David Bařina

3. listopadu 2013

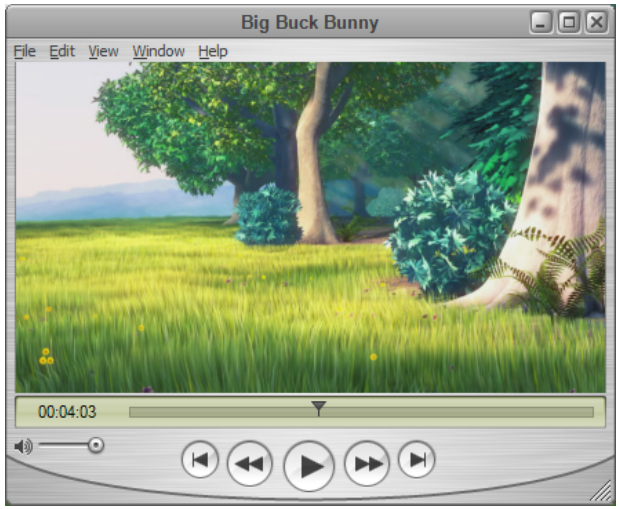
Multimédia



- multimédia:

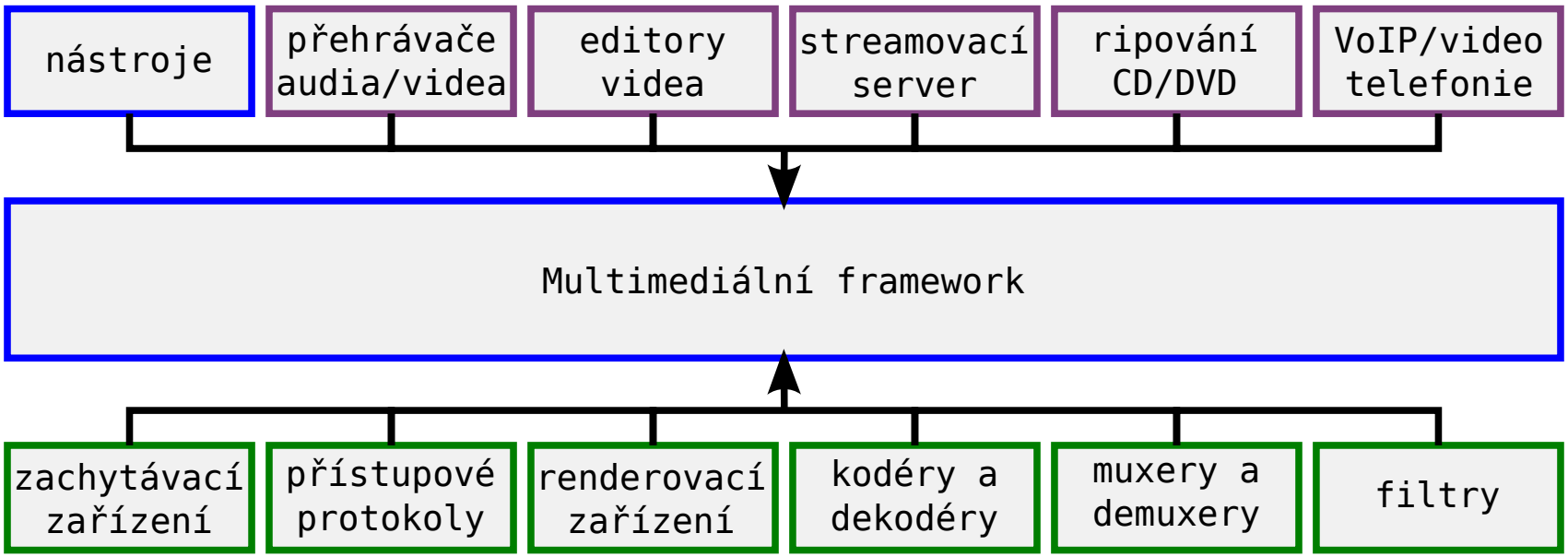
  text, zvuk, statický obraz, **video**, metainformace, ...
- potřeba:
  - získávat (kamera),
  - ukládat (pevný disk, komprese),
  - vyhledávat (podle popisu),
  - přehrávat,
  - upravovat (střih videa), ...
- ukládání: kontejner + kodeky

Multimediální framework

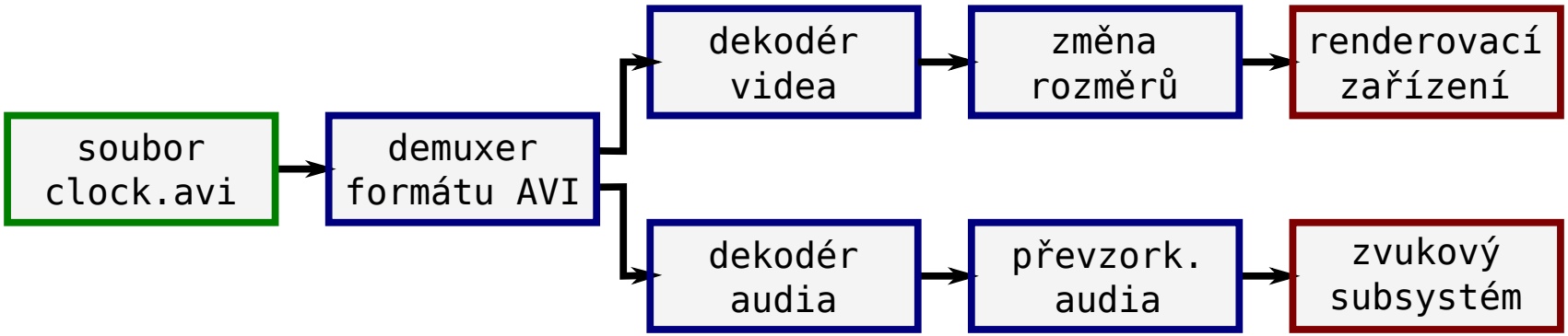


- vše zaobaluje
- knihovny (API), nástroje (přehrávač, CLI)
- formáty: kontejnery, kodeky, protokoly, ...
- požadavky: modularita, široká podpora formátů, intuitivní použití, dokumentace, výkon, platforma, ...
- problém: žádný neumí vše

