Introdução ao *framework* Gstreamer

Thilo Koch
Grupo Computação Musical - IME - USP

11 de março de 2014

A Apresentação dará uma visão geral do Gstreamer, conceitos e elementos básicos assim como ferramentas simples e exemplos de uso.

Gstreamer é

- um *framework* para desenvolver aplicações multimídias
- uma biblioteca para construir grafos de componentes para processamento de áudio e vídeo
- usado por exemplo para:
  - tocadores de música e de vídeo
  - editores, transformadores (filtros), codificadores
  - *streaming* media *broadcasters*
  - aplicações de síntese
  - como base para outros *frameworks*

Panorama

- o projeto existe desde 1999, primeiro lançamento em 2001
- escrito principalmente em C
- software livre (licenciado com LGPL versão 2)
- **Versões**:
  - atual: 1.2.3 (chamada série 1.0)
  - estável desde 2005: 0.10
- **Multiplataforma**: Linux, Solaris, MS Windows, Mac OSX, Symbian, Android, iOS
- **Bridge**: conectar a outros *frameworks* como DirectShow (Win), QuickTime (Mac OS X), OMX (OpenMAX)
- **Bindings**: Python, Perl, .NET, C++, Qt, Guile, Haskell, Java, Ruby, Vala

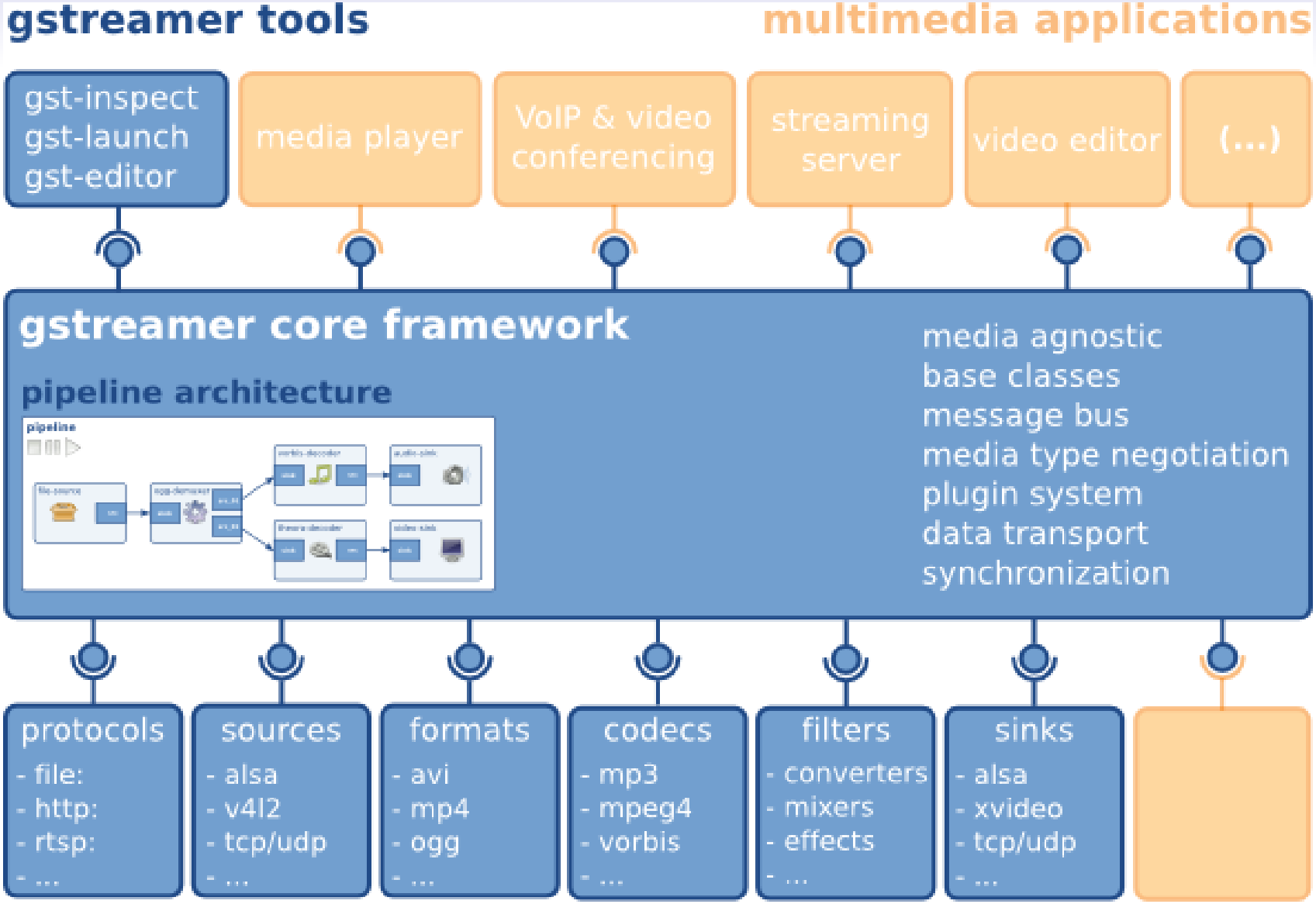
Pacotes

Nome de pacotes Linux - `gstreamer1-plugins-<spec>`

		Compatibilidade da Licença	
		LGPL 2	*não* compatível ou problemas com patentes
Qualidade de código	boa	• base • good • good-extras	• ugly
	mais ou menos, pouco testado	• bad-free • bad-free-extras	• bad-freeworld

Conceitos de design

- **Orientação ao objeto**: usando o modelo GObject da GLib2 (GTK+) com sinais e propriedades (para por exemplo pesquisar capacidades no tempo de execução)
- **Interfaces separadas**:
  - API para *programador de aplicação*: construir uma pipeline (grafo) usando plugins como componentes
