詠の GStreamer 勉強ノート

目次

第 1 章
GStreamer とは何か

まずは GStreamer がどんなものなのかをざっくりメモしておくのです。

1.1 マルチメディアライブラリ

GStreamer は動画や音声といったマルチメディアを扱うライブラリである。例えば、動画や音声をパソコン上で再生したり、ネットワーク経由でストリーミング配信したり、エフェクトをかけたり、切り貼りしたりといったことができる。また、単なるソフトウェアではなくプログラムからも使えるライブラリなので、こういった機能を自作のプログラムに組み込むことができる。

1.2 オープンソースソフトウェア

GStreamer はオープンソースソフトウェアである。GStreamer 本体のコードは LGPL (GNU Lesser General Public License) に基づいて提供されている。また、GStreamer 本体から呼び出して使うプラグインも多くは LGPL や GPL (GNU General Public License) といったライセンスに基づいたオープンソースソフトウェアとして配布されている。

GStreamer の開発は有志のチームにより行われており[*1]、コードは freedesktop.org の GitLab に置かれている[*2]。

1.3 マルチプラットフォーム

GStreamer は Linux、Windows、Mac をはじめとした様々なプラットフォームに対応したマルチプラットフォームのライブラリである。

1.4 多言語対応

GStreamer は C 言語で書かれたライブラリである。C 言語ではあるが、GObject というオブジェクトシステムを利用してオブジェクト指向で書かれている。そのおかげで別のオブジェクト指向言語から扱えるようにするライブラリ（バインディング）が作りやすく、Python を始めとした様々な言語のバインディングが存在する。ただし、きちんとメンテナンスが続いているバインディングは少数であることには注意。

[*1] Wikipedia には GStreamer Team と書かれている（2023 年 12 月現在）が、公式サイトにはそういう名前は書かれていないようである。

[*2] `https://gitlab.freedesktop.org/gstreamer/gstreamer`

1.5 拡張が容易

GStreamer は、本体にはあまり機能を入れずプラグインで様々な処理を行う設計になっている。動画や画像を扱う実質的な処理はほぼ全てプラグインで行う。従って、必要なプラグインを導入することによって様々なメディア形式に対応できる上、必要な機能を持った既存のプラグインがない場合には自身でプラグインを書くことも可能である。

と、こんなかんじでいいでしょうか。まあ、こういう輪郭をなぞるよりも百聞は一見に如かずなのです。さっさとインストールして使ってみましょう。

第 3 章
付属 CLI で GStreamer と戯れる

インストールができたのです。しかし、いくら私が天才魔術師とはいえいきなり C のプログラムを書くというのも難ですね。……ん、どうも CLI ツールがついていてそれを使うとプログラムを書かずに GStreamer の基本的な操作ができるみたいなのです。まずはこれで様子を見てみましょうか。

3.1　パスが通っていることの確認

端末を開いて、シェルで以下のように入力する。

```
$ gst-play-1.0
Usage: gst-play-1.0 FILE1|URI1 [FILE2|URI2] [FILE3|URI3] ...

You must provide at least one filename or URI to play.
```

このように使い方が表示されたら良し。だめだったら、パスが通っているか確認する。

3.2　gst-play で動画再生

gst-play というのが、コマンドひとつで動画再生までしてくれる一番簡単なやつなのです。コマンドは gst-play じゃなくて gst-play-1.0 と番号がついているのが不思議ですが、これは GStreamer 1.0 系であることを示すバージョン番号みたいですね。

まず、GStreamer のサイトからサンプルファイルをダウンロードする。

```
# 動画ファイル
$ wget https://gstreamer.freedesktop.org/data/media/sintel_trailer-480p.webm

# 音声ファイル
$ wget https://gstreamer.freedesktop.org/data/media/medium/sugar.ogg
```