Multimediální framework



Graf filtrů



pojmy:

- kontejnerový formát, muxer, demuxer/splitter
- formát datového toku, coder, decoder, kodek
- tee, overlay

FFmpeg, knihovny

- svobodný multiplatformní software
- využívají jej MPlayer, VLC media player, Avidemux, ffdshow
- knihovny:
  - libavutil (matematické rutiny, pro zjednodušení programování)
  - libavcodec (audio a video kodeky)
  - libavformat (muxery a demuxery/splittery pro kontejnery)
  - libavdevice (grabování a renderování přes V4L(2), VfW, ALSA)
  - libavfilter (filtry)
  - libswscale (změna rozlišení a barevného modelu obrazu)
  - libswresample (změna vzorkovací frekvence a formátu audia)
- podporované formáty na `http://www.ffmpeg.org/general.html`
- Libav (fork FFmpegu), `http://libav.org/`

Nástroje pro CLI

`ffmpeg`	překódování multimediálních souborů
`ffserver`	streamovací server
`ffplay`	jednoduchý přehrávač založený na SDL
`ffprobe`	zobrazí informace o multimediálních souborech

Příklady

```
ffmpeg -formats
ffmpeg -codecs
ffmpeg -filters
ffmpeg -protocols
ffplay clock.avi
```

Syntaxe příkazů

- syntaxe příkazů

```
ffmpeg [globalni volby] [[volby pro vstup] [-i vstup]]...
[volby pro vystup] vystup...
```

- základní parametry
  - `-i vstup` vstupní soubor
  - `vystup` výstupní soubor
  - `-f format` formát vstupu/výstupu
    - zařízení (`oss`, `alsa`, `x11grab`, `video4linux2`, `fbdev`, `lavfi`, `sdl`)
    - kontejner (`avi`, `image2`, `rawvideo`, `flv`, `rtsp`, `mpegts`, `null`)

Parametry příkazů

- výběr kodeku

  `-b bitrate` datový tok

  `-c kodek` kodek

- výsek videa

  `-ss cas` skok na pozici ve videu

  `-t cas` délka videa (trvání)

  - sekundy
  - `hh:mm:ss[.xxx]`

Parametry příkazů

- video

`-vn`	vypne video
`-c:v kodek`	videokodek
`-b:v bitrate`	datový tok
`-r fps`	snímková frekvence
`-s rozmery`	rozměry
`-vf filtry`	graf filtrů

- audio

`-an`	vypne audio
`-c:a kodek`	audiokodek
`-b:a bitrate`	datový tok
`-af filtry`	graf filtrů

Skok na pozici

`-ss cas`

- rychle a nepřesně, `-ss` pro vstup (před `-i`)

```
ffmpeg -ss 00:03:00 -i input output
```

- pomalu a přesně, `-ss` pro výstup

```
ffmpeg -i input -ss 00:03:00 output
```

- rychle a přesně, `-ss` pro vstup i pro výstup

```
ffmpeg -ss 00:02:30 -i input -ss 00:00:30 output
```

Příkaz ffprobe

- informace o videu

```
ffprobe clock.avi
```

```
Input #0, avi, from 'clock.avi':
  Duration: 00:00:12.00, start: 0.000000, bitrate: 55 kb/s
    Stream #0:0: Video: msrle ([1][0][0][0] / 0x0001), pal8, 321x321,...
    Stream #0:1: Audio: truespeech ([34][0][0][0] / 0x0022), 8000 Hz,...
```

- informace o snímcích

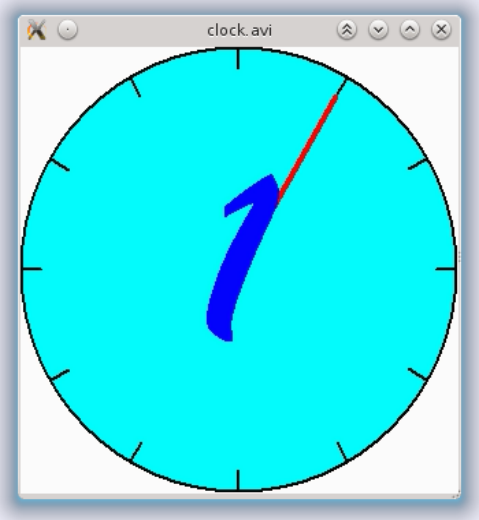
```
ffprobe -show_frames clock.avi
```

```
media_type=video
key_frame=1
pkt_pts=0
pkt_dts=0
pkt_duration=1
width=321
height=321
pix_fmt=pal8
```

Příkaz ffplay

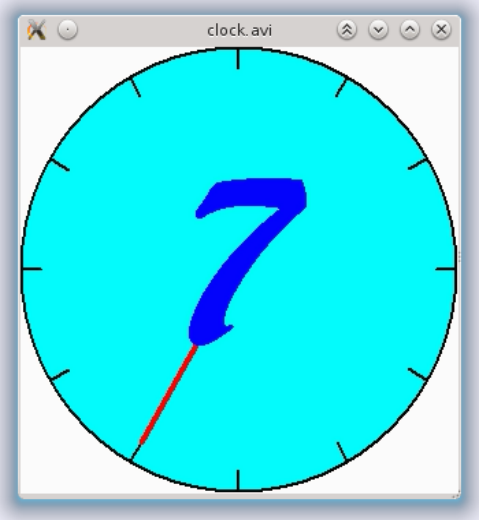
- přehrání videa

```
ffplay clock.avi
```



