NAME

gst−launch−1.0 – build and run a GStreamer pipeline

SYNOPSIS

gst−launch−1.0 *[OPTION...]* PIPELINE−DESCRIPTION

DESCRIPTION

gst−launch−1.0 is a tool that builds and runs basic *GStreamer* pipelines.

In simple form, a PIPELINE−DESCRIPTION is a list of elements separated by exclamation marks (!). Properties may be appended to elements, in the form *property=value*. A "preset" can also be set using the *@preset=<preset name>* syntax.

For a complete description of possible PIPELINE-DESCRIPTIONS see the section *pipeline description* below or consult the GStreamer documentation.

Please note that *gst−launch−1.0* is primarily a debugging tool for developers and users. You should not build applications on top of it. For applications, use the gst_parse_launch() function of the GStreamer API as an easy way to construct pipelines from pipeline descriptions.

OPTIONS

gst−launch−1.0 accepts the following options:

−−help   Print help synopsis and available FLAGS

−v, −−verbose
Output status information and property notifications

−q, −−quiet
Do not print any progress information

−m, −−messages
Output messages posted on the pipeline's bus

−t, −−tags
Output tags (also known as metadata)

−e, −−eos−on−shutdown
Force an EOS event on sources before shutting the pipeline down. This is useful to make sure muxers create readable files when a muxing pipeline is shut down forcefully via Control-C.

−i, −−index
Gather and print index statistics. This is mostly useful for playback or recording pipelines.

−f, −−no−fault
Do not install a fault handler

−T, −−trace
Print memory allocation traces. The feature must be enabled at compile time to work.

−−no−position
Do not print current position of pipeline. If this option is unspecified, the position will be printed when stdout is a TTY. To enable printing position when stdout is not a TTY, use "force-position" option.

−−force−position
Allow printing current position of pipeline even if stdout is not a TTY. This option has no effect if the "no-position" option is specified.

GSTREAMER OPTIONS

gst−launch−1.0 also accepts the following options that are common to all GStreamer applications:

−−gst−version
>   Prints the version string of the *GStreamer* core library.

−−gst−fatal−warnings
>   Causes *GStreamer* to abort if a warning message occurs. This is equivalent to setting the environment variable G_DEBUG to 'fatal_warnings' (see the section *environment variables* below for further information).

−−gst−debug=STRING
>   A comma separated list of category_name:level pairs to specify debugging levels for each category. Level is in the range 0-9 where 0 will show no messages, and 9 will show all messages. The wildcard * can be used to match category names. Note that the order of categories and levels is important, wildcards at the end may override levels set earlier. The log levels are: 1=ERROR, 2=WARNING, 3=FIXME, 4=INFO, 5=DEBUG, 6=LOG, 7=TRACE, 9=MEMDUMP. Since GStreamer 1.2 one can also use the debug level names, e.g. −−gst−debug=*sink:LOG. A full description of the various debug levels can be found in the GStreamer core library API documentation, in the "Running GStreamer Applications" section.
>
>   Use −−gst−debug−help to show category names
>
>   Example: GST_CAT:5,GST_ELEMENT_*:3,oggdemux:5

−−gst−debug−level=LEVEL
>   Sets the threshold for printing debugging messages. A higher level will print more messages. The useful range is 0-9, with the default being 0. Level 6 (LOG level) will show all information that is usually required for debugging purposes. Higher levels are only useful in very specific cases. See above for the full list of levels.

−−gst−debug−no−color
>   *GStreamer* normally prints debugging messages so that the messages are color-coded when printed to a terminal that handles ANSI escape sequences. Using this option causes *GStreamer* to print messages without color. Setting the **GST_DEBUG_NO_COLOR** environment variable will achieve the same thing.

−−gst−debug−color−mode
>   *GStreamer* normally prints debugging messages so that the messages are color-coded when printed to a terminal that handles ANSI escape sequences (on *nix), or uses W32 console API to color the messages printed into a console (on W32). Using this option causes *GStreamer* to print messages without color ('off' or 'disable'), print messages with default colors ('on' or 'auto'), or print messages using ANSI escape sequences for coloring ('unix'). Setting the **GST_DE-BUG_COLOR_MODE** environment variable will achieve the same thing.

−−gst−debug−disable
>   Disables debugging.

−−gst−debug−help
>   Prints a list of available debug categories and their default debugging level.

−−gst−plugin−spew
>   *GStreamer* info flags to set Enable printout of errors while loading *GStreamer* plugins

−−gst−plugin−path=PATH
>   Add directories separated with ':' to the plugin search path

−−gst−plugin−load=PLUGINS
>   Preload plugins specified in a comma-separated list. Another way to specify plugins to preload is to use the environment variable GST_PLUGIN_PATH

PIPELINE DESCRIPTION

A pipeline consists *elements* and *links*. *Elements* can be put into *bins* of different sorts. *Elements*, *links* and *bins* can be specified in a pipeline description in any order.

