Audio in embedded Linux systems

Free Electrons

Rights to copy

© Copyright 2004-2009, Free Electrons
feedback@free-electrons.com

Document sources, updates and translations:
http://free-electrons.com/docs/audio

Corrections, suggestions, contributions and translations are welcome!

Latest update: Sep 15, 2009

Scope of this training

- Audio in embedded Linux systems
  This training targets the development of audio-capable embedded Linux systems. Though it can be useful to playing or creating sound on GNU/Linux desktops, it is not meant to cover everything about audio on GNU/Linux.
- Linux 2.6
  This training only targets new systems based on the Linux 2.6 kernel. This way, you leverage the most advanced technology and don't learn about something getting obsolete.

Contents (1)

Contents (2)

Free Software audio

Quick Glossary

- PCM: Pulse Code Modulation
  Digital audio encoding, representing the amplitude of a signal at uniform intervals.
- Codec: coder / decoder
  Program or device coding and / or decoding a data stream or a signal.
- MIDI: Musical Instrument Digital Interface.
  Standard to control electronic musical instruments.

See http://wikipedia.org for details!

Free Software Audio

Audio codecs and file formats

MP3

MPEG-1 Audio Layer III
from the Fraunhofer Society

- Lossy audio format
- Bitrates from 32 to 320 kbit/s
- Quality depends on the bitrate: 128-192: good, 192-224: very good, 224-320: excellent
- Depends also pretty much on the encoder and on the source.
- Depends on the listener too!

- The most popular. Users have lots of files in this format.
- Free Software encoders and decoders exist
- But relies on patented algorithms. Depending on which country you sell to, you may have to pay for a license.
- Licenses can apply to encoding, decoding or even songs!
- Ask for legal advice!

See http://en.wikipedia.org/wiki/Mp3 for details

AAC

Advanced Audio Coding
MPEG-4 Audio

- Standard format from the MPEG group: Dolby, Fraunhofer, AT&T, Sony, and Nokia
- Lossy audio format
- Designed to replace MP3. Consistently better audio quality than MP3 at lower bitrates.
- Can be DRM encrypted (FairPlay).

- Used on some on-line music stores (Apple iTunes) and portable players (Apple iPod).
- Also burdened by patents, like MP3. License needed to encode and read this format.
- Free Software decoders available.
- Just one Free Software encoder available (faac).

More details on http://en.wikipedia.org/wiki/Advanced_Audio_Coding

RealAudio

From RealNetworks
http://realnetworks.com/

- Lossy audio format
- Proprietary format
- Designed for
  very low bandwidth connections.
- Bitrates: 12 to 800 kbit/s
  Now uses AAC at 128 kbit/s and more.
- Lossless format also supported

- Free Software decoder available: mplayer
- Mainly used for streaming, used by a significant number of on-line media.
  Useful for mobile devices connecting to these media.
- Only proprietary encoders.
  RealNetworks encoder free of charge only for personal use.

More details on http://en.wikipedia.org/wiki/Real_Audio

WMA

Windows Media Audio

- Microsoft proprietary, as a alternative to MP3 (patented by somebody else) and now AAC.
- Almost always encapsulated in an Advanced Systems Format (ASF) file.
- File extensions: `asf` or `wma`
- Supports constant and variable bitrates, and lossless compression.
- Can be DRM encrypted.

- Now supported by more and more digital players and on-line music stores. Users may ask for WMA playing capability.
- Lack of Free Software players (except `libavcodec`) and encoders.
- Relies on patented algorithms.
- Licenses may apply to encoding, decoding or even songs, though MS is still very tolerant so far (to achieve dominance).

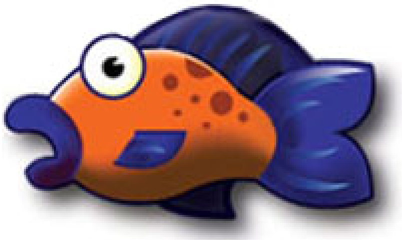
See http://en.wikipedia.org/wiki/WMA for details

Ogg Vorbis

From the Xiph foundation
http://xiph.org/

- Ogg: container for multimedia streams
- Vorbis: lossy audio format
- Open, patent and royalty free!
- Bitrates from 45 to 500 kbit/s
- Variable bitrate
- Achieves better quality than MP3 at low bitrates.

- Growing in popularity. More and more hardware players available.
- Xiph.org releases libraries under a BSD-style license and GPL for tools.
- Various Free Software decoders and encoders available. Supported by many proprietary players too.



See for http://en.wikipedia.org/wiki/Ogg_vorbis details

Ogg Speex

From the Xiph foundation
http://www.speex.org/

- Ogg: container
  Usual file extension: `.spx`
- Speex: lossy audio dedicated to speech encoding.
- Targets Voice over IP applications, voice mail archival, audio books...
- Open, patent and royalty free!
- Constant or variable bitrate, from 2 to 44 kbit/s.
- Listen to samples on http://speex.org/samples/.
- Free Software encoder, decoders and applications available.
- Even supported by proprietary tools (e.g. MS NetMeeting).

See http://en.wikipedia.org/wiki/Speex for details

Flac

http://flac.sourceforge.net/
Supported by Xiph.org

- Lossless audio compression format
  Compress audio files at no risk!
- Preferred format for trading live music on-line.
- Supports streaming.
- Ogg: also used as a container.
- Integer-only coder and decoder available.

- Libraries available under a BSD-like license, and tools under the GPL.
- Free Software players available.
- Even starts to be supported by hardware players.