- skok na pozici

```
ffplay -ss 6 clock.avi
```



Graf filtrů

- jediný filtr

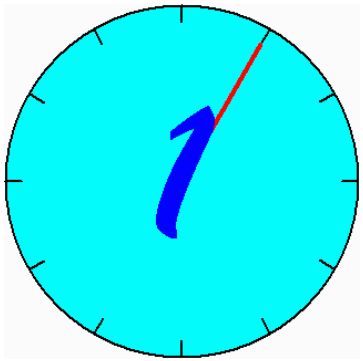
```
ffplay -vf vflip clock.avi
```

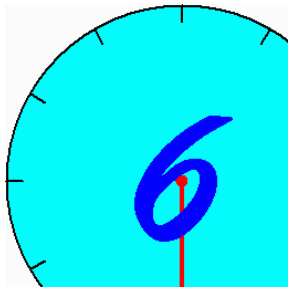
- parametry

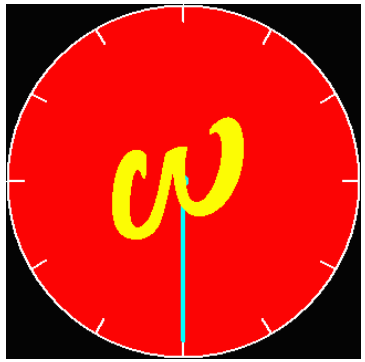
```
ffplay -vf crop=256:256:0:0 clock.avi
```

- řetězec filtrů

```
ffplay -vf "transpose, negate" clock.avi
```



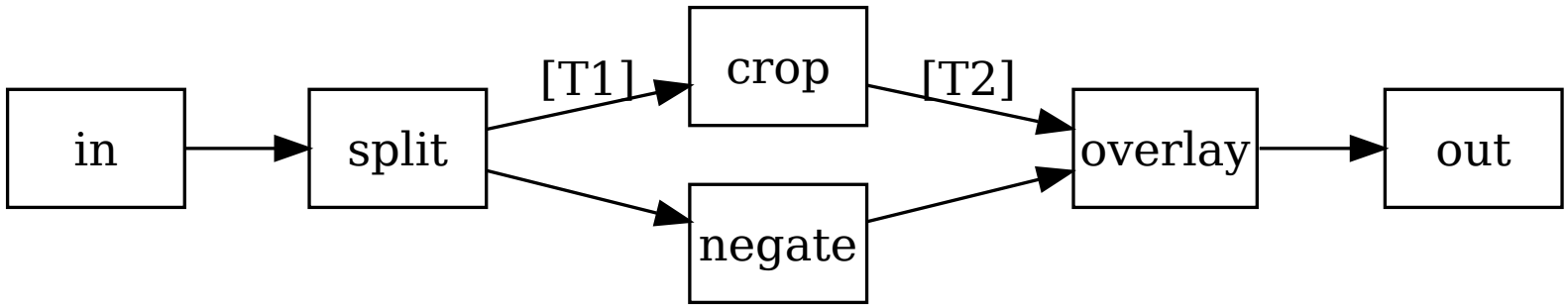


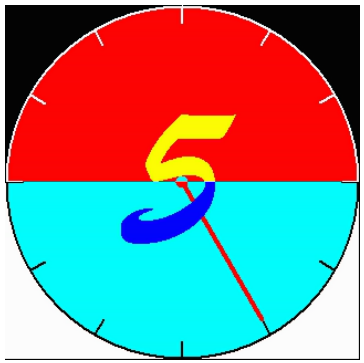


Graf filtrů, větve

- pojmenované pady, větvení

```
ffplay -vf "[in] split [T1], negate, [T2] overlay=0:H/2
[out]; [T1] crop=iw:ih/2:0:ih/2 [T2]" clock.avi
```





Filtry

- klíčové snímky

```
ffplay -vf select='eq(pict_type\,I)' video.mov
```

- dektekce hran

```
ffplay -vf edgedetect -an video.mov
```

- některé video filtry

```
colorbalance, delogo, format, hflip, negate, rotate,
subtitles, unsharp, yadif
```

- dokumentace: `http://ffmpeg.org/ffmpeg-filters.html`

Převod videa

- převod do FFV1 a FLAC v Matroska

```
ffmpeg -i input.avi -c:v ffv1 -c:a flac output.mkv
```

- tok videa 64 kbit/s

```
ffmpeg -i input.avi -b:v 64k output.avi
```

- formát H.264 (libx264)

```
ffmpeg -i input.avi -c:v libx264 -preset fast output.mkv
```

`-preset` může být `fast`, `medium`, `slow` (více v `x264 --fullhelp`)

Práce s webkamerou

- vstup z kamery

```
ffplay -f video4linux2 /dev/video0
```

- záznam i se zvukem

```
ffmpeg -f oss -i /dev/dsp -f video4linux2 -i /dev/video0
output.mpg
```

Streamování po síti

- přes UDP

```
ffmpeg -i file.mkv -c:v h264 -f mpegts
udp://localhost:1234
```

následně např. `vlc udp://@:1234`

Streamování po síti

- přes HTTP pomocí `ffserver`

ffserver.conf

```
Port 8090
BindAddress 0.0.0.0
<Feed feed1.ffm>
    ...
</Feed>
<Stream live.flv>
    ...
</Stream>
```

```
ffserver -f ffserver.conf
```

```
ffmpeg -i file.mkv http://localhost:8090/feed1.ffm
```

následně např. `http://localhost:8090/live.flv`

API

1. nainstalovat/přeložit FFmpeg
   - `pkg-config --cflags --libs libavformat`
2. překlad aplikace

```
cc -I/usr/include/libavformat app.c -lavformat -o app
```

3. hlavičkové soubory, např. `#include <avformat.h>`
   - v C++ obalit `extern "C"`
4. na začátku zavolat