Elements

ELEMENTTYPE *[PROPERTY1 ...]*

Creates an element of type ELEMENTTYPE and sets the PROPERTIES.

Properties

PROPERTY=VALUE ...

Sets the property to the specified value. You can use **gst−inspect−1.0**(1) to find out about properties and allowed values of different elements.
Enumeration properties can be set by name, nick or value.

Presets

@preset=<preset name> ...

Sets the preset on the element. you can use gst−inspect−1.0(1) to find out what presets are available for a specific element.

Bins

[BINTYPE.] (*[PROPERTY1 ...]* PIPELINE-DESCRIPTION)

Specifies that a bin of type BINTYPE is created and the given properties are set. Every element between the braces is put into the bin. Please note the dot that has to be used after the BINTYPE. You will almost never need this functionality, it is only really useful for applications using the gst_launch_parse() API with 'bin' as bintype. That way it is possible to build partial pipelines instead of a full-fledged top-level pipeline.

Links

[[SRCELEMENT].[PAD1,...]] ! *[[SINKELEMENT].[PAD1,...]]* *[[SRCELEMENT].[PAD1,...]]* ! CAPS ! *[[SINKELEMENT].[PAD1,...]]* *[[SRCELEMENT].[PAD1,...]]* : *[[SINKELEMENT].[PAD1,...]]* *[[SRCELEMENT].[PAD1,...]]* : CAPS : *[[SINKELEMENT].[PAD1,...]]*

Links the element with name SRCELEMENT to the element with name SINKELEMENT, using the caps specified in CAPS as a filter. Names can be set on elements with the name property. If the name is omitted, the element that was specified directly in front of or after the link is used. This works across bins. If a padname is given, the link is done with these pads. If no pad names are given all possibilities are tried and a matching pad is used. If multiple padnames are given, both sides must have the same number of pads specified and multiple links are done in the given order.
So the simplest link is a simple exclamation mark, that links the element to the left of it to the element right of it.
Linking using the : operator attempts to link all possible pads between the elements

Caps

MEDIATYPE *[, PROPERTY[, PROPERTY ...]]] [; CAPS[; CAPS ...]]*

Creates a capability with the given media type and optionally with given properties. The media type can be escaped using " or '. If you want to chain caps, you can add more caps in the same format afterwards.

Properties

NAME=*[(TYPE)]*VALUE
in lists and ranges: *[(TYPE)]*VALUE

Sets the requested property in capabilities. The name is an alphanumeric value and the type can have the following case-insensitive values:
- **i** or **int** for integer values or ranges
- **f** or **float** for float values or ranges
- **b**, **bool** or **boolean** for boolean values
- **s**, **str** or **string** for strings
- **fraction** for fractions (framerate, pixel–aspect–ratio)
- **l** or **list** for lists
If no type was given, the following order is tried: integer, float, boolean, string.
Integer values must be parsable by **strtol**(), floats by **strtod**(). FOURCC values may either be integers or strings. Boolean values are (case insensitive) *yes*, *no*, *true* or *false* and may like strings be escaped with " or '.
Ranges are in this format: [VALUE, VALUE]
Lists use this format: { VALUE *[, VALUE ...]* }

PIPELINE EXAMPLES

The examples below assume that you have the correct plug-ins available. In general, "pulsesink" can be substituted with another audio output plug-in such as "alsasink" or "osxaudiosink" Likewise, "xvimagesink" can be substituted with "ximagesink", "glimagesink", or "osxvideosink". Keep in mind though that different sinks might accept different formats and even the same sink might accept different formats on different machines, so you might need to add converter elements like audioconvert and audioresample (for audio) or videoconvert (for video) in front of the sink to make things work.

Audio playback

Play the mp3 music file "music.mp3" using a libmpg123-based plug-in and output to an Pulseaudio device
    gst–launch–1.0 filesrc location=music.mp3 ! mpegaudioparse ! mpg123audiodec ! audioconvert ! audioresample ! pulsesink

Play an Ogg Vorbis format file
    gst–launch–1.0 filesrc location=music.ogg ! oggdemux ! vorbisdec ! audioconvert ! audioresample ! pulsesink

Play an mp3 file or an http stream using GIO
    gst–launch–1.0 giosrc location=music.mp3 ! mpegaudioparse ! mpg123audiodec ! audioconvert ! pulsesink
    gst–launch–1.0 giosrc location=http://domain.com/music.mp3 ! mpegaudioparse ! mpg123audiodec ! audioconvert ! audioresample ! pulsesink

Use GIO to play an mp3 file located on an SMB server
    gst–launch–1.0 giosrc location=smb://computer/music.mp3 ! mpegaudioparse ! mpg123audiodec ! audioconvert ! audioresample ! pulsesink