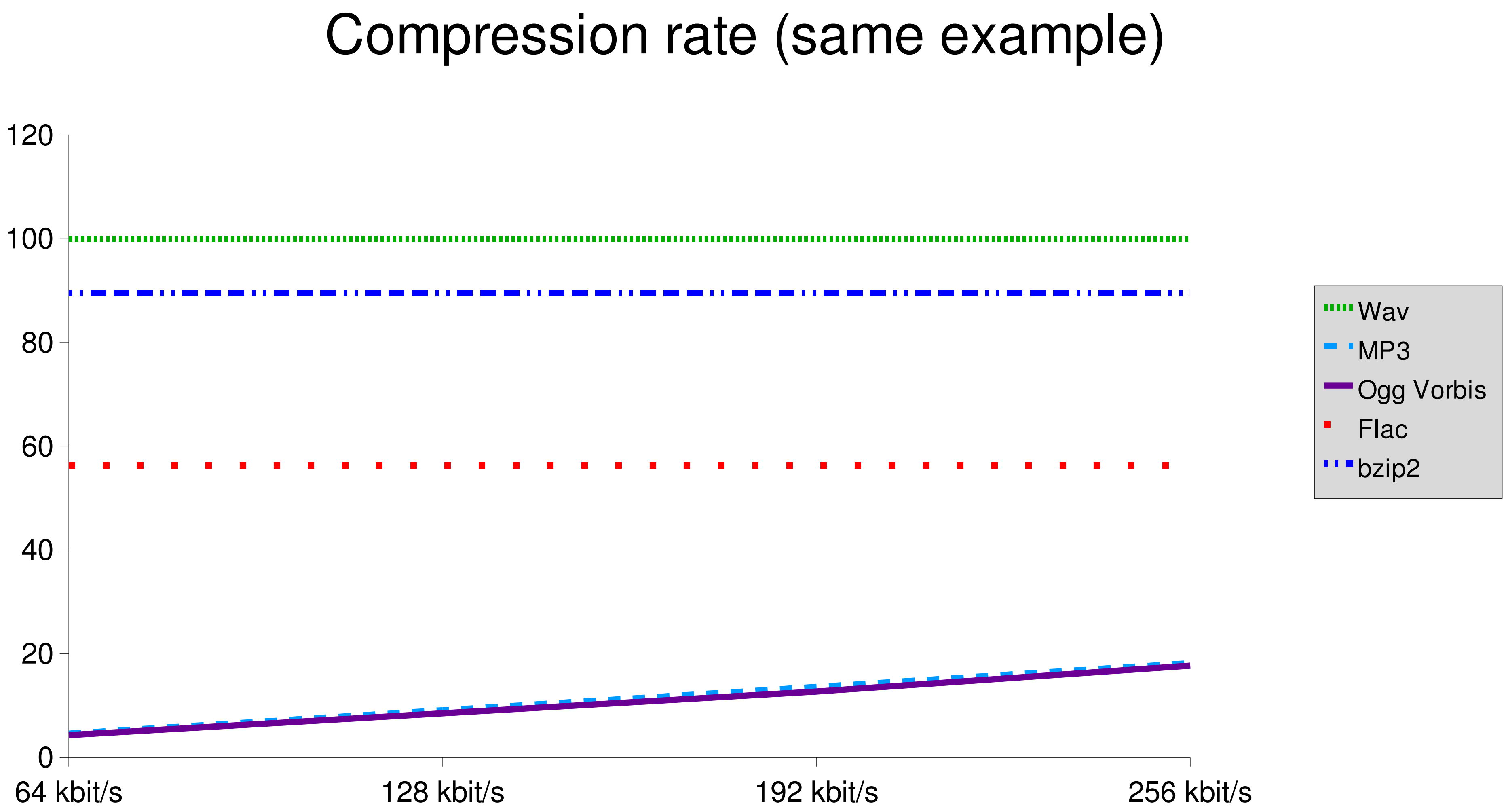
See http://en.wikipedia.org/wiki/FLAC for details

Compression rate example comparison (1)

Format / bitrate	64 kbit/s	128 kbit/s	192 kbit/s	256 kbit/s
Wav	100.00%			
MP3 (lame 3.96.1)	4.6% (22:1)	9.1% (11:1)	13.6% (7:1)	18.2% (5:1)
Ogg Vorbis (oggenc 1.0.1)	4.3% (23:1)	8.5% (12:1)	12.7% (8:1)	17.7 % (6:1)
Flac (flac 1.1.0)	56.30%			
bzip2 (1.0.2)	89.50%			

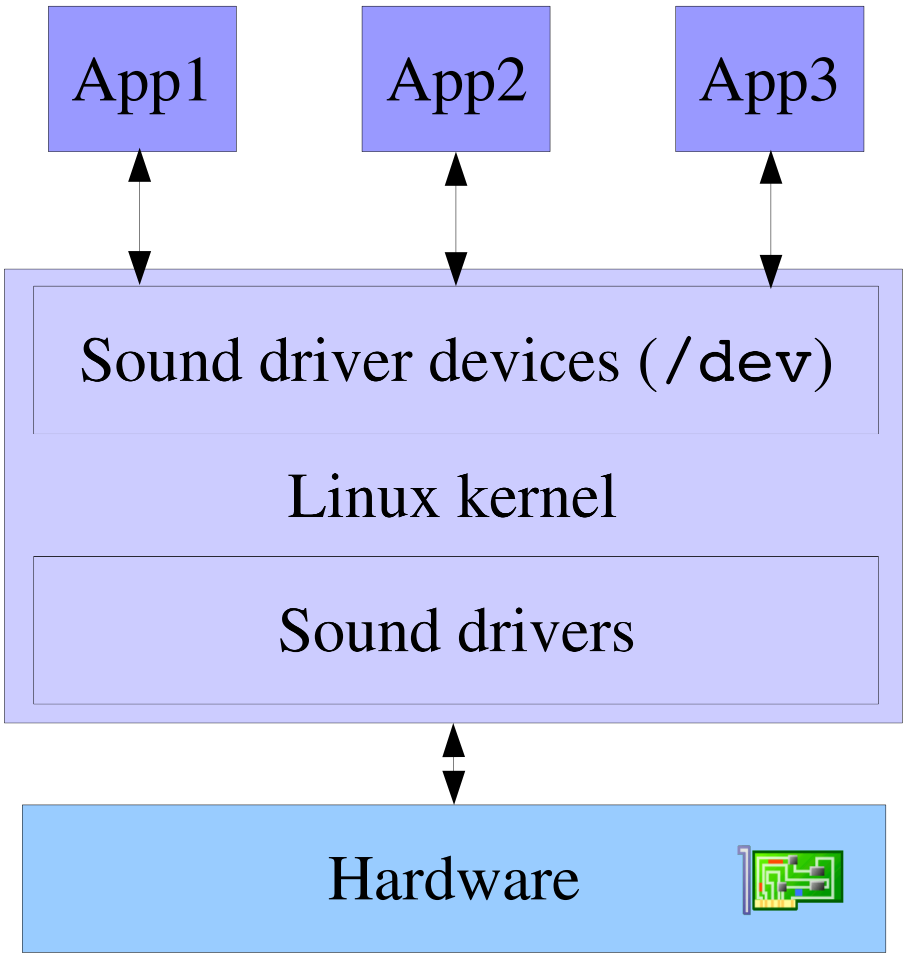
Source: Omara Portuondo, Flor de Amor (Cuban Salsa)

Compression rate example comparison (2)



System perspective

Traditional system architecture



User applications (concurrent access to resources)

OSS

The Open Sound System
http://www.4front-tech.com/oss.html

- Old sound card support system in Linux versions up to 2.4.
  Still used for some cards in 2.6 (porting to ALSA in progress).
- Originates from the Linux driver for the Sound Blaster 16 sound card.
  Extended to support other (often compatible) sound cards.
- Was also made available as a proprietary and enhanced version of OSS, also targeting other Unix systems (such as Solaris).
- June 14, 2007: open-sourced, but too late?
  Even if some drivers are reported to be better than ALSA ones, unlikely to be merged in mainstream Linux sources.

OSS sound devices

Main ones

- `/dev/dsp`
  D/A and A/D converter device, access, to generate audio or to read audio input.
- `/dev/mixer`
  Mixer control (mainly for controlling volume)
- `/dev/audio`
  Sun compatible digital audio (`.au` file format)
- `/dev/sequencer`
  Audio sequencer (MIDI)
- `/dev/sequencer2`
  Alternate sequencer device

To create the device files:

```
sudo mknod /dev/dsp c 14 3
sudo mknod /dev/mixer c 14 0
```

The major and minor numbers for these devices are defined in `Documentation/devices.txt` in the kernel sources.

OSS dsp interface

API for accessing playback and capture controls

- Writing to `/dev/dsp`: playback
  Reading from `/dev/dsp`: capture (recording)
- Only one application can open `/dev/dsp` at at time.
  Full duplex (playing and recording) not possible.
- Available ioctl settings: sample size and sample rate, number of read or write channels (1: mono, 2: stereo).