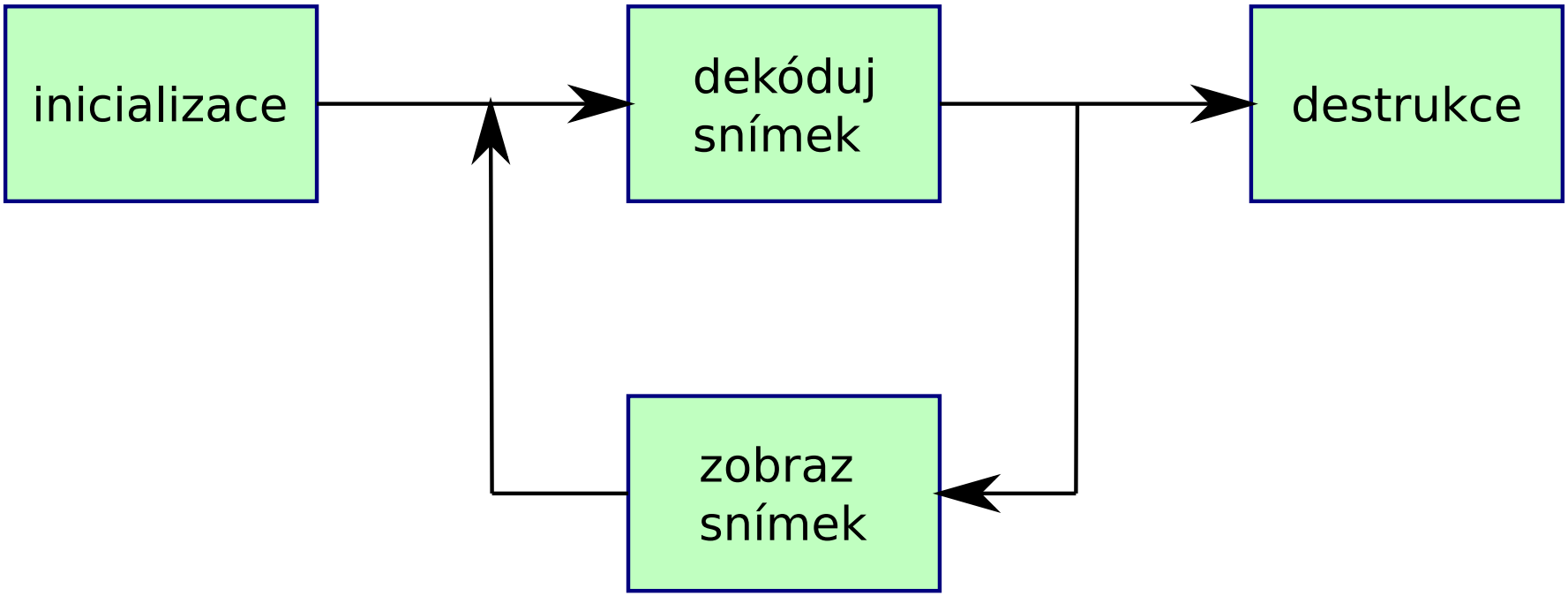
```
av_register_all();
```

5. používat funkce z libavformat

Otevření video stopy

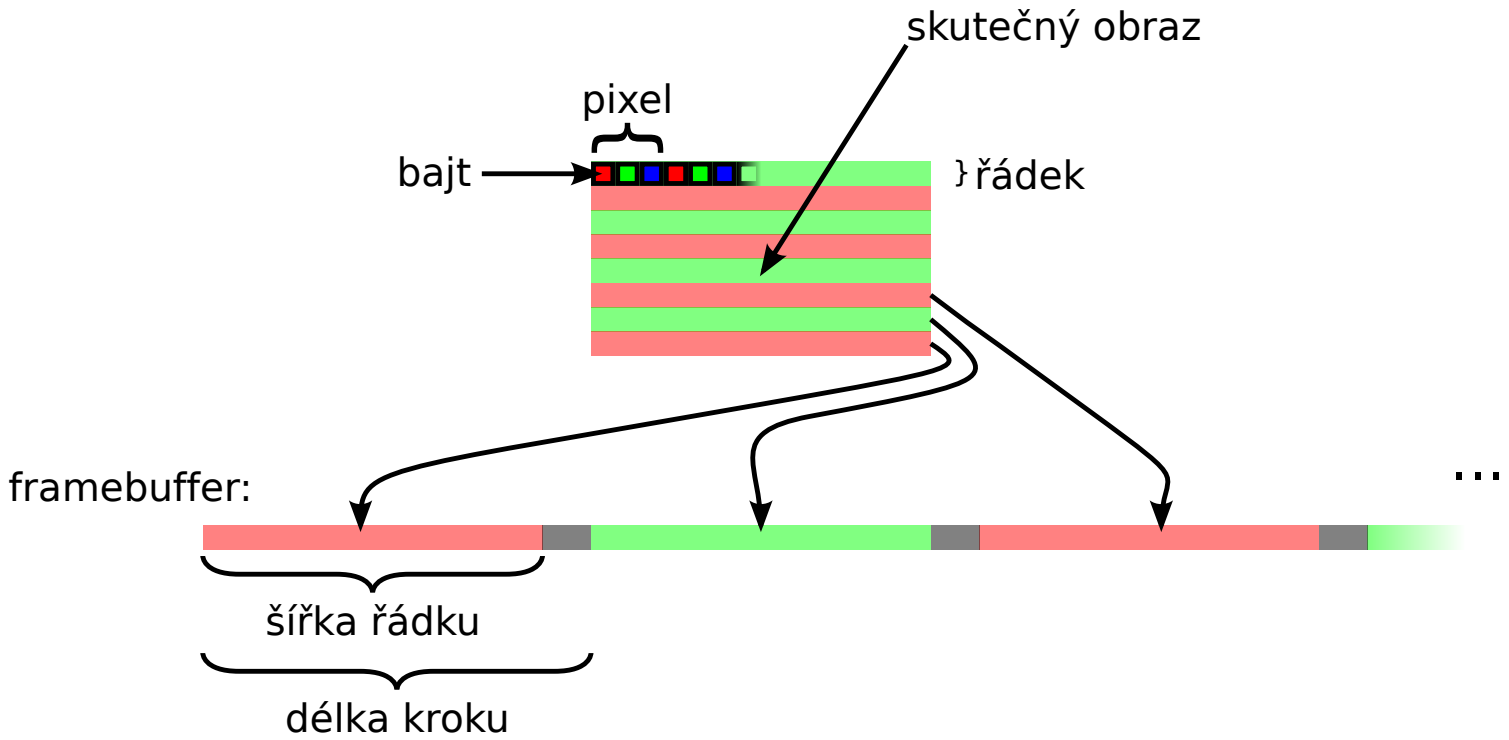
```
#include <avcodec.h>
#include <avformat.h>

int main(int argc, charg *argv[])
{
        av_register_all();

        AVFormatContext *pFormatCtx;

        if(av_open_input_file(&pFormatCtx, argv[1], NULL, 0, NULL) != 0)
                return -1;

        if(av_find_stream_info(pFormatCtx) < 0)
                return -1;

        AVCodecContext *pCodecCtx;

        if(pFormatCtx->streams[0]->codec.codec_type != CODEC_TYPE_VIDEO)
                return -1;

        pCodecCtx = &pFormatCtx->streams[0]->codec;
```

