Gstreamer : Lecteur Multimedia

1-Utilisation

J'ai créé un lecteur multimédia qui permet la lecture des formats suivants :
audio : mp3, wav
vidéo : avi (xvid mpeg-4 + mp3), ogg (theora + vorbis), mpg (mpeg-1 video, mpeg-2 audio Layer 2)

Le lecteur prend en compte divers paramètres (en options) :

- Les informations sur le fichier source(info)
- Le démarrage à un point donnée de la vidéo(start=secondes)
- Le volume audio et vidéo(volume=valeur)
- La correction gamma de la vidéo(gamma=valeur)
- La luminosité, le contraste et la saturation de la vidéo(respectivement brightness=valeur, contrast= valeur et saturation=valeur)
- L'affichage en ASCII Art de la vidéo(ascii)

Le lecteur s'utilise comme suit :
./nom_du_lecteur video.format info start=8.0 volume=5.0 gamma=1.0 brightness=8.0 contrast=1.0 saturation=0.0 ascii
Pour compiler : gcc lecteur.c flux.c avi.c ogg.c mpg.c mp3.c wav.c tag.c -o lire `pkg-config --cflags --libs gstreamer-0.10`

→ bien-sûr vidéo est soit le nom de la vidéo dans le cas où elle se situe dans le même dossier soit le chemin relatif
→ ici toutes les options sont présentes mais ce n'est pas forcément nécessaires, on met celles qu'on veut et dans l'ordre désiré
→ les valeurs numériques sont décimales pour la précision mais peuvent aussi être des entiers

2- Le programme

1-Général

J'ai décomposé le programme en plusieurs fichiers :
lecteur.c : le fichier principal contenant le main
flux.c: détermine le type d'un flux(audio ou vidéo)
tag.c:permet d'afficher toutes sortes d'informations sur un fichier(encodage utilisé, fréquence échantillonnage...)
mp3.c : construit le pipeline pour lecture d'un mp3(prend en compte les options sur le flux : volume...)
wav.c : construit le pipeline pour lecture d'un wav(prend en compte les options sur le flux : volume...)
avi.c : construit le pipeline pour lecture d'un avi(prend en compte les options sur le flux : volume...)
mpg.c : construit le pipeline pour lecture d'un mpg(prend en compte les options sur le flux : volume...)
ogg.c : construit le pipeline pour lecture d'un ogg(prend en compte les options sur le flux : volume...)
Mis à part le lecteur.c, chaque fichier c à son header afin de pour utiliser chaque fonctions depuis les autres programmes.

Étant donné que l'on construit des pipelines statiques, que l'on ne modifie donc pas, j'ai décidé de

décomposer ainsi le code afin de pouvoir traiter chaque type de pipeline simplement et efficacement.

2-Choix de programmation

→ Chaque traitement de fichier est effectué de manière similaire :

- Définir les éléments : GstElement *pipeline, *source...
- Initialisations : gst_init (&argc, &argv); + options(vol=0 ;...)
- Gestion et initialisation des options (gamma, volume) : for(i=1;i<argc;i++){ if(strstr (argv[i], "volume")!=NULL){vol=...
- Création des éléments gstreamer : pipeline = gst_pipeline_new ("video-player");...
- Configuration des valeurs des éléments : g_object_set (G_OBJECT (source), "location", argv[1], NULL);...
- La gestion de messages :gst_bus_add_watch
- Liaison des éléments : gst_element_link
- Passage a l'etat "playing" du pipeline :gst_element_set_state
- Itération :g_main_loop_run (loop);
- Fin de lecture, on nettoie

Les fonctions on_pad_added et bus_call permette respectivement de connecter le demuxer et de gérer les messages(erreur, fin de lecture...)

→ Pour le type de flux, il fallait un analyseur car l'extension(ex : ,mp3) ne renseigne pas précisément, de plus un fichier nommer d'extension .mp3 pourrait contenir un jpeg par exemple.
J'ai donc utilisé le gst_element typefind :
typefind = gst_element_factory_make ("typefind", "typefinder");(flux.c)
Il suffit de l'insérer dans le pipeline et de l