Format conversion

Convert an mp3 music file to an Ogg Vorbis file

gst−launch−1.0 filesrc location=music.mp3 ! mpegaudioparse ! mpg123audiodec ! audioconvert ! vorbisenc ! oggmux ! filesink location=music.ogg

Convert to the FLAC format
gst−launch−1.0 filesrc location=music.mp3 ! mpegaudioparse ! mpg123audiodec ! audioconvert ! flacenc ! filesink location=test.flac

Other

Plays a .WAV file that contains raw audio data (PCM).
gst−launch−1.0 filesrc location=music.wav ! wavparse ! audioconvert ! audioresample ! pulsesink

Convert a .WAV file containing raw audio data into an Ogg Vorbis or mp3 file
gst−launch−1.0 filesrc location=music.wav ! wavparse ! audioconvert ! vorbisenc ! oggmux ! filesink location=music.ogg
gst−launch−1.0 filesrc location=music.wav ! wavparse ! audioconvert ! lamemp3enc ! filesink location=music.mp3

Rips all tracks from compact disc and convert them into a single mp3 file
gst−launch−1.0 cdparanoiasrc mode=continuous ! audioconvert ! lamemp3enc ! mpegaudioparse ! id3v2mux ! filesink location=cd.mp3

Rips track 5 from the CD and converts it into a single mp3 file
gst−launch−1.0 cdparanoiasrc track=5 ! audioconvert ! lamemp3enc ! mpegaudioparse ! id3v2mux ! filesink location=track5.mp3

Using **gst−inspect−1.0**(1), it is possible to discover settings like the above for cdparanoiasrc that will tell it to rip the entire cd or only tracks of it. Alternatively, you can use an URI and gst−launch−1.0 will find an element (such as cdparanoia) that supports that protocol for you, e.g.:
gst−launch−1.0 cdda://5 ! lamemp3enc vbr=new vbr−quality=6 ! filesink location=track5.mp3

Records sound from your audio input and encodes it into an ogg file
gst−launch−1.0 pulsesrc ! audioconvert ! vorbisenc ! oggmux ! filesink location=input.ogg

Video

Display only the video portion of an MPEG-1 video file, outputting to an X display window
gst−launch−1.0 filesrc location=JB_FF9_TheGravityOfLove.mpg ! dvddemux ! mpegvideoparse ! mpeg2dec ! xvimagesink

Display the video portion of a .vob file (used on DVDs), outputting to an SDL window
gst−launch−1.0 filesrc location=/flflfj.vob ! dvddemux ! mpegvideoparse ! mpeg2dec ! sdlvideosink

Play both video and audio portions of an MPEG movie
gst−launch−1.0 filesrc location=movie.mpg ! dvddemux name=demuxer demuxer. ! queue ! mpegvideoparse ! mpeg2dec ! sdlvideosink demuxer. ! queue ! mpegaudioparse ! mpg123audiodec ! audioconvert ! audioresample ! pulsesink

Play an AVI movie with an external text subtitle stream
gst−launch−1.0 filesrc location=movie.mpg ! mpegdemux name=demuxer demuxer. ! queue ! mpegvideoparse ! mpeg2dec ! videoconvert ! sdlvideosink demuxer. ! queue ! mpegaudioparse ! mpg123audiodec ! audioconvert ! audioresample ! pulsesink

This example also shows how to refer to specific pads by name if an element (here: textoverlay) has multiple sink or source pads.

gst−launch−1.0 textoverlay name=overlay ! videoconvert ! videoscale ! autovideosink   filesrc location=movie.avi ! decodebin ! videoconvert ! overlay.video_sink   filesrc location=movie.srt ! subparse ! overlay.text_sink

Play an AVI movie with an external text subtitle stream using playbin

gst−launch−1.0 playbin uri=file:///path/to/movie.avi suburi=file:///path/to/movie.srt

Network streaming

Stream video using RTP and network elements.

This command would be run on the transmitter

gst−launch−1.0 v4l2src ! video/x−raw,width=128,height=96,format=UYVY ! videoconvert ! ffenc_h263 ! video/x−h263 ! rtph263ppay pt=96 ! udpsink host=192.168.1.1 port=5000

Use this command on the receiver

gst−launch−1.0 udpsrc port=5000 ! application/x−rtp, clock−rate=90000,payload=96 ! rtph263pdepay queue−delay=0 ! ffdec_h263 ! xvimagesink

Diagnostic

Generate a null stream and ignore it (and print out details).

gst−launch−1.0 −v fakesrc num−buffers=16 ! fakesink

Generate a pure sine tone to test the audio output

gst−launch−1.0 audiotestsrc ! audioconvert ! audioresample ! pulsesink

Generate a familiar test pattern to test the video output

gst−launch−1.0 videotestsrc ! xvimagesink
gst−launch−1.0 videotestsrc ! ximagesink

Automatic linking

You can use the decodebin element to automatically select the right elements to get a working pipeline.