More details on
http://www.oreilly.de/catalog/multilinux/excerpt/ch14-05.htm

OSS mixer interface

C API for accessing mixer controls: mainly setting channel volume (left, right or mono), and selecting recording sources.

- Mainly based on ioctl commands,
  to either query PCM device capabilities and parameters
  or to assign values to these parameters.
- Applications don't have to open `/dev/mixer` to issue these ioctls. They can also use `/dev/dsp` if it is already open.
- Settings are kept even after the applications exit.

More details on
http://www.oreilly.de/catalog/multilinux/excerpt/ch14-07.htm

OSS issues and limitations

At the time ALSA was created.

- No support for software mixing
- No support for full-duplex.
- No hardware midi support.
- Lack of support for advanced features of many popular soundcards (like the Gravis Ultrasound one).
- The OSS developers decided to go closed-source. Community developers chose to create a whole new system.

Useful links about OSS

- O'Reilly's Multimedia Guide (currently out of print)
  Free excerpt: Programming Sound Devices
  http://www.oreilly.de/catalog/multilinux/excerpt/ch14-01.htm
  Full of details about the OSS API.

System perspective

ALSA kernel drivers

ALSA

Advanced Linux Sound Architecture
http://www.alsa-project.org/

- Project to provide full audio and MIDI functionality to Linux. Official Linux sound system since Linux 2.6.
- Started in 1998 by Jaroslav Kysela, originally to fully support all the features of the Gravis Ultrasound card.
- OSS emulation: fully supports applications originally created for OSS (still accessing `/dev/sound`, `/dev/dsp` or `/dev/mixer`).
- Device files in `/dev/snd/`.
  You don't need to use them directly. Use alsa-lib instead.

ALSA kernel space features

- Efficiently and fully supports from consumer sound cards to professional multichannel audio interfaces, bringing features not supported by OSS, such as hardware based MIDI synthesis, software mixing of multiple channels and full-duplex operation.
- Supports SMP (multiprocessor) systems.
  Thread-safe device drivers and user-space library.
- Consistent and generic control API
  for managing hardware controls.
- Fully modularized sound drivers.
  Shares code for similar chipsets.

ALSA /proc interface

- `/proc/asound/version`
  ALSA version
- `/proc/asound/cards`
  List of available sound cards

```
0 [I82801DBICH4   ]: ICH4 - Intel 82801DB-ICH4
                     Intel 82801DB-ICH4 with STAC9750/51 at 0xf4fff800, irq 5
1 [Modem          ]: ICH-MODEM - Intel 82801DB-ICH4 Modem
                     Intel 82801DB-ICH4 Modem at 0xb400, irq 5
```

- `/proc/asound/devices`
  List of card devices
- `/proc/asound/card<i>/id`
  Card identifier
- `/proc/asound/card<i>/pcm[c|p]<j>/info`
  Information about a capture (`c`) or playback (`p`) PCM device.
- More on http://alsa.opensrc.org/index.php/Proc_asound_documentation

ALSA and Linux 2.6 sources

- Official Linux 2.6 sources now use ALSA
- However, Linux releases do not always include the latest ALSA releases.
- Example: Linux 2.6.25 (Apr. 16, 2008) includes ALSA 1.0.16rc2 (Jan. 29, 2008), and not ALSA 1.0.16 (Feb. 6, 2008).
- How to check the ALSA version in your kernel sources? See `include/sound/version.h`.
- How to check the ALSA version in your running system? `cat /proc/asound/version`.
- If needed, you may install a more recent ALSA version.

Creating ALSA device files (1)

- Not needed if you have an elaborate system with udev.
  You can use the below udev rules to create these device files automatically
  (put these rules in a file in `/etc/udev/rules.d`):

```
# Sound devices, group under /dev/snd
KERNEL=="controlC[0-9]*",                 NAME="snd/%k"
KERNEL=="hwC[D0-9]*",                     NAME="snd/%k"
KERNEL=="midiC[D0-9]*",                   NAME="snd/%k"
KERNEL=="pcmC[D0-9cp]*",                  NAME="snd/%k"
KERNEL=="seq",                            NAME="snd/%k"
KERNEL=="timer",                          NAME="snd/%k"
```

- In an embedded system, you can create these device files manually.

Creating ALSA device files (2)

ALSA device files are easy to create by hand!

```
cat /proc/asound/devices
  0: [ 0]    : control
  8: [ 0- 0]: raw midi
 16: [ 0- 0]: digital audio playback
 17: [ 0- 1]: digital audio playback
 24: [ 0- 0]: digital audio capture
 33:         : timer
```

Minor number

card number

device number

```
mkdir /dev/snd
cd /dev/snd
mknod controlC0 c 116 0
mknod midiC0D0 c 116 8
mknod pcmC0D0p c 116 16
mknod pcmC0D1p c 116 17
mknod pcmC0D0c c 116 24
mknod timer c 116 33
```

C: Card

0: Card number

D: Device

0/1: Device number

p/c: playback / capture

Dummy ALSA driver

Dummy ALSA device discarding any sound played on it.

- Can be useful to test your audio applications even if the audio hardware is not ready yet, or to check that whether problems come from your experimental driver or from your experimental application.
- To use it:

```
> modprobe snd-dummy
> cat /proc/asound/cards
 0 [I82801DBICH4   ]: ICH4 - Intel 82801DB-ICH4
                      Intel 82801DB-ICH4 with
STAC9750,51 at 0xf4fff800, irq 5
 1 [Dummy          ]: Dummy - Dummy
                      Dummy 1
```

Writing ALSA drivers

Useful references

- "Writing an ALSA Driver", Takashi Iwai
  http://www.alsa-project.org/~iwai/writing-an-alsa-driver/
  A very comprehensive guide!
  We made small contributions to it.
- ALSA driver API reference
  http://www.alsa-project.org/~iwai/alsa-driver-api/

System perspective

Kernel requirements for sound

Real-time requirements for audio

Very low latency requirements in some audio applications

- < 3 ms, when the output is combined with the original signal. Otherwise, “comb filtering”.
- Audio applications need high priority, so that the output devices are always fed. Otherwise: choppy audio.
- Musicians: need to hear immediately what they are playing.
- When audio needs to be in sync with video.
- Audio communications.

Reducing Linux latency

- Standard Linux: latency can reach a 100 ms magnitude.
- Typical target latency: 1 to 5 ms.
- Hard real-time Linux (such as RTAI) would complicate application development too much (special API to start real-time jobs).
- Since Linux 2.2 and 2.4, low latency patches have been used by audio users.
- Better responsiveness in standard Linux 2.6, but not enough yet.
- Fortunately, realtime-preempt patches now satisfy Linux audio user needs. The system can reach latencies under 100 µs.

Real-time preemption patches

http://www.kernel.org/pub/linux/kernel/projects/rt/

- Patches from Ingo Molnar, Thomas Gleixner, and the kernel development community to eliminate sources of latency.
- They patiently have their changes accepted in the mainstream Linux kernel, and find solutions which do not hurt the general purpose nature of Linux.
- Available for most hardware architectures supported by Linux, in particular on embedded ones. Getting stable on most common platforms.