  - API para *programador de plug-in*: desenvolver plugins auto-contingentes que podem ser auto-carregados na hora de execução da aplicação
- **Separação core e plugin**: core é agnóstico quanto ao tipo de mídia, só os plugins conhecem tipos de mídia e comandam o núcleo como processar e manipular os dados
- **Alto desempenho**:
  - Gstreamer usa mecanismo da GLib para alocação de memória
  - passar dados é feito com ponteiros
  - acesso direto à memória (também diretamente à memória da placa de áudio)
  - forte apoio a *multi-threading*
  - *threads* dedicados somente ao processamento de dados (*streaming threads*) escalonados pelo kernel
  - uso de aceleração de hardware em plugins


gstreamer tools
multimedia applications
gst-inspect
gst-launch
gst-editor
media player
VoIP & video conferencing
streaming server
video editor
(...)
gstreamer core framework
pipeline architecture
media agnostic
base classes
message bus
media type negotiation
plugin system
data transport
synchronization
protocols
- file:
- http:
- rtsp:
- ...
sources
- alsa
- v4l2
- tcp/udp
- ...
formats
- avi
- mp4
- ogg
- ...
codecs
- mp3
- mpeg4
- vorbis
- ...
filters
- converters
- mixers
- effects
- ...
sinks
- alsa
- xvideo
- tcp/udp
- ...
gstreamer plugins
gstreamer includes over 250 plugins
3rd party plugins

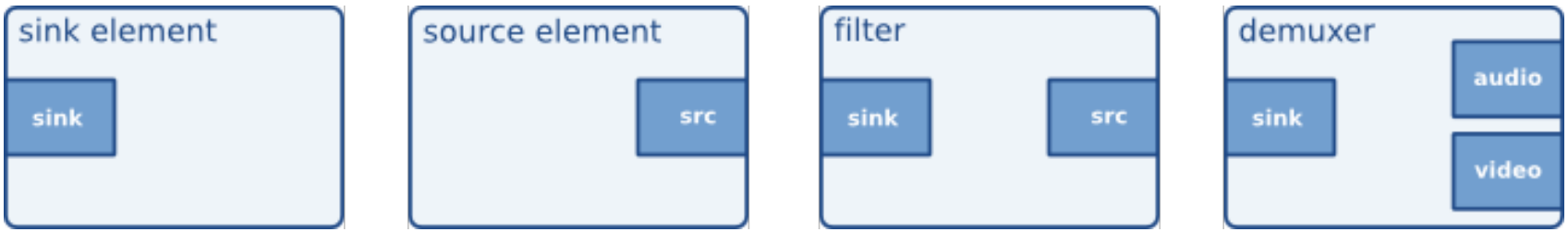
Pedras básicas

Element (gst-element)

- a classe mais importante
- os dados fluem por uma corrente (pipeline) de *elements*
- normalmente tem uma única função (ler dados, decodificar, amortecer, colocar dados na placa de áudio, ...)
- recebe e pode mandar eventos, mensagens e pesquisas
- pode ter vários *pads* e adicionar / apagar *pads* dinamicamente (*at runtime*)

