complexity, which is not very good, considering the Linux sound system is already overly complex. But despite that the sound server is widely used by most Linux distributions. Slackware is not one of them because of its KISS principle and the fact that ALSA does its job fairly well in majority of cases. However, there are some programs (cough, Skype v4.3+, cough) that require PulseAudio in order to have working sound. There is a project called apulse that tries to emulate PulseAudio, but it doesn't work for everyone. This article will describe how to install and use PulseAudio in case you have/want to use it.

Installation

SlackBuilds.org is a great place to obtain packages missing on the stock system. Luckily for us, it has PulseAudio. It has very few dependencies (PA itself, at least), but there might be a few caveats.

Besides PulseAudio itself, you'll need alsa-plugins in order to get sound from non-pulse applications. Also, `pavucontrol` is highly recommended, because PA's config files are pain.

You can build the packages either directly using SBo, or using `sbotools`, or using `sbopkg`. There are more tools available, but I mainly use these two. The manual method is more tedious, but you have more control. sbotools can resolve dependencies and are able to build compat32 packages on x86_64 multilib systems. sbopkg gives you more control than sbotools, but requires more intervention. It doesn't matter which method you use, it's down to a personal preference.

You can also use pre-compiled packages in order to install all necessary applications, but it's so trivial I won't cover it here. You can use, for example, Slacky repo.

It's worth mentioning that if you need PulseAudio and/or multilib *for Skype only*, there's a package by zerouno with statically linked PA and necessary libraries that doesn't require you to install neither PA, nor multilib (for x86_64 systems).

Installation on a x86 or pure x86_64 system

This part used to be as easy as “install speex and json-c, then pulseaudio, then alsa-plugins”. Unfortunately, with the realease of speex-1.2rc2 there's a bug preventing alsa-plugins to build correctly even if you disable speex support. The solution for the time being is to add `-DHAVE_STDINT_H` to apulse-plugins' CFLAGS by editing the SlackBuild.

Other than that the installation part is pretty straightforward and SlackBuilds work perfectly.

For pavucontrol you need to resolve more dependencies, so sbotools or queue files for sbopkg are recommended, because building everything by hand is very tedious.

Installation on a x86_64 multilib system

Here I'll assume you know how to set up a multilib system and how to build 32bit packages on it. In case you don't, read this article carefully.

If you want 32bit applications (like Skype) on a multilib system to have sound through PulseAudio, you'll have to build compat32 package. And this is where things get complicated and manual intervention is needed.

First, install 64bit package of PulseAudio as described above.

Second, you need to compile 32bit package. Unfortunately, you can't use sbotools to do that for you because of a few errors that require you to edit SlackBuild for PA. You can compile speex and json-c as compat32 packages without any problem, but PulseAudio for some reason doesn't want to compile with 32bit version of libxcb and libcap, so, if it fails to compile as is, you'll need to edit SlackBuild to add `--disable-x11 --without-caps` for it to build successfully. Sometimes you need to disable more options, inspect the output in case compilation fails. It doesn't matter here, because all we are after is 32bit libpulse. Next, for some reason even after importing 32dev.sh PA still tries to use some 64bit libraries. So you'll need to start SlackBuild like this:

```
  PKG_CONFIG_PATH=/usr/lib/pkgconfig PKG_CONFIG_LIBDIR=/usr/lib/pkgconfig
LDFLAGS="-L/usr/lib" ./pulseaudio.SlackBuild
```

This will force PA to use the correct libraries and allow it to compile. Make compat32 package and install it. You don't need 32bit pavucontrol. 32bit alsa-plugins is needed only for 32bit applications, usually wine, to work properly.

Using PulseAudio

Actually, it's difficult to *not* use it, because it just keeps spawning. If you want to use PulseAudio exclusively as you sound server, then that's fine. If not, I'll cover what to do a bit later.