Přehrávač



Pojmy

- barevný model (RGB, YC_bC_r)
- formát pixelu (RGB24)
- framebuffer



Smyčka přehrávače

```
AVPacket pkt;

while( av_read_frame(pFormatCtx, &pkt) == 0 )
{
        if( pkt.stream_index == videoStream )
        {
                int frameFinished = 0;
                if( avcodec_decode_video2(pCodecCtx, pFrame, &frameFinished,
                        &pkt) < 0 )
                        abort();
                if(frameFinished)
                {
                        // sws_scale

                        // avcodec_encode_video2

                        // ...
                }
        }
        av_free_packet(&pkt);
}
```

Kostra kodeku

```
static int dbv1_decode_frame(AVCodecContext *avctx,
                             void *outdata, int *outdata_size,
                             const uint8_t *buf, int buf_size)
{
        // dekoduj snimek
}

AVCodec dbv1_decoder =
{
        .name           = "dbv1",
        .type           = CODEC_TYPE_VIDEO,
        .id             = CODEC_ID_DBV1,
        .priv_data_size = sizeof(DBV1Context),
        .init           = dbv1_decode_init,
        .close          = dbv1_decode_close,
        .decode         = dbv1_decode_frame,
        .long_name      = NULL_IF_CONFIG_SMALL("DaBler's Video codec v1"),
        .capabilities   = CODEC_CAP_DR1,
};
```

- přeložit modul + libavcodec + libavformat

Přehled funkcí

- `avformat_open_input` otevře vstupní kontejner, přečte hlavičku
- `avformat_find_stream_info` načte z kontejneru informace
- `av_dump_format` zobrazí informace o kontejneru a stopách
- `avcodec_find_decoder` podle ID kodeku najde dekodér
- `avcodec_find_encoder` podle ID kodeku vrátí kodér
- `avcodec_alloc_frame` alokuje snímek
- `av_read_frame` přečte z kontejneru jeden paket (snímek)
- `avformat_write_header` zapíše do kontejneru hlavičku stopy
- `av_write_frame` zapíše do kontejneru paket
- `av_write_trailer` zapíše do kontejneru patičku stopy
- `avcodec_decode_video2` z paketu dekóduje jeden snímek videa
- `avcodec_encode_video` zkomprimuje snímek videa do bufferu
- `avformat_new_stream` přidá do kontejneru novou stopu