Pads

- conectores (*port*, *plug*) entre os *elements* para estabelecer fluxo de dados
- tem capacidades que descrevem e restringem os tipos de dados que podem fluir por eles
- capacidades são usadas nas *negociações de tipos* (formatos) entre os *elements*
- *elements* aceitam dados em *sink pads*, *source pads* funcionam como saída



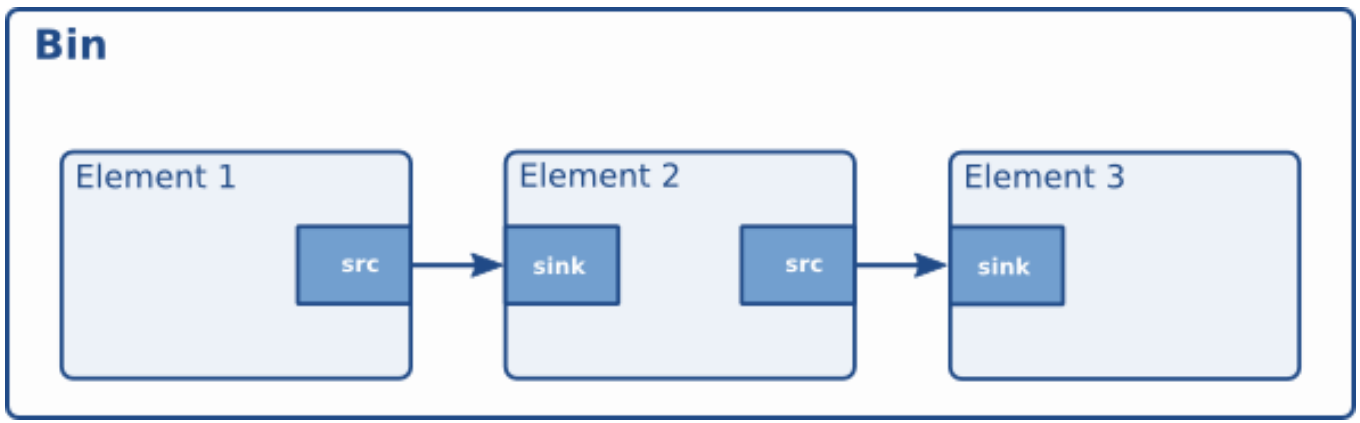
Mais pedras básicas

Bin

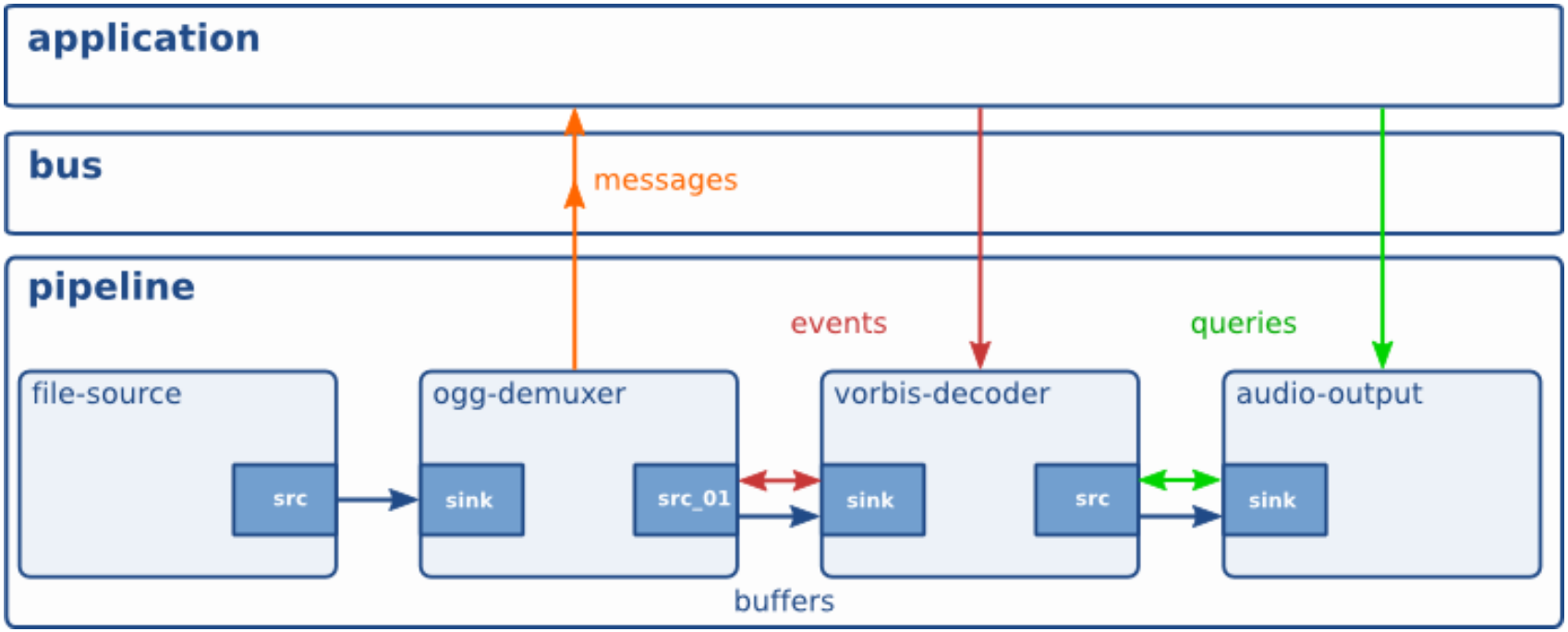
- são *containers* para juntar e conectar *elements*
- podem conter outros *bins* (para construir hierarquias / abstrações)
- os *elements* são controlados através do *bin* que os contém (*start*, *stop*, *seek*, *pads* dinâmicos)