PulseAudio as the main sound system

All you need to do now it just create `/etc/asound.conf` (or `~/.asoundrc`) with this content:

```
  pcm.pulse {
      type pulse
  }
  ctl.pulse {
      type pulse
  }
  pcm.!default {
      type pulse
  }
  ctl.!default {
      type pulse
```

```
}
```

That'll make ALSA-only applications play sound through PA. The basic configuration of PulseAudio is done, because it tries to be smart and doesn't require much configuration if you want to use it.

PulseAudio as an additional sound system

It is possible to have PulseAudio not use ALSA exclusively for itself. For that you'll need to comment out all strings about udev in `default.pa` config file, uncomment the ones with ALSA and use `device=dmix`. Now pulse-only applications can use PulseAudio, while the rest of the system will continue to use ALSA.

If you are annoyed by PA constantly spawning, define `autospawn = no` in `client.pa` config file. Then you'll have to start PulseAudio manually each time you want sound from a pulse-only application.

The downside of this approach is that you are limited to only one device at a time, while udev allows PulseAudio to use all sound devices it finds. You'll have to configure your devices manually through ALSA's asoundrc config file.

In case of sound capturing, you have two options. First, use `device=dsnoop`. You can read about dsnoop here. Second, you specify your actual capture device for PA to use exclusively, like `device=hw:2,0`.

When using PA as an additional sound system, you'll need to choose it as an output sound device in applications you want to use PA, while other will use the default sound system.

Starting PulseAudio

XDG-compliant DEs will start PulseAudio automatically thanks to `/etc/xdg/autostart/pulseaudio{,-kde}.desktop` files.

In case you aren't using those, you can use `start-pulseaudio-x11` command. Just add it to your DE/WM autostart. Plain `pulseaudio --start` will also work. To stop PA use `pulseaudio --kill` command. Be warned, that if you didn't disable autospawning, it'll respawn as soon as some application requires the sound.

PulseAudio in Xfce

If you use PA as the default sound system, everything will work just fine, except the multimedia keys. For them to work correctly, install xfce4-volumed-pulse. At the moment of writing, the autostart .desktop file is located in the wrong place: `/etc/xfce/xdg/autostart/xfce4-volumed-pulse.desktop`. Move it to `/etc/xdg/autostart/` for it to start automatically. For the mixer to work, choose your default sound card and 'Master' channel.

PulseAudio in KDE

KDE uses GStreamer to play sound. ALSA applications will work fine, as they'll be using `asoundrc` config file. Phonon/GStreamer applications may have some problems.

While `start-pulseautio-kde.desktop` file re-routes the sound from Phonon to PA, the PA server is not set as the default device. You'll need to move it to the top everywhere in the Phonon settings.

Gstreamer applications will try to use ALSA, which re-routes the sound to PA, but in some rare cases it may not work. For them to use PA directly, you'll need to rebuild `gst-plugins-good` package. Since this package is provided by the system, you can get the sources and the SlackBuild from the Slackware tree.

PulseAudio mixers

The main mixer for PulseAudio is `pavucontrol`, although it doesn't provide neither tray icon/applet, nor keyboard shortcuts. For that you'll have to use other mixers.

In case you are running PA as your main sound system and set up `asoundrc` as I described above, any ALSA mixer will do just fine. Just make sure it's configured to use the default (or “pulse”, or your default output sound card) device and “Master” channel.

If you like CLI commands, there's ponymix. Actually, it does the same thing you can achieve with `pactl`, but you can never have too much pony in Linux 😬

Speaking of `pactl`. These are commands to define your own keybindings, e. g. in `~/.fluxbox/keys`:

```
  pactl set-sink-volume 0 +5% #raise the main sink's volume by 5%
  pactl -- set-sink-volume 0 -5% #lower the main sink's volume by 5%; notice
the "--"
  pactl set-sink-mute 0 toggle #mute/unmute the main sink
```

For more commands and options see `man pactl`.

By the way, by default PA also installs bash completion script, which may come in very handy when using `pactl`, so `bash-completion` package from `extra` set is recommended.

Sources

Some information was taken from ArchLinux Wiki.

Originally written by fsLeg

howtos, multimedia, sound, pulseaudio, needs attention