Zdroje informací

- `man ffmpeg`
- `http://trac.ffmpeg.org/wiki`
- `http://www.ffmpeg.org/`

David Bařina

3. listopadu 2013

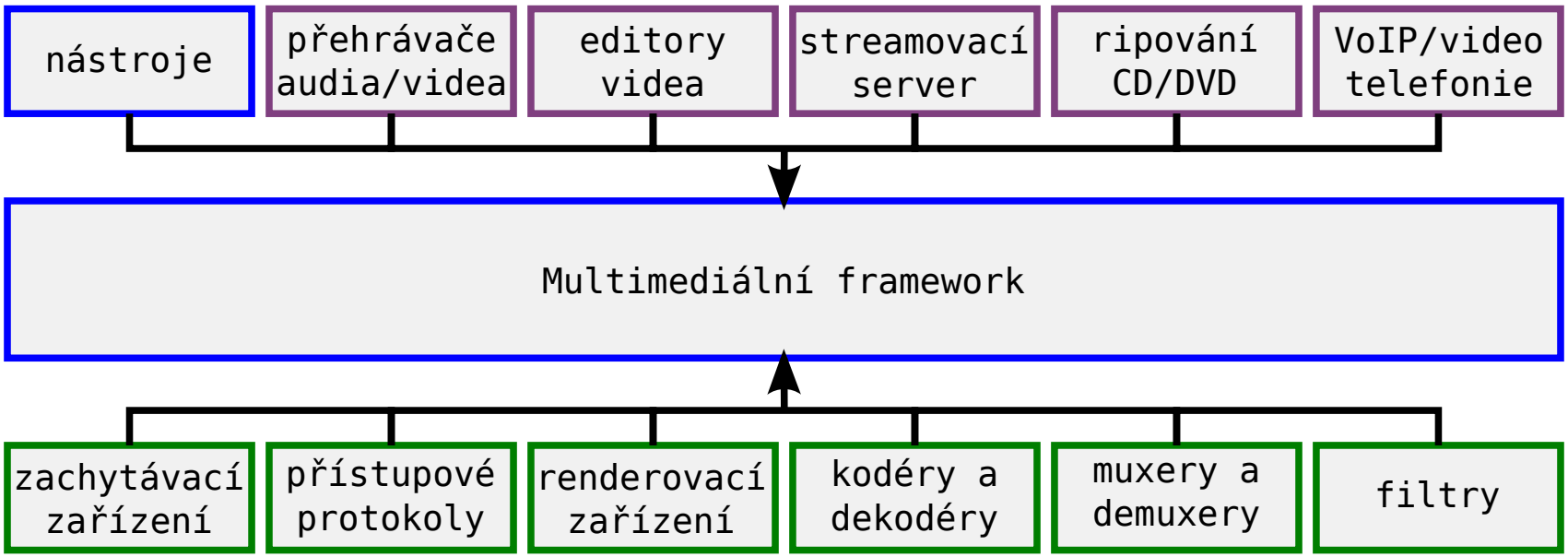
Multimédia



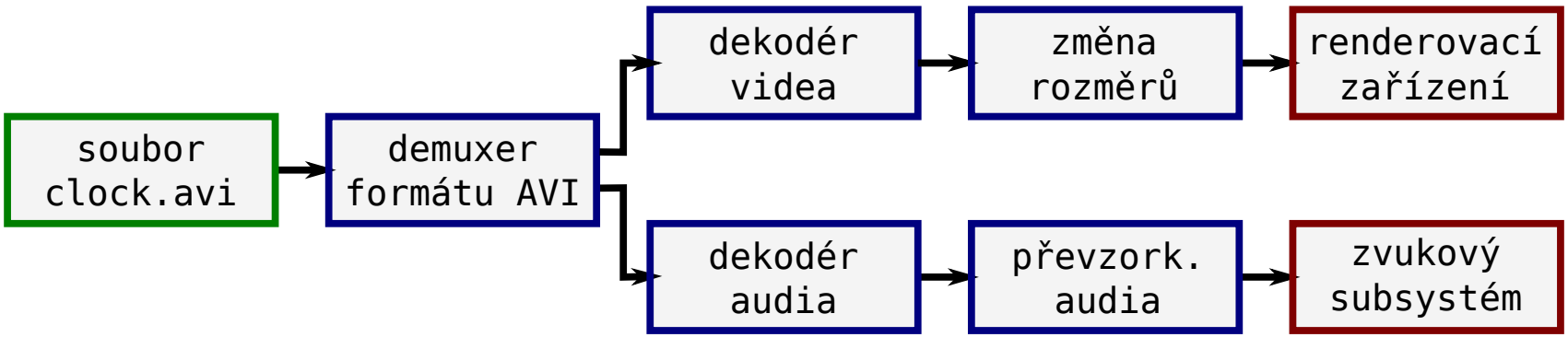
- multimédia:

  text, zvuk, statický obraz, **video**, metainformace, ...
- potřeba:
  - získávat (kamera),
  - ukládat (pevný disk, komprese),
  - vyhledávat (podle popisu),
  - přehrávat,
  - upravovat (střih videa), ...
- ukládání: kontejner + kodeky

Multimediální framework



Graf filtrů



- modely pro přenos dat
  - push – zdroj neustále produkuje data, další filtr pasivně přijímá
  - pull – filtr aktivně požaduje data (parser od zdroje)
- data předávána v bufferech
- stavy: zastaven, pozastaven, spuštěn

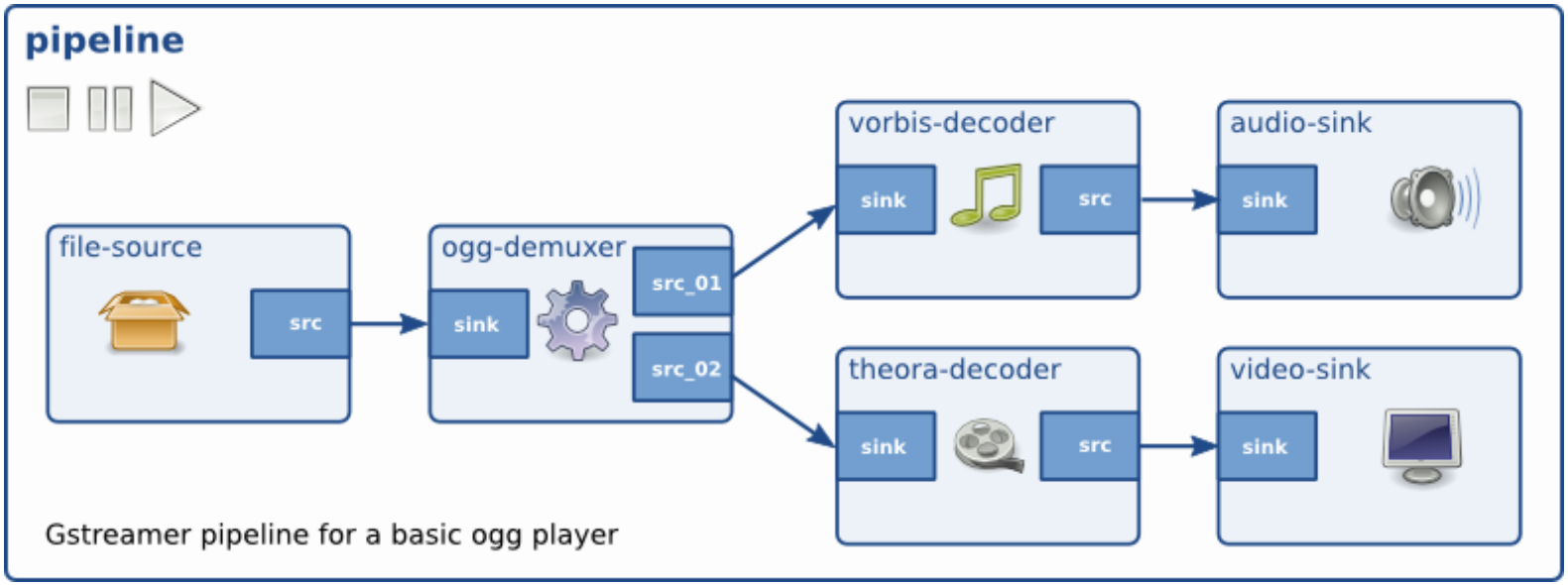
GStreamer

- svobodný multiplatformní, 1999
- založen na GLib, primárně pro GNOME
- založen na grafu filtrů (pipeline), jako DirectShow
- nástroje: `gst-launch`, `gst-inspect`, `gst-editor`
- terminologie
  - pads = spoje mezi filtry
  - source pad se propojí do sink pad
  - typ dat se zjistí pomocí capabilities
  - element, bin, pipeline
- tři balíčky pluginů: The Good, the Bad and the Ugly

Terminologie

- elementy
  - zdrojové (source), filtry, cílové (sink)
  - seskupeny do kontejnerů (binů)
  - tvoří pipeline
- přípojné body (pad)
  - spojují elementy
  - vstupní (sink), výstupní (source)
  - podporovaný typ dat (capabilities)
- capabilities
  - data identifikována pomocí typů MIME
  - např. `audio/x-vorbis`, `audio/x-raw-float`
  - formát dat je třeba vyjednat podle podporovaných