Pipeline

- um *top-level-bin*
- responsável pela sincronização dos filhos
- oferece serviço de barramento (*bus*) que transmite mensagens, eventos, pesquisas
- está em um dos estados: *PAUSED, PLAYING, READY, NULL, PREROLL*



Fluxo de dados numa pipeline / Comunicação



- **mensagens**: troca de informações (por exemplo: metadata, mensagens de erro) entre aplicação e elementos ou entre elementos
- **eventos**: podem viajar em ambas as direções da pipeline, por exemplo: pedidos de mudança de estado, de *seeks* etc.
- **pesquisas**: descobrir capacidades dos pads, posição no stream

Conceitos avançados

- **Negociação de capacidades**: o Gstreamer oferece a possibilidade de autoconfiguração para adaptar os elementos entre si através de um método de negociação.
  A negociação de capacidades é o ato de encontrar um formato que dois elementos conectados são capazes de processar (indicado pelas capacidades dos *pads*). Esse processo é realizado através de mensagens de pesquisa e mensagens de eventos.
  Esquema básico do processo de negociação:
  - um elemento *downstream* supõe um formato para os elementos para *upstream*
  - um elemento *upstream* decide o formato e manda evento com a decisão para *downstream*
  - um elemento *downstream* pode pedir a reconfiguração, recomeçando o processo de negociação
- **Relógios e sincronização**: o Gstreamer oferece vários mecanismos de sincronização para cumprir diferentes *casos de uso*
- **Amortecedores**: existem *buffers* de vários tipos para determinados usos, por ex. para downloads, *streams, timeshift buffering, live buffering*
- **Qualidade de serviço**: apoio a mecanismos de qualidade de serviço (*QoS*)
- **[...]**

Ferramenta - gst-discoverer

Mostra informações sobre o formato de uma mídia (arquivo, stream ...)

```
$ gst-discoverer http://docs.gstreamer.com/media/sintel_trailer-480p.webm

Analyzing http://docs.gstreamer.com/media/sintel_trailer-480p.webm
Done discovering http://docs.gstreamer.com/media/sintel_trailer-480p.webm

Topology:
  container: WebM
    audio: Vorbis
    video: VP8

Properties:
  Duration: 0:00:52.250000000
  Seekable: yes
  Tags:
      video codec: VP8 video
      container format: Matroska
      language code: en
      application name: ffmpeg2theora-0.24
      encoder: Xiph.Org libVorbis I 20090709
      encoder version: 0
      audio codec: Vorbis
      nominal bitrate: 880000
      bitrate: 80000
```

Ferramenta - gst-inspect

Mostra informações sobre o plug-in

- nome, classe, autores, descrição, versão, licença, url
- propriedades: nome, qualidade de serviço, propriedades específicas
- informações sobre capacidades dos pads (como formato, bitdepth, tamanho, framerate)