See our http://free-electrons.com/articles/realtime/ presentation for technical and usage details.

System perspective

ALSA userspace interface

ALSA user space features

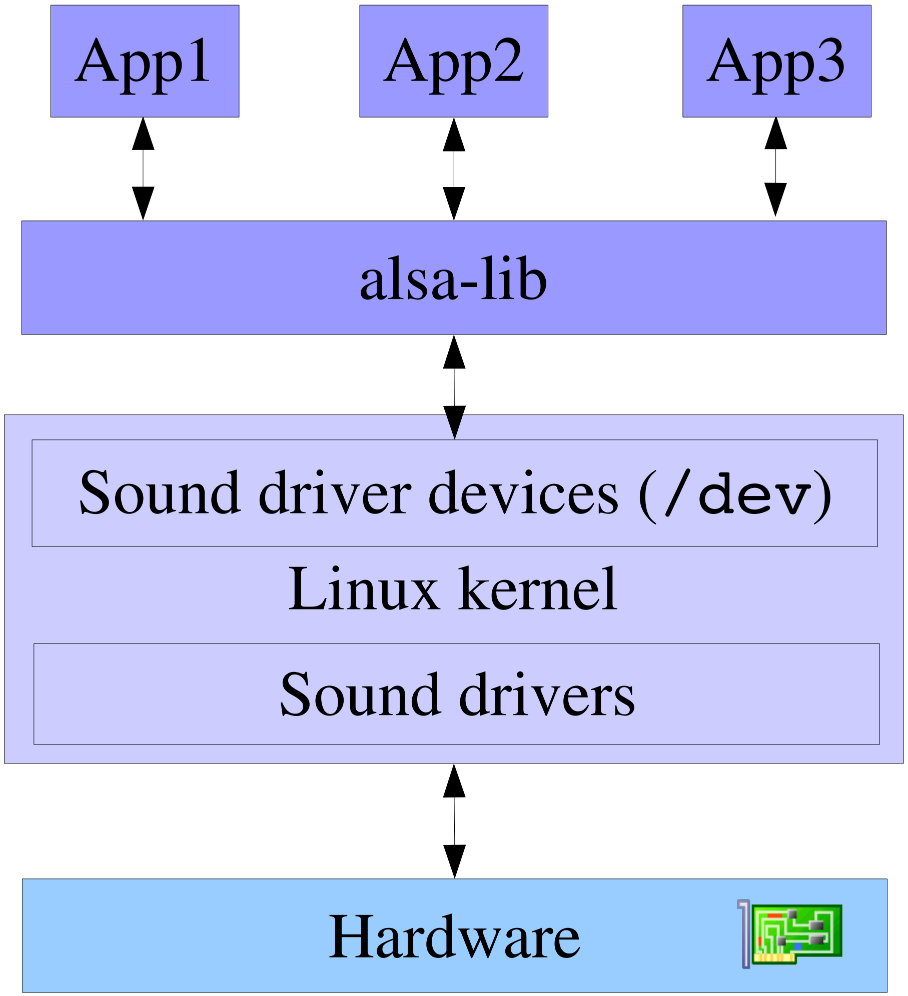
Based on the alsa-lib user-space library
to delegate sound control to user space.

- Lots of functionalities provided to user programs,
  such as software mixing (dmix).
- Support for the older OSS API,
  providing binary compatibility for most OSS programs.
- Supports user-specific configuration (`$HOME/.asoundrc`)
- Multi-thread safe
  Essential capability for the design of modern audio applications.

alsa-lib

- Named `libasound` in `/usr/lib`
- Size: 784 KB on Ubuntu 8.04 (i386)
- The size can be reduced by removing features at configuration time.

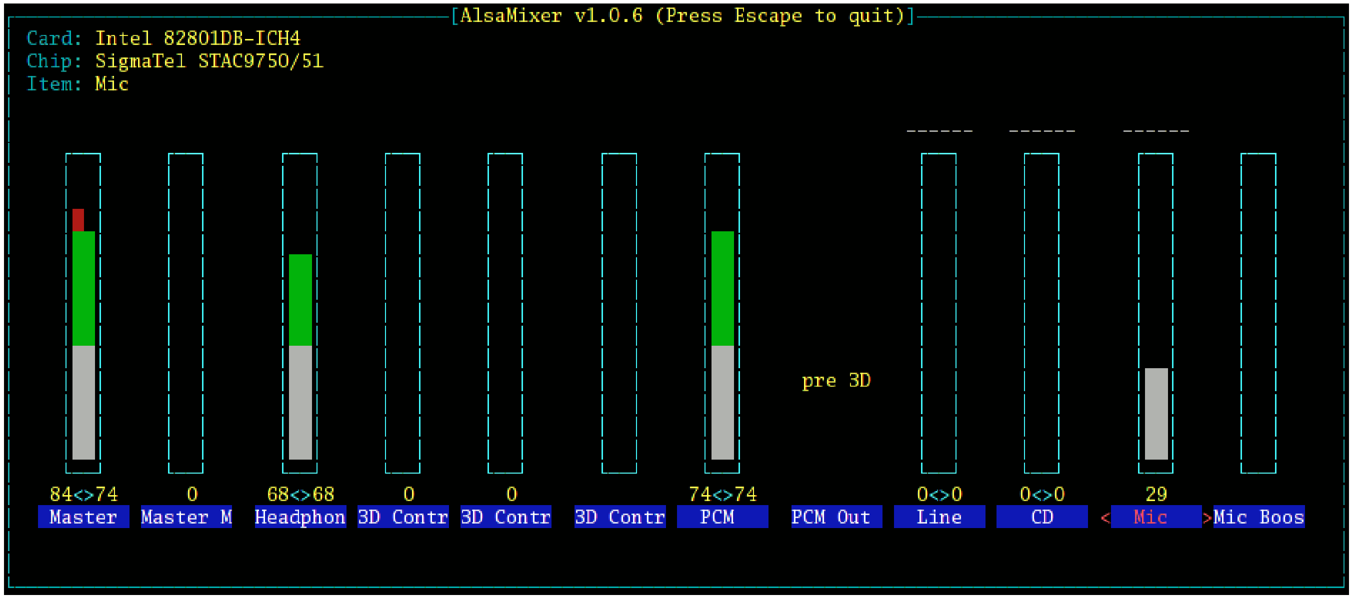
ALSA system architecture



User applications (concurrent access to resources)

No longer needed to access `/dev/` files!

alsamixer



- User interface to set channel volume control, and microphone input level.
- Text only (ncurses). Easy to use in embedded systems!

amixer

Same functionality as alsamixer, but from the command line (or from scripts). Examples:

Examples:

- `amixer -c 1 sset Line,0 80%,40% unmute cap`
  Sets the second soundcard's left line input volume to 80% and right line input to 40%, unmute it, and select it as a source for capture (recording).
- `amixer -c 2 cset iface=MIXER,name='Line Playback Volume",index=1 40%`
  Sets the third soundcard's second line playback volume(s) to 40%
- `amixer -c 2 cset numid=34 40%`
  Sets the 34th soundcard element to 40%

alsactl

Command available to the root user

- alsactl store [card_num]
  Stores the current alsamixer settings
  in /etc/asound.state.
  Otherwise, not saved after reboot.
- alsactl restore [card_num]
  Restores the saved alsamixer state.
- alsactl power [card_num] [state]
  Displays / sets the power state of soundcards.

alsa-lib configuration

Elaborate PCM stream handling can be defined by each user!