Pipeline



Sestavení pipeline

```
export GST_PLUGIN_PATH=./.libs

gst-launch-0.10 v4l2src device="/dev/video0" ! videoscale ! video/x-raw-yuv,
width=160 ! ffmpegcolorspace ! video/x-raw-gray ! abr2 ! ffmpegcolorspace !
videoscale ! video/x-raw-rgb, width=640 ! ximagesink
```

Nástroj gst-inspect

gst-inspect-0.10

```
alsa: alsasrc: Audio source (ALSA)
```

gst-inspect-0.10 alsa

```
Plugin Details:
  Name: alsa
  Description: ALSA plugin library
  Filename: /usr/lib64/gstreamer-0.10/libgstalsa.so

  alsasink: Audio sink (ALSA)
  alsasrc: Audio source (ALSA)
  alsamixer: Alsa mixer
```

gst-inspect-0.10 alsasrc

...

Nástroj gst-inspect

gst-inspect vorbisdec

```
Pad Templates:
  SRC template: 'src'
    Availability: Always
    Capabilities:
      audio/x-raw-float
                    rate: [ 8000, 50000 ]
                channels: [ 1, 2 ]
              endianness: 1234
                   width: 32
          buffer-frames: 0

  SINK template: 'sink'
    Availability: Always
    Capabilities:
      audio/x-vorbis
```

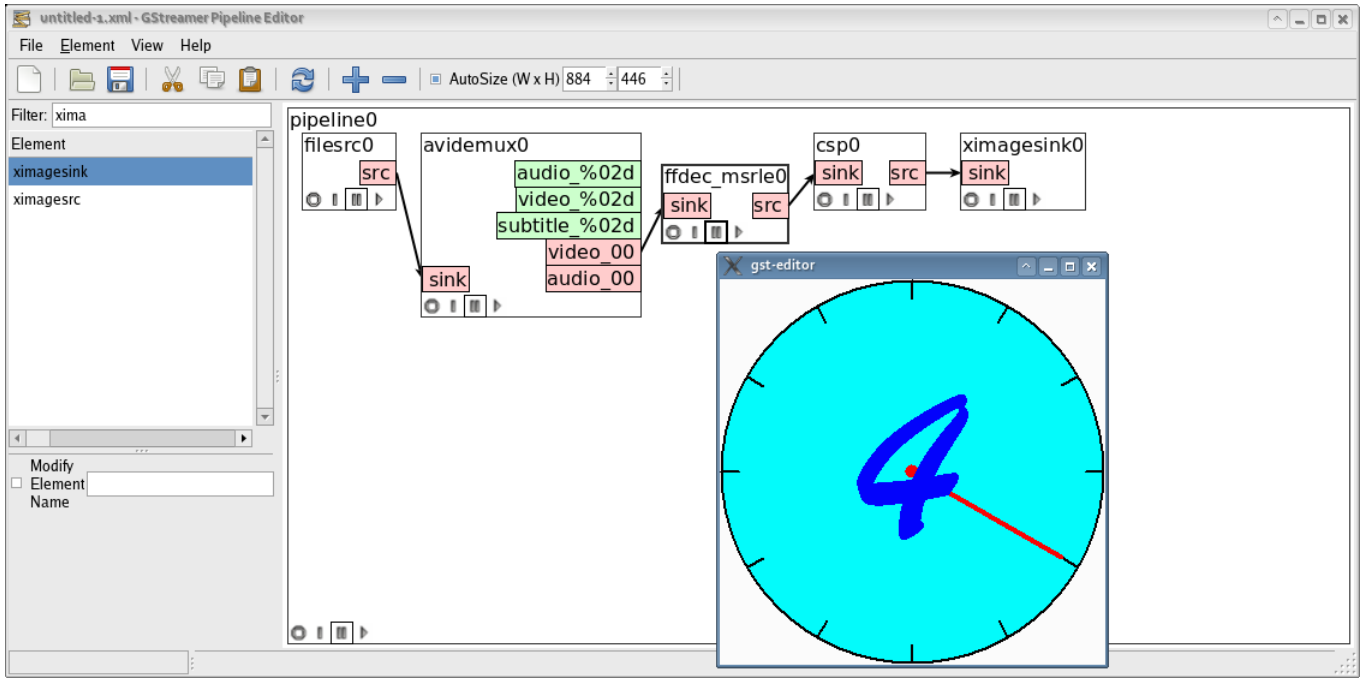
Nástroj gst-launch

```
gst-launch-0.10 videotestsrc ! ximagesink
```