このファイルを gst-play-1.0 の引数として渡すと、再生される（動画の場合は別ウインドウが開き、音声はそのまま再生される）。

```
$ gst-play-1.0 sintel_trailer-480p.webm
$ gst-play-1.0 sugar.ogg
```

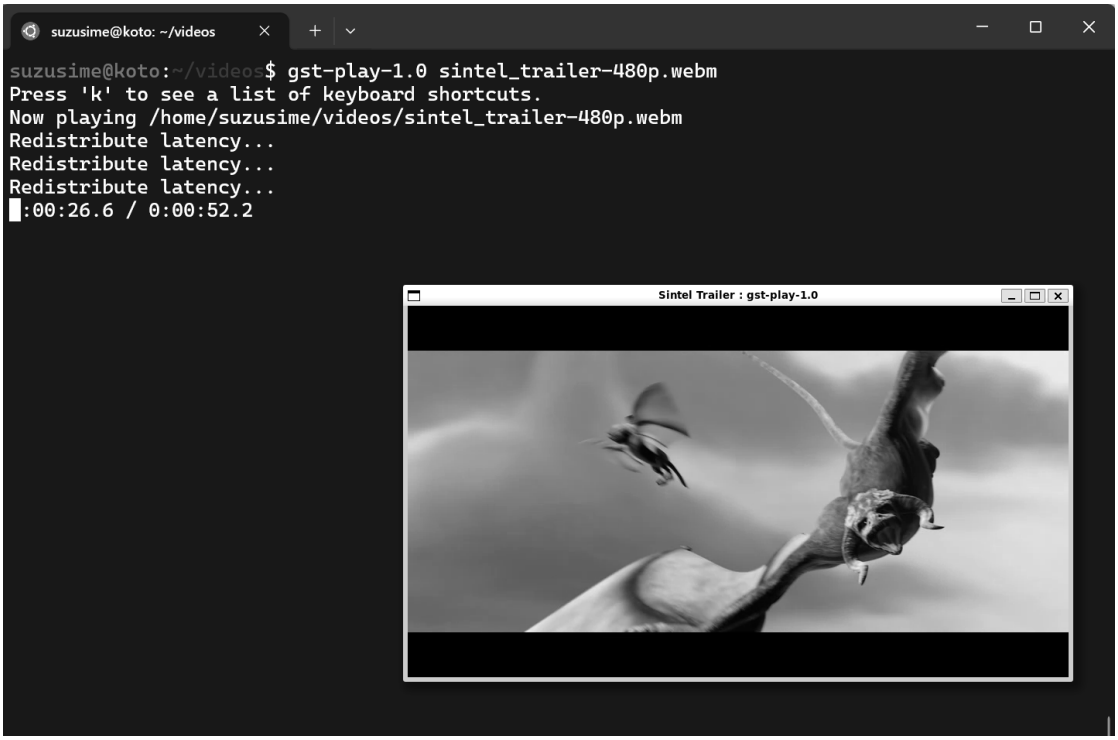

図 3.1　gst-play で動画が再生されている様子

《詠のひとことメモ》

gst-play が動くことの確認は、GStreamer で動画再生できる環境が整っていることを確かめる意味でも大事なのです。ここでは GStreamer のサイトからダウンロードしたファイルで確認しましたが、当然好きな動画ファイルで確認してもよいのですよ。

ただ、「インストール」の章の説明に従って `base-plugins` と `good-plugins` だけしか入れていない状態では WebM や OGV などの限られた形式の動画しか扱えないのです。だから、そういった形式に変換してから読み込ませるのですよ。`ugly-plugins` と `bad-plugins` も入れると再生できる形式が増えるのですが、プラグインが増えすぎて全体像を摑みにくくなるのと、`gst-play` の動作が変わるかもしれないので注意するのです（特に WSL だとウインドウの表示が変わって使いにくくなる[a]ようなのです）。

[a] videosink として `glimagesink` が使われるようになるが WSLg ではこれにうまく対応できていないためと思われる。gst-play 以外であれば videosink に `xvideosink` を指定するようにすれば回避できる。