- `/etc/asound.conf`
  System wide definitions
- `$HOME/.asoundrc`
  User specific definitions

ALSA device naming

alsa-lib uses logical device names rather than device files

- Either raw hardware devices: `hw:i,j` or `plughw:i,j`
  `i`: card number, `j`: device number on the card
  `plughw`: automatically converts the sample format, rate, access type and number of channels to the hardware's native format.
  `hw`: requires a compatible configuration.
- Or aliases
  (defined in `/etc/asound.conf` or in `$HOME/.asoundrc`)
  `default`: `hw:0,0`
- Or plugins (see next page)

ALSA device naming example

Give a name to your soundcards

- Example: (in `/etc/asound.conf` or in `$HOME/.asoundrc`):

```
pcm.acmesound {
        type hw
        card 0
        device 0
}
```

- You can now build more PCM devices with it, and of course use it to play sound:

```
aplay -D acmesound rageagainstthewindows.wav
```

alsa-lib PCM plugins

User interface to alsa-lib for use in the command line or in `/etc/asound.conf` or `$HOME/.asoundrc`

- Extends the functionality and features of PCM devices. Correspond to alsa-lib library functions.
- Accepts parameters, which can also be passed through the command line.
- In a system running ALSA, all the plugins are defined in `/usr/share/alsa/alsa.conf` (the master configuration file for ALSA). Useful to see what their parameters are.

See http://www.alsa-project.org/alsa-doc/alsa-lib/pcm_plugins.html

A few plugin examples

- `hw`: it is itself a plugin, giving access to the specified hardware.
- `copy`: copies the contents of a PCM stream to another
- `null`: discards the contents of a PCM stream
- `file`: stores the contents of a PCM stream to a file.
  Can also be used to use a file as an input data stream.
- `tee`: plays the audio on a PCM stream
  and dumps it to a file too.
- `dmix`: mixes several streams.
  Enabled as default for sound cards without hardware mixing.
- `multi`: combines several hardware PCM devices
  into a virtual one.
- More: rate and format conversion, soft volume, etc.

The plug plugin

- `plug` automatically performs channel duplication, sample value conversion and resampling when necessary.
- For example, `dmix` needs to resample all the audio to the same sample rate (48000 Hz by default) before doing the mixing work. That's why most of the time `dmix` is used together with `plug` (examples are coming soon).
- `hwplug`: used to directly do the `plug` work and play on the specified hardware (we already explained it).

Defining PCM devices from others

You can use `plug` to create more PCM devices from others:

- Rate conversion device

```
pcm.f44100 {
    type plug
    slave {
        pcm default
        rate 44100
    }
}
```

a PCM device name,

- Example use:

```
aplay -D f44100 foo.wav
```

- Virtual device recording to a given file:

```
pcm.recorder {
    type plug
    slave {
        pcm "file:sound.raw"
    }
}
```

a PCM device name, here the file plugin!

plugin parameters

- Example use:

```
aplay -F recorder bar.wav
```

Plugin declaration example

```
pcm.tee {
        @args [ SLAVE FILE FORMAT ]
        @args.SLAVE {
                type string
        }
        @args.FILE {
                type string
        }
        @args.FORMAT {
                type string
                default raw
        }
        type file
        slave.pcm $SLAVE
        file $FILE
        format $FORMAT
}

pcm.file {
        @args [ FILE FORMAT ]
        @args.FILE {
                type string
        }
        @args.FORMAT {
                type string
                default raw
        }
        type file
        slave.pcm null
        file $FILE
        format $FORMAT
}
```

From `/usr/share/alsa/alsa.conf`

You can see what the plugin parameters are.

Playing sound examples

- `aplay -D hw:0,0 yoohoo.wav`
  Plays to the first device on the first sound card.
- `aplay -D plughw:1,0 yoohoo.wav`
  Plays to the first device on the second sound card,
  with automatic conversion to a sample rate supported by this card.
- `aplay -D mycard yoohoo.wav`
  Plays to the `mycard` PCM device
  (defined in `/etc/asound.conf` or in `$HOME/.asoundrc`)
- `aplay -D null yoohoo.wav`
  Plays to the `Null` plugin
- `aplay -D file:/tmp/sounddump.raw`
  Plays to the file `plugin`, passing `/tmp/sounddump.raw` as a parameter to this plugin.

Software mixing example

ALSA makes it easy to mix several audio sources:

Run the below 2 commands:

```
alsaplayer -d plug:dmix simon.ogg &
alsaplayer -d plug:dmix garfunkel.ogg &
```

With aplay (supporting mainly WAV files):

```
aplay -D plug:dmix simon.wav &
aplay -D plug:dmix garfunkel.wav &
```

You can run any number of parallel processes.

Similarly, you can access other plug-ins and set their parameters from the command line.

If you don't have hardware mixing support, remember that mixing is enabled by default. You don't even have to specify the plugin!

Other utilities

speaker-test (alsa-utils package)

- Allows to test different ALSA configurations
- Example:
  `speaker-test -f 440 -t sine`
  Generates and plays a 440 Hz sine signal.
  `speaker-test  -c 2  -t wav`
  Says “front left” in the front left speaker, and
  “front right” in the right left speaker (`c`: number of channels)

Recording sound

Easy to do from the command line

- First make sure your microphone is enabled.
  In `alsamixer`, select the capture devices, and enable your microphone (maybe another device depending on your card), and if needed adjust the recording level.
- Then record from your microphone with the `arecord` command:
  `arecord -f cd input.wav` (CD quality wav file).
- Type `man arecord` for more options.

alsa-lib API

- Low-level API for sound programming. Most applications should probably be written with higher-level APIs
  - See http://0pointer.de/blog/projects/guide-to-sound-apis
- In ALSA
  - Control interface: general-purpose facility for managing registers of sound cards and querying available devices.
  - PCM interface: managing digital audio capture and playback.
  - Mixer interface: controls the devices on sound cards that route signals and control volume levels. Built on top of the control interface.
  - Timer interface: access to timing hardware on sound cards, used for synchronizing sound events.
  - Raw MIDI interface: access to a MIDI bus on a sound card. Works directly with the MIDI events. Protocol and timing management up to the programmer.
  - Sequencer interface: a higher-level interface for MIDI programming and sound synthesis than the raw MIDI interface. Handles much of the MIDI protocol and timing.