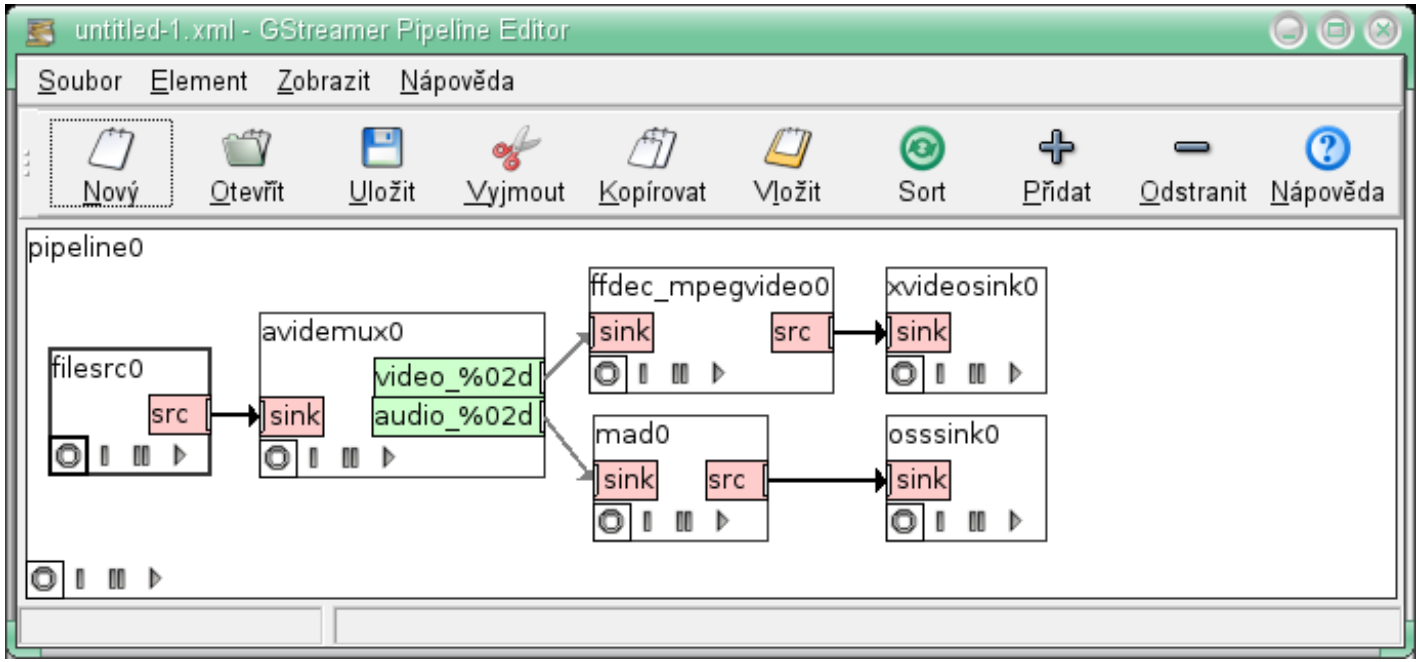
```
gst-launch-0.10 videotestsrc ! videoflip method="2" ! ximagesink
```

Nástroj gst-launch: přehrávač

```
gst-launch-0.10 playbin uri=file:///tmp/clock.avi

gst-launch-0.10 filesrc location=/tmp/clock.avi ! decodebin !
        colorspace ! ximagesink

gst-launch-0.10 filesrc location=/tmp/clock-rle.avi ! avidemux !
        ffdec_msrle ! colorspace ! ximagesink
```



Nástroj gst-editor, XML



Uložení/načtení pipeline

```
gst_xml_write_file (GST_ELEMENT (pipeline), fopen ("xmlTest.gst", "w"));

xml = gst_xml_new ();
ret = gst_xml_parse_file(xml, "xmlTest.gst", NULL);
g_assert (ret == TRUE);
pipeline = gst_xml_get_element (xml, "pipeline");
g_assert (pipeline != NULL);
gst_element_set_state (pipeline, GST_STATE_PLAYING);
```

Vlastní aplikace

1. nainstalovat/přeložit
   - `pkg-config --cflags --libs gstreamer-0.10`
2. hlavičkový soubor `<gst/gst.h>`
3. zavolat funcki `gst_init()`
4. používat funkce gstreameru

Triviální aplikace

```
#include <gst/gst.h>

int main(int argc, char *argv[])
{
    gst_init(&argc, &argv);

    g_print("Started...\n");

    return 0;
}
```

Spojení dvou elementů

```
GstElement *pipeline = gst_pipeline_new("pipeline");
GstElement *source = gst_element_factory_make(
    "videotestsrc", "source");
GstElement *sink = gst_element_factory_make(
    "ximagesink", "sink");

gst_bin_add_many(GST_BIN(pipeline), source, sink, NULL);
gst_element_link(source, sink);
gst_element_set_state(pipeline, GST_STATE_PLAYING);

...

gst_element_set_state(pipeline, GST_STATE_NULL);
gst_object_unref(GST_OBJECT(pipeline));
```

Smyčka zpráv

```
static GMainLoop *loop;
static gboolean bus_callback(GstBus *bus,
    GstMessage *message, gpointer data)
{
  if(GST_MESSAGE_TYPE(message) == GST_MESSAGE_EOS)
    g_main_loop_quit(loop);
  return TRUE;
}

GstBus *bus = gst_pipeline_get_bus(
  GST_PIPELINE(pipeline));
gst_bus_add_watch(bus, bus_callback, NULL);
gst_object_unref(bus);

loop = g_main_loop_new(NULL, FALSE);
g_main_loop_run(loop);
```

Zásuvné moduly

Plugin

```
$ git clone git://anongit.freedesktop.org/gstreamer/gst-template.git

$ ../tools/make_element abr2

static gboolean abr2_init (GstPlugin * abr2) {
        // ...
}
static GstFlowReturn gst_abr2_chain (GstPad * pad, GstBuffer * buf) {
        // ...
        GstStructure *structure = gst_caps_get_structure (pad->caps, 0);
        gst_structure_get_int (structure, "width", &width);
        gst_structure_get_int (structure, "height", &height);
        // ...
        img.imageData = (char*) GST_BUFFER_DATA(buf);
        // ...
}

$ ./autogen.sh

$ make

$ export GST_PLUGIN_PATH=./.libs

$ gst-launch-0.10 v4l2src device="/dev/video0" ! videoscale ! video/x-raw-yuv,
width=160 ! ffmpegcolorspace ! video/x-raw-gray ! abr2 ! ffmpegcolorspace !
videoscale ! video/x-raw-rgb, width=640 ! ximagesink
```

- kodek: zkompilovat jen plugin

Přehled API

`GstObject`	základní třída objektové hierarchie
`GstElement`	abstraktní třída pro všechny elementy
`GstElementFactory`	třída šablony elementu
`GstPad`	přípojný bod, má směr `GstPadDirection`
`GstCaps`	formáty dat
`GstBin`	kontejner elementů
`GstPipeline`	celá pipeline
`GstBus`	sběrnice zpráv
`GstMessage`	zpráva

`gst_init`	inicializuje knihovnu
`gst_element_factory_make`	vytvoří element z názvu šablony
`gst_pipeline_new`	vytvoří novou pipeline
`gst_bin_add_many`	přidá do kontejneru několik elementů
`gst_pad_link`	propojí dva pady
`gst_bus_add_watch`	přidá ke sběrnici callback funkci

Zdroje

- `http://gstreamer.freedesktop.org/documentation/`
  (Application Development Manual, Plugin Writer's Guide)