$ gst-inspect videoflip

```
[...]
Element Properties:
  name                : The name of the object
                        flags: legível, gravável
                        String. Default: "videoflip0"
  qos                 : Handle Quality-of-Service events
                        flags: legível, gravável
                        Boolean. Default: true
  method              : method
                        flags: legível, gravável, controlável
                        Enum "GstVideoFlipMethod" Default: 0, "none"
                           (0): none             - Identity (no rotation)
                           (1): clockwise        - Rotate clockwise 90 degrees
                           (2): rotate-180       - Rotate 180 degrees
                           (3): counterclockwise - Rotate counter-clockwise 90 degrees
                           (4): horizontal-flip  - Flip horizontally
                           (5): vertical-flip    - Flip vertically
                           (6): upper-left-diagonal - Flip across upper left/lower right diagonal
                           (7): upper-right-diagonal - Flip across upper right/lower left diagonal
[...]
```

Ferramenta - gst-launch

- uma ferramenta para construir pipelines a partir da linha de comando
- usa sintaxe elaborada para descrever / definir a pipeline

Exemplos

Tocar 1 CD de áudio

```
$ gst-launch cdiocddasrc track=2 !  autoaudiosink
```

- **cdiocddasrc** lê o track 2 do CD
- **autoaudiosink** seleciona automaticamente a saída de som viável
- **!** conecta o *source pad* do primeiro *element* com *sink pad* do segundo *element*

Mostrar o que a webcam capta e girar a imagem

```
$ gst-launch v4l2src device=/dev/video1 !  videoflip
method=counterclockwise !  xvimagesink
```

- **v4l2src** a entrada é a webcam
- **videoflip** filtro que vira imagem em $\pi/2$
- **xvimagesink** mostra na tela com *driver* Xv protocol (xorg)

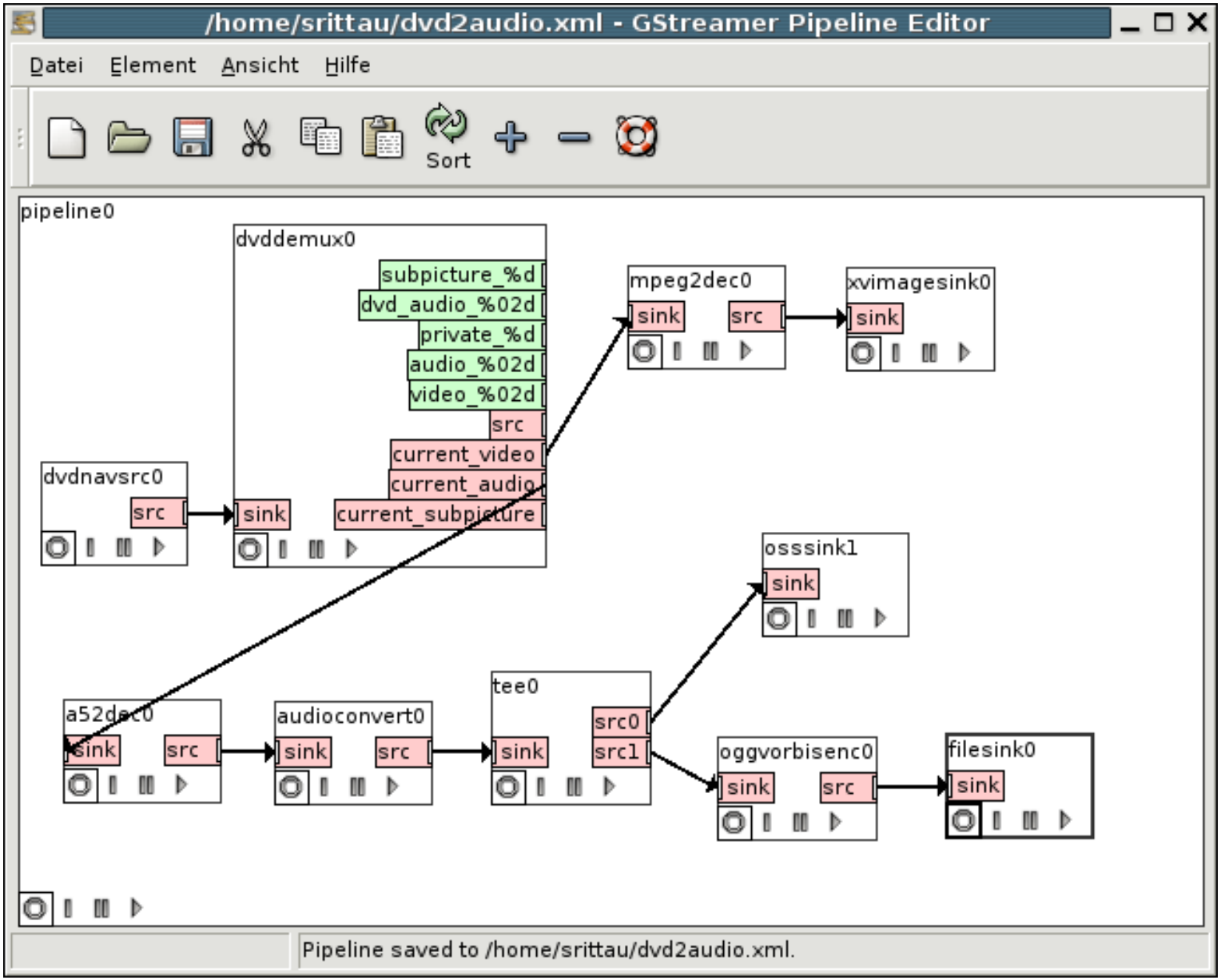
Ferramenta - gst-launch cont.