Opening and closing a device

- Opening a device

```
int snd_pcm_open( snd_pcm_t **handle,
                  const char *name,
                  snd_pcm_stream_t stream,
                  int mode)
```

  - `handle` is a returned value, containing an handle to reference the opened sound device
  - `name` is the name of the sound device to open. Can be "`default`" or "`hw:0,0`", "`plughw:1,0`", etc.
  - `stream` is either playback or capture
  - `mode` can be 0 (default value), or can be use to request non-blocking or asynchronous modes
- Closing a device

```
int  snd_pcm_close (snd_pcm_t *pcm)
```

Opening and closing : example

```
#include <alsa/asoundlib.h>

int ret;
snd_pcm_t *handle;

ret = snd_pcm_open( &handle, "default",
                    SND_PCM_STREAM_PLAYBACK,
                    0);

/* Use handle */

snd_pcm_close(handle);
```

PCM device parameters

- A PCM handle is configured through parameters, using a `snd_pcm_hw_params_t` structure
- Structure allocation
  `snd_pcm_hw_params_alloca()`
- Initialization to default values
  `int  snd_pcm_hw_params_any (snd_pcm_t *pcm, snd_pcm_hw_params_t *params)`
- Modification of the parameters
  `snd_pcm_hw_params_set_access()`, `snd_pcm_hw_params_set_format()`, `snd_pcm_hw_params_set_channels()`, etc.
- Associating the parameters to the device
  `int snd_pcm_hw_params (snd_pcm_t *pcm, snd_pcm_hw_params_t *params)`

PCM device parameters example

```
snd_pcm_hw_params_t *params;

snd_pcm_hw_params_alloca(&params);
snd_pcm_hw_params_any(handle, params);

snd_pcm_hw_params_set_access
     (handle, params, SND_PCM_ACCESS_RW_INTERLEAVED);

snd_pcm_hw_params_set_format
     (handle, params, SND_PCM_FORMAT_S16_LE);

snd_pcm_hw_params_set_channels
     (handle, params, 2);

val = 44100;
snd_pcm_hw_params_set_rate_near
     (handle, params, &val, &dir);

rc = snd_pcm_hw_params(handle, params);
```

PCM period

- The period is the number of frames played or recorded between two sound interrupts
- Its size in frames can be read using `snd_pcm_hw_params_get_period_size()`
  - The size in frame can be converted to a size in bytes with the proper multiplier (4 for stereo 16 bits)
- Its duration can be read using `snd_pcm_hw_params_get_period_time()`
- It is also possible to configure the period length, between minimum and maximum values exported by ALSA
- It allows to make a balance between latency and CPU usage
- It is then best to work with buffers of a size corresponding to the period

Playing sound

- Playing is done using
```
snd_pcm_sframes_t snd_pcm_writei
   (snd_pcm_t *pcm, const void *buffer,
    snd_pcm_uframes_t size)
```
  - pcm is the PCM handle
  - buffer the buffer containing the data to be played in the proper format
  - size the number of frames in the buffer to play
  - Returns the number of played frames, or an error. If the error is `-EPIPE`, it means that an underrun occured: the program didn't feed data fast enough for the soundcard
- The `i` in `snd_pcm_writei()` stands for interleaved. A non-interleaved variant exists, `snd_pcm_writen()`.

Recording sound and other APIs

- Similarly, recording is done using
  ```
  snd_pcm_sframes_t snd_pcm_readi
     (snd_pcm_t *pcm, void *buffer,
      snd_pcm_uframes_t size)
  ```
  - When `-EPIPE` is returned, an overrun occurred (the application didn't record the data fast enough)
- And the corresponding non-interleaved variant `snd_pcm_readn()`
- These two functions block until the frame has been played or recorded. Other, more complicated, I/O modes are available with ALSA
  - Asynchronous interface, notification by signal. Be careful, signals are difficult!
  - Memory-mapped API

PCM states and xrun recovery

- Each PCM handle is associated with a state
  - `open, setup, prepare, running, xrun, draining, paused, suspended, disconnected`
- After the configuration, the device is in the `prepare` state, and any read or write will move it to the running state
- And underrun or overrun while reading or writing will move it to the `xrun` state
  - A call to `snd_pcm_prepare()` is then necessary to put it back in the proper state
- Before closing the PCM handle, it's best to drain the remaining buffers using `snd_pcm_drain()`.

SALSA library

http://www.alsa-project.org/main/index.php/SALSA-Library

- Small, light-weight, hot and spicy version of the ALSA library, mainly for embedded systems with limited resources.
- Designed to be source-level compatible with ALSA library API for limited contents.
- Not supported: ALSA sequencer, pcm plugins and configuration. No format conversion!
- Size: reported to be 1/10th of `libasound`.

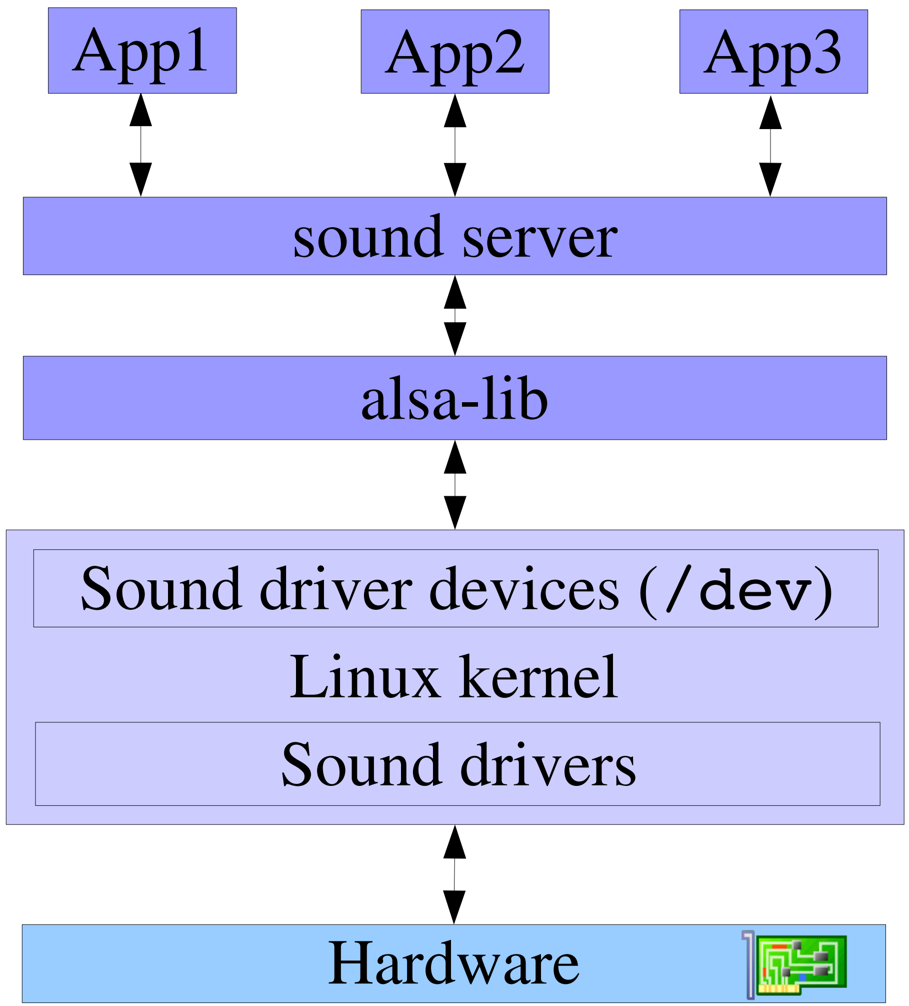
ALSA documentation

- Kernel sources:
  `Documentation/sound/alsa`
- Official ALSA documentation
  http://www.alsa-project.org/main/index.php/Documentation
- A close look at ALSA
  (useful explanations about ALSA configuration and plugins)
  http://www.volkerschatz.com/noise/alsa.html
- ALSA Wiki: lots of resources!
  http://alsa.opensrc.org/

System perspective

Sound servers

Sound server based system architecture



Sound servers take care of handling sound resource access and sound flows between apps

Traditional sound servers

Handle multiple audio streams, but primarily designed for incidental sound support such as desktop event sounds and lightweight game sound.