Play any supported audio format

gst−launch−1.0 filesrc location=musicfile ! decodebin ! audioconvert ! audioresample ! pulsesink

Play any supported video format with video and audio output. Threads are used automatically. To make this even easier, you can use the playbin element:

gst−launch−1.0 filesrc location=videofile ! decodebin name=decoder decoder. ! queue ! audioconvert ! audioresample ! pulsesink   decoder. ! videoconvert ! xvimagesink
gst−launch−1.0 playbin uri=file:///home/joe/foo.avi

Filtered connections

These examples show you how to use filtered caps.

Show a test image and use the YUY2 or YV12 video format for this.

gst−launch−1.0 videotestsrc ! ’video/x−raw,format=YUY2;video/x−raw,format=YV12’ ! xvimagesink

Record audio and write it to a .wav file. Force usage of signed 16 to 32 bit samples and a sample rate between 32kHz and 64KHz.

gst−launch−1.0 pulsesrc ! ’audio/x−raw,rate=[32000,64000],format={S16LE,S24LE,S32LE}’ ! wavenc ! filesink location=recording.wav

ENVIRONMENT VARIABLES

GST_DEBUG

Comma-separated list of debug categories and levels (e.g. GST_DEBUG=totem:4,typefind:5). ’*’ is allowed as a wildcard as part of debug category names (e.g. GST_DEBUG=*sink:6,*audio*:6). Since 1.2.0 it is also possible to specify the log level by name (1=ERROR, 2=WARN, 3=FIXME, 4=INFO, 5=DEBUG, 6=LOG, 7=TRACE, 9=MEMDUMP) (e.g. GST_DEBUG=*audio*:LOG)

GST_DEBUG_NO_COLOR

When this environment variable is set, coloured debug output is disabled.

GST_DEBUG_DUMP_DOT_DIR

When set to a filesystem path, store ’dot’ files of pipeline graphs there. These can then later be converted into an image using the ’dot’ utility from the graphviz set of tools, like this: dot foo.dot −Tsvg −o foo.svg (png or jpg are also possible as output format). There is also a utility called ’xdot’ which allows you to view the .dot file directly without converting it first.

When the pipeline changes state through NULL to PLAYING and back to NULL, a dot file is generated on each state change. To write a snapshot of the pipeline state, send a SIGHUP to the process.

GST_REGISTRY

Path of the plugin registry file. Default is ˜/.cache/gstreamer−1.0/registry−CPU.bin where CPU is the machine/cpu type GStreamer was compiled for, e.g. ’i486’, ’i686’, ’x86−64’, ’ppc’, etc. (check the output of "uname −i" and "uname −m" for details).

GST_REGISTRY_UPDATE

Set to "no" to force GStreamer to assume that no plugins have changed, been added or been removed. This will make GStreamer skip the initial check whether a rebuild of the registry cache is required or not. This may be useful in embedded environments where the installed plugins never change. Do not use this option in any other setup.

GST_PLUGIN_PATH

Specifies a list of directories to scan for additional plugins. These take precedence over the system plugins.

GST_PLUGIN_SYSTEM_PATH

Specifies a list of plugins that are always loaded by default. If not set, this defaults to the system-installed path, and the plugins installed in the user’s home directory

GST_DEBUG_FILE

Set this variable to a file path to redirect all GStreamer debug messages to this file. If left unset, debug messages with be output unto the standard error.

ORC_CODE

Useful Orc environment variable. Set ORC_CODE=debug to enable debuggers such as gdb to create useful backtraces from Orc-generated code. Set ORC_CODE=backup or ORC_CODE=emulate if you suspect Orc’s SIMD code generator is producing incorrect code. (Quite a few important GStreamer plugins like videotestsrc, audioconvert or audioresample use Orc).

G_DEBUG

Useful GLib environment variable. Set G_DEBUG=fatal_warnings to make GStreamer programs abort when a critical warning such as an assertion failure occurs. This is useful if you want to find out which part of the code caused that warning to be triggered and under what circumstances. Simply set G_DEBUG as mentioned above and run the program in gdb (or let it core dump). Then get

a stack trace in the usual way.

FILES

˜/.cache/gstreamer−1.0/registry−*.bin

The plugin cache; can be deleted at any time, will be re-created automatically when it does not exist yet or plugins change. Based on XDG_CACHE_DIR, so may be in a different location than the one suggested.

SEE ALSO

gst−inspect−1.0(1), **gst−launch−1.0**(1),

AUTHOR

The GStreamer team at http://gstreamer.freedesktop.org/