Mostrar *stream* da webcam na tela e salvar no disco

```
$ gst-launch v4l2src ! 'video/x-raw-yuv,width=640,height=480' !
tee name=t_vid ! queue ! xvimagesink sync=false
t_vid.! queue ! videorate ! 'video/x-raw-yuv,width=640,height=480' !
theoraenc ! queue ! oggmux !
filesink location=me_video.ogv
```

- **'video/x-raw-yuv,width=640,height=480'** selecionar o *source pad* com essas propriedades
- **tee name=t_vid** elemento T: duplicar o stream que chega no sink e denomina o *source pad* adicional *t_vid*
- **queue** uma queue (para ter um *buffer*)
- **t_vid.!** conecta stream com nome *t_vid*

Ferramenta - gst-editor - GUI



Plugins

de entrada

- Alsa, Jack, DirectSound, Firewire, TCP/UDP, RTSP, video4linux2
- arquivo, ximagesrc (*videostream* de screenshots)

de saída

- a maioria dos plugins que existem para entrada, porém "invertidos"
- curl - upload com http / ftp / smtp
- autoaudiosink, autovideosink - selecionam automaticamente uma saída viável

Filtros, Transformadores

- **(de)codificar, (de)multiplex** Flac, Jpeg, Vp8, A52, DVD, Matroska, Id3demux
- **filtros** dinâmica, FIR- e IIRfilter, equalizador, videocrop, videofilter
- **mixagem** videomixer, clockoverlay, cairooverlay
- **miscelânea** análise: fft, rpgain; buffer, goom (visualização)

Plugins 2

de alto nível

- **decodebin** autodetectação da codificação e autoconfiguração
- `gst-launch filesrc location=slow_dieter.mp3 ! decodebin ! autoaudiosink`
- **uridecodebin** autodetectação da fonte + buffering
- **playbin** faz tudo para tocar uma fonte
- `gst-launch playbin uri=http://docs.gstreamer.com/media/sintel_trailer-480p.webm`

Extensões

- **GStreamer Editing Services** suporte (mas classes) para aplicações de edição de vídeo
- **GStreamer RTSP Server** servidor RTSP
- **QtGStreamer** *Bindings* C++ e integração em Qt

Aplicações que usam o Gstreamer

- **Amarok** tocador de áudio com muitas opções
- **Auditive** tocador de áudio usando Ncurses
- **Sound Converter** conversão de áudio
- **Buzztrax** ambiente de estúdio como FastTracker
- **Freemix** mixagem de vídeo
- **Longomatch** analisador de vídeo
- **Pitivi** Editor de vídeo (não-linear)
- **Arista Transcoder** conversão de vídeo
- **Empathy** *messaging client* com vídeo e *voice-over-ip*
- **Flumotion** servidor de streaming de mídia distribuido
- **Landell** ferramenta para streaming de vídeos e de áudio
- **[...]**

Fontes e links

- **Gstreamer em casa** http://gstreamer.freedesktop.org/
- **Lista de plugins** http://gstreamer.freedesktop.org/documentation/plugins.html
- **Lista de Aplicações que usam GST** http://gstreamer.freedesktop.org/apps
- **Gstreamer an summer of code** http://gstreamer.freedesktop.org/GSOC
- **Fun with Gstreamer pipelines (linux.conf.au)** https://www.youtube.com/watch?v=MCRKfXipAkU
- **Visualização da história do desenvolvimento do GST** https://www.youtube.com/watch?v=VMiMGOg53Ec

Muito obrigado!