- aRts (`artsd`) - an Analog Real-Time Synthesizer
  Used by KDE until version 3. Replaced by Phonon in KDE 4.
  http://www.arts-project.org/
- esound (`esd`) - the Enlightened Sound Daemon
  Used by Gnome
  http://www.tux.org/~ricdude/EsounD.html

Both projects have achieved their goals. No active development.

Jack Audio Connection Kit

http://jackaudio.org/

- New sound server designed
from the ground up for professional audio work.
- Supports POSIX compliant operating systems,
such as GNU/Linux and MacOS X.
- Main focus
  - Low latency operation,
taking advantage of Linux low latency capabilities.
  - Synchronous execution of all clients.
- Not designed for embedded systems and not really useful
for them, except for professional audio devices

PulseAudio

http://pulseaudio.org by Lennart Poettering

- An increasingly popular sound server for POSIX and Win32 systems.
  Intended to be a drop-in replacement to esound.
  Better networking support (streaming).
- Library: LGPL, for connection to the server
  Server daemon: GPL. Also based on plug-ins (modules).
- Can be used by esound, ALSA, OSS, gstreamer applications...
- Low latency operation and latency measurement.
- Now features integer-only resampling (good for FPU-less embedded systems)
- The default sound server in Fedora (since version 8)
  and Ubuntu (since 8.04).

The best choice for embedded systems if you need a sound server!

Free Software Audio

Audio players

for the embedded Linux target

Console based sound players

- `alsaplayer -i text`
  AlsaPlayer's text interface. Universal and powerful. Can be built without GTK.
- `mplayer`
  Another universal solution. Most formats supported through plug-ins. Even supports on-line streams!
- `ogg123`
  Ogg/Vorbis player from Xiph.org.

Other console based sound players

- `aplay`
  From the ALSA project.
  Supported formats: wav, au (Sun), voc (Sound Blaster)

Integer-only audio decoders

Targeted to architectures with no hardware floating point unit (such as ARM ones)

- Tremor library (BSD license, from Xiph.org)
  http://xiph.org/vorbis/
  Can play any Ogg Vorbis file or stream.
  The project even includes a low memory branch.
- MAD: MPEG Audio Decoder (GNU GPL)
  http://www.underbit.com/products/mad/
  Can decode MPEG Audio layer I, II and III.
  Library (`libmad`) and command-line front-end (`madplay`).

Free Software Audio

Encoders

LAME

LAME Ain't an Mp3 Encoder: http://lame.sourceforge.net/

- License: LGPL
- MPEG1,2 and 2.5 layer III encoding.
  Constant and variable bitrates supported
- Quality comparable to Fraunhofer encoding engines and substantially better than most other encoders.
- GPL GPSYCHO psycho acoustic and noise shaping model:
  http://lame.sourceforge.net/gpsycho/gpsycho.html
- Available as a shared library, embedded in many applications
- Use subject to patents in some countries!

Misc mp3 encoders

- GOGO: http://freshmeat.net/projects/gogo/
  Patch against LAME doubling its speed, using MMX, 3D Now!, and SSE if supported by your processor.

Ogg Vorbis encoder

OggEnc

- Released with Ogg Vorbis software from Xiph.org.
- Simple example:
  `oggenc input.wav -b 128 -M 160 -o output.ogg`
- No integer-only encoder available yet.
  Relies heavily on floating-point computation.

Speex encoder

`speexenc`

- Released with the speex package from http://speex.org/
- Simple example:
  `speexenc --quality 7 input.wav output.spx`
- See `man speexenc` for full command line options
- No integer-only encoder available yet

Flac encoder

flac

- Available from http://flac.sourceforge.net
- Same command for encoding and decoding
- See `man flac` for command line options
- You can choose the compression rate.
  Just slower to encode, of course no quality loss at all!
- The encoder and decoder can now be compiled
  in integer-only mode!

Free Software Audio

Creating your own applications

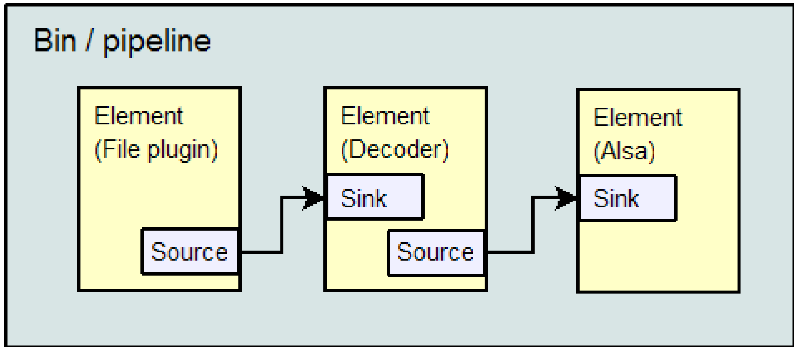
gstreamer (1)

http://gstreamer.freedesktop.org

- A cross-platform framework
  for building multimedia applications.
- Supported platforms:
  Linux (x86, arm, ppc), Solaris, (x86 and sparc), MacOS X, Windows.
- Small core library of less than 150KB.
  Already used in embedded systems (such as the Nokia 770).
- License: LGPL
- Many audio and multimedia applications are now based on it:
  http://gstreamer.freedesktop.org/apps/
  Highlights: Rythmbox, Totem (Gnome), Kaffeine (KDE), amaroK.

gstreamer (2)

- gstreamer uses the abstraction of *pipeline*.
- Elements in a pipeline are implemented by plug-ins. No need to recompile applications when a new plug-in is added.



- Lots of plug-ins are available:
  video and audio decoders (for most existing formats), encoders, and filters. The plug-ins make it easy to use the different coding / decoding / filter libraries without having to learn their API.
- gstreamer will really make it easier to create your custom multimedia application for your embedded system!
- See http://gstreamer.freedesktop.org/documentation/ for details.

Various utilities

libfishsound

- http://www.annodex.net/software/libfishsound/index.html
- Simple programming interface for decoding and encoding audio data using codecs from Xiph.org (mainly Vorbis and Speex).
- License: BSD-like.

Free Software Audio

Miscellaneous

Speech synthesis

Flite: http://www.speech.cs.cmu.edu/flite/
“Festival Lite” from the Carnegie Mellon University

- Festival is a free speech synthesis program,
but it is far from meeting the requirements of embedded systems.
- Unlike Festival, Flite is light (< 4 MB), very fast (very well suited for slow CPUs), portable (written in C), and thread-safe.
- Typically targets devices like PDA, GPS or phones.