3.3 gst-launch

gst-play は全部入りでそのまま動画プレイヤーとして使えるコマンドでしたが、gst-launch は部品を組み合わせて望みの動作を実現させるコマンドなのです。この「部品を組み合わせる」というのが GStreamer の基本のようなので、gst-launch で遊んでみるのが構造を理解するのによさそうですね。

3.3.1 playbin3 で動画再生

まずは、playbin3 という部品を使って、動画を再生してみる。以下のようなコマンドを実行すると、gst-play のときと同様に画面が開いて動画が再生される。

```
$ gst-launch-1.0 playbin3 uri=file:///$(pwd)/sintel_trailer-480p.webm
```

playbin3 という部品は、読み込み元を指定するための引数として uri を取るので uri=file:/// (略) を与えている。file:/// の後にはファイルのフルパスを与える必要があるため、pwd コマンドで現在のディレクトリを取得している（フルパスを指定してもいい）。

《詠のひとことメモ》

MSYS2 を使っている場合は、URI を指定するときに file:/// (Windows のフルパス) を指定しないといけないのが注意点なのです。この Windows のフルパスは cygpath というコマンドで取得できるので、上記の代わりに次のようにするといいのですよ。

```
$ gst-launch-1.0 playbin uri=file:///$(cygpath -m $(pwd))/sintel_trailer-480p.webm
```

3.3.2 要素を繋げる

playbin3 はファイルの読み込みから再生まで全て行う部品だったが、そうではなく「ファイルの読み込み」「ファイルの変換」「ファイルの再生」のような単機能を行う部品も存在する。それを使って動画を再生するには以下のようにする（なお、この指定だと音は出ない）。

```
$ gst-launch-1.0 filesrc location=sintel_trailer-480p.webm ! decodebin3 ! videoconvert !
↪  autovideosink
```

これは、filesrc と decodebin3 と videoconvert と autovideosink の 4 つの部品を繋げている。それぞれの部品の間を繋いでいるのは ! である。これは Unix のシェルの |（パイプ）に相当するもの。それぞれの部品は以下のようなことをしている。

- filesrc：ファイルを読み込む。引数 location にファイル名を指定する。
- decodebin3：ファイルを復号する。動画や音声には様々な形式があるが、形式を検出して自動的にいい感じに復号してくれるすごいやつ。
- videoconvert：後ろの autovideosink に入れる動画（の各フレーム）を autovideosink 側が

第 4 章
C 言語で書いてみる

さて、いよいよプログラムを書いていくのです。GStreamer は GObject を使って作られているのでちょっと独特な見た目ですが、深いことは考えずに慣れれば勝ちということでやっていくのです。

《詠のひとことメモ》
この章の内容は MSYS2 や WSL（Ubuntu）で動作確認したのです。残念ながら、Mac では動作しない場合があるようだったのです。

4.1 インストール

C 言語で GStreamer を利用したプログラムを書くためには、GStreamer の実行に必要な共有ライブラリ（Windows でいう dll、Linux でいう so）に加え、ヘッダファイル等が必要になる。開発に必要なファイルは以下によりインストールできる。

4.1.1 Windows や Mac のインストーラー版の場合

development installer でインストールできる。「インストール」の章の説明を参照。

4.1.2 Linux の場合

パッケージマネージャによるが、ヘッダファイルの入った開発用パッケージをインストールする必要がある。Ubuntu の場合は以下の通り。

```
$ sudo apt install libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev
 ↪  libgstreamer-plugins-good1.0-dev
```

4.1.3 MSYS2 の場合

MSYS2 の場合、「インストール」の章でインストールしたパッケージにヘッダファイルも同梱されているため、別途インストールする必要はない。

4.2 ビルドできることの確認

まずは、GStreamer を読み込むだけで何もしないコードでビルドできることの確認を行う。

4.2.1 GCC を使う場合

Linux、Mac、MSYS2 で GStreamer をインストールした場合は、以下のように GCC (GNU Compiler Collection) を利用してプログラムをビルドする。

