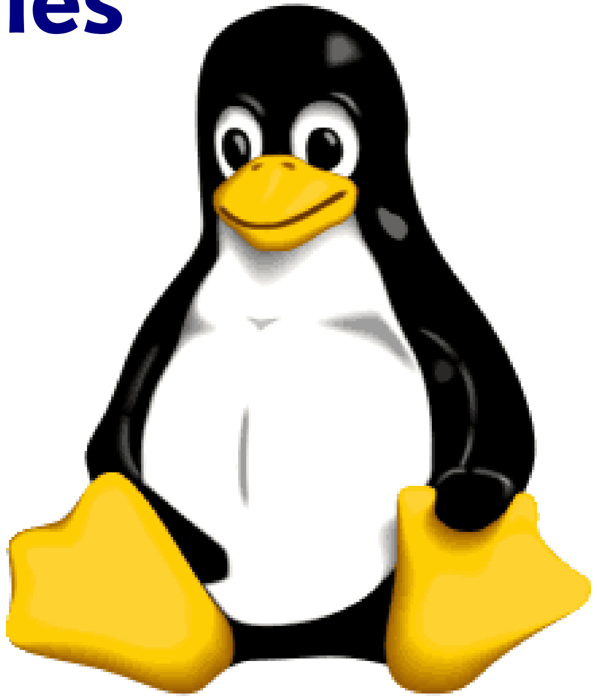
Playing Arbitrary Video Files with GStreamer



GStreamer Conference
Berlin, 2016-10-11
Michael Olbrich <m.olbrich@pengutronix.de>

Pengutronix.

About Me

- Embedded Linux developer
- Working in the Pengutronix userspace team
- Using GStreamer since late 0.10 days

Motivation

- Embedded Linux device
- Video player as one (important) feature
- ongoing development since 2014
- x86 hardware with Intel graphics
- Wayland
- enough resources to play most Videos

Motivation

- This file doesn't work....
  ... but I can play it with VLC

- If I do this: insert 'strange sequence of actions no sane user would do' here ...
  ... the player stops responding

Motivation

- Why was this bug not detected until now?
- What can we learn from it?

Overall Impression

- Playing arbitrary Video files just works in most cases

- The situation is steadily improving

➔ many problems can be fixed by upgrades or backports

Old Formats in new Surroundings

- MPEG1 video file
- As local file:
  - ➔ works as expected
- Via network:
  - ➔ sometime it works sometimes 'broken' audio only

Old Formats in new Surroundings

- type-find and buffer sizes:
- `gst-typefind-1.0 first_2k.mpeg`

```
first_2k.mpeg - audio/mpeg, mpegversion=(int)1,
layer=(int)2, parsed=(boolean)false
```

- 90% probability

- `gst-typefind-1.0 all.mpeg`

```
all.mpeg - video/mpeg, systemstream=(boolean)true,
mpegversion=(int)1
```

- 100% probability but needs >4k bytes for detection

Network Sources and Buffer sizes

- Network sources can produce arbitrary and nondeterministic buffer sizes
- Any element that operates on unstructured data needs to handle this

Push and Pull

- “Audio isn't muted during fast forward if the file comes from the network”

- Matroskademux
- Push mode

➔ GST_SEEK_FLAG_TRICKMODE_NO_AUDIO got lost

Push and Pull

- Some elements have code to handle push and pull mode (mostly demuxer)
- Make sure any change works with both modes

Hardware decoder (vaapi)

- "failed to parse SEI messages"
- Ignore the error or drop the buffer?

- How strict must the code be to avoid problems with the hardware decoder?

Vaapi and Unreliable Transports

- Streaming via rtp / udp
- H.264 decoding with vaapi

➔ decoding stops at the first decode failure caused by packet loss

- Drop frames instead of fatal errors

User Interaction

- 4k video
- play with 10x speed for a few seconds
- play with 1x speed for one second
- pause

➔ pipeline stuck in preroll

- No solution yet
- Workaround: recreate pipeline and seek to last position

Questions?