Various applications

- Ecasound: http://www.eca.cx/ecasound/
  Multitrack audio processing package. Can be used for simple tasks like audio playback, recording and format conversions, as well as for multitrack effect processing, mixing, recording and signal recycling. Supports a wide range of audio inputs, outputs and effect algorithms.
- LADSPA (Linux Audio Developer's Simple Plugin API)
  http://www.ladspa.org/
  A plugin audio processor framework.
  Several sound effect plugins available (reverb, etc.).

Audio distributions

Useful to discover Linux sound applications!

- Ubuntu Studio: http://ubuntustudio.org/
  A Ubuntu based system (with a live CD)
  for multimedia creating (sound, graphics, video)
- 64-studio: http://www.64studio.com
  Debian based distribution for audio and multimedia creation.

More interesting distributions on http://linux-sound.org/distro.html

Useful reading

- Introduction to Linux Audio, by Filippo Pappalardo
  http://www.osnews.com/story.php?news_id=6720
  A very nice and synthetic review. Good summary.

Free music and sounds

Artists sharing their creations under a free license!
Device makers: can be used in product demos.

- Jamendo: http://www.jamendo.com
  Very popular. Many artists. Many users. All songs seem to be available in both mp3 and Ogg/Vorbis. Artists get some funding with revenues from commercials and gifts from users.
- Freesound: http://www.freesound.org/
  Free sound samples released under a Creative Commons license. Great for making sound capable devices!
- Yahoo Creative Commons Search: http://search.yahoo.com/cc
  Makes it easy to find works released with a Creative Commons license.

More similar sites on http://creativecommons.org/audio/

Useful links

- Sound & MIDI Software For Linux
  http://sound.condorow.net/
  The most exhaustive catalog of Linux audio projects, programs and articles!
- FreeBSD audio software catalog
  http://www.freebsdsoftware.org/audio/
  An extensive list of Unix programs for audio.
- Linux Audio User Guide
  http://lau.linuxaudio.org/
  A collection of documents and HOWTOs.

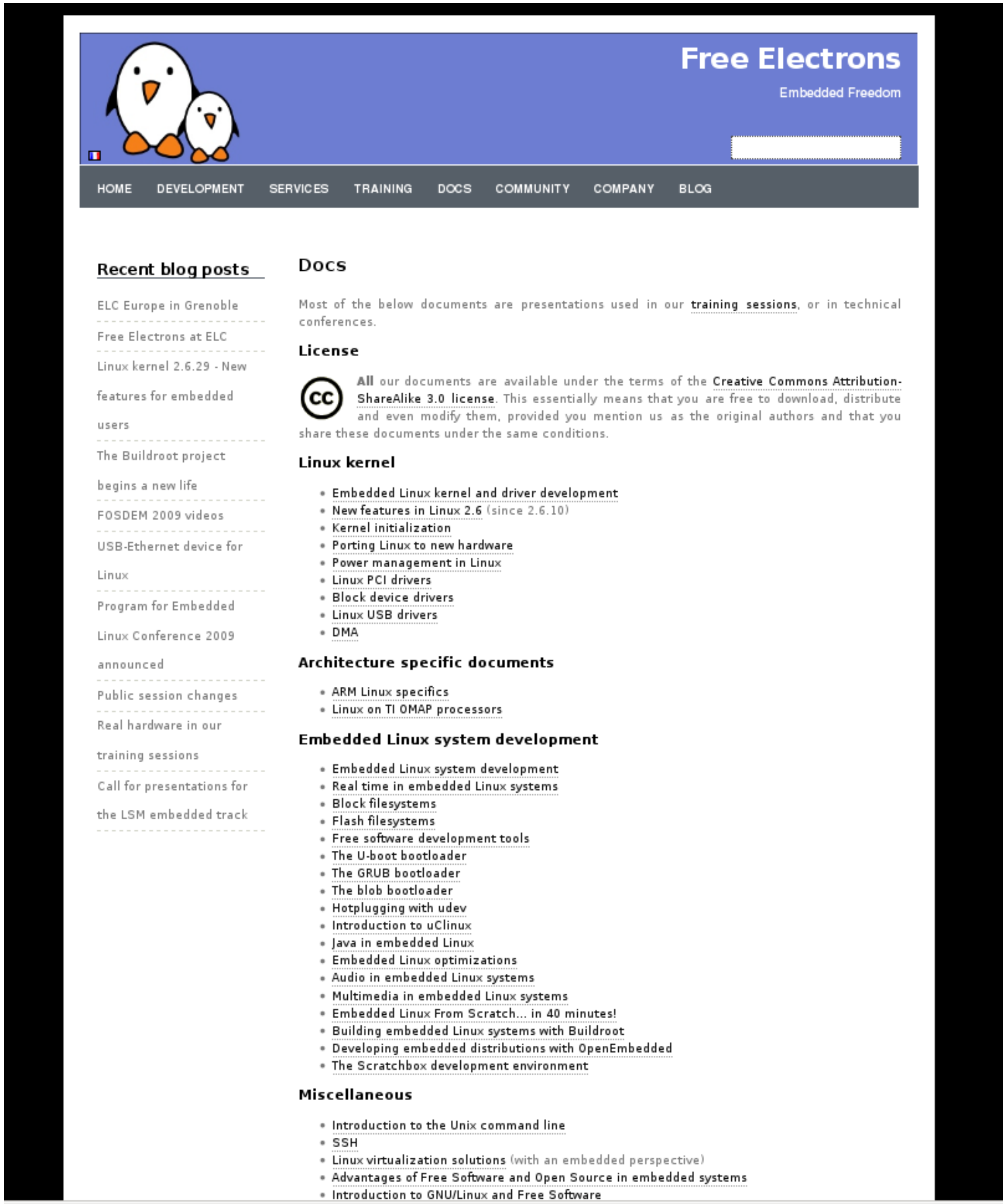
Conclusion

The major strength of the Linux sound solution is again its modularity. Each module takes care of a single task, and does it very well.

- ALSA: provides a unified interface to the hardware.
- Sound server: takes care of managing shared access to sound resources by sound applications.
- Sound libraries: decode, encode, or transform sound.
- User applications: provide given functionalities to the end user.

Another strength is that the whole software solution can be developed on the PC host in parallel with embedded HW and SW development.

Related documents



All our technical presentations
on http://free-electrons.com/docs

- Linux kernel
- Device drivers
- Architecture specifics
- Embedded Linux system development

How to help

You can help us to improve and maintain this document...

- By sending corrections, suggestions, contributions and translations
- By asking your organization to order development, consulting and training services performed by the authors of these documents (see http://free-electrons.com/).
- By sharing this document with your friends, colleagues and with the local Free Software community.
- By adding links on your website to our on-line materials, to increase their visibility in search engine results.

Free Electrons

Our services

Custom Development

System integration
Embedded Linux demos and prototypes
System optimization
Application and interface development

Consulting and technical support

Help in decision making
System architecture
System design and performance review
Development tool and application support
Investigating issues and fixing tool bugs

Linux kernel

Linux device drivers
Board support code
Mainstreaming kernel code
Kernel debugging

Embedded Linux Training

All materials released with a free license!

Unix and GNU/Linux basics
Linux kernel and drivers development
Real-time Linux, uClinux
Development and profiling tools
Lightweight tools for embedded systems
Root filesystem creation
Audio and multimedia
System optimization