まず、以下のようなプログラムを `main.c` として保存する。

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

int main(int argc, char** argv) {
    gst_init(&argc, &argv);
    return 0;
}
```

これをビルドする際には、GStreamer のヘッダファイルとライブラリの場所をコマンドラインオプションで指定する必要がある。必要なコマンドラインオプションは、`pkg-config` というコマンドで取得できる（もし `pkg-config` がインストールされていなければ、事前にパッケージマネージャでインストールする）。

```
$ pkg-config --cflags --libs gstreamer-1.0
-pthread -I/usr/include/gstreamer-1.0 -I/usr/include/x86_64-linux-gnu
 ↪  -I/usr/include/glib-2.0 -I/usr/lib/x86_64-linux-gnu/glib-2.0/include -lgstreamer-1.0
 ↪  -lgobject-2.0 -lglib-2.0
```

この出力を GCC のコマンドラインオプションに渡せばよいので、ビルドに必要なコマンドは以下のようになる。

```
$ gcc -Wall -o main main.c $(pkg-config --cflags --libs gstreamer-1.0)
```

このコマンドがエラーメッセージなしに終了し、`main` という実行ファイルが生成されれば成功である。

なお、ソースコードを書く際、エディタの補完機能を利かせるためにはヘッダファイルの場所をエディタの設定ファイルにも書く必要がある。このときにも `pkg-config` は役に立つ。例えば、Visual Studio Code を使うのであれば、以下のようなコマンドで取得したヘッダファイルの場所一覧を `c_cpp_properties.json` の `includePath` に設定するとよい（詳細は各エディタのドキュメントを参照）。

```
$ pkg-config --cflags-only-I gstreamer-1.0 | perl -nE 'while(/-I(\S+)/g){ say "\"$1\",”}'
”/usr/include/gstreamer-1.0”,
”/usr/include/x86_64-linux-gnu”,
”/usr/include/glib-2.0”,
”/usr/lib/x86_64-linux-gnu/glib-2.0/include”,
```

4.2.2　Visual Studio を使う場合

Windows でインストーラー版を使って GStreamer をインストールした場合は、Visual Studio でプログラムをビルドすることになる。ヘッダファイルやライブラリの参照の追加は、pkg-config の代わりに Visual Studio のプロパティシートという機能で行う。この方法については公式ドキュメントの“Installing on Windows”[*1]を参照のこと。

《詠のひとことメモ》
公式ドキュメントでは Visual Studio 2010 で説明しているので不安になるかもしれないですが、最新の Visual Studio 2022 でも同じ方法でちゃんと動くので安心するのです。

4.3　パイプラインを組んでみる

環境構築ができたので、早速パイプラインを組んでいくのです。まずは gst-launch で最初にやったように、playbin3 を使って要素ひとつのパイプラインを作ってみるのです。

以下のようなプログラムを playbin-proto1.c として保存する。

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

int main(int argc, char **argv) {
  // GStreamer の初期化
  gst_init(&argc, &argv);

  // 開くファイルをコマンドラインオプションから取得
  if (argc != 2) {
    g_printerr("Usage: ./playbin-proto1 <URI>\n");
    return -1;
  }
  gchar *uri = argv[1];

  // パイプラインの作成
  GstElement *pipeline = gst_pipeline_new("first-pipeline");
  if (!pipeline) {
    g_printerr("Failed to create pipeline.\n");
    return -1;
  }

  // 要素 playbin3 を作ってパイプラインに追加する
  GstElement *playbin = gst_element_factory_make("playbin3", "play-bin");
  if (!playbin) {
    g_printerr("Failed to create playbin.\n");
```

[*1] https://gstreamer.freedesktop.org/documentation/installing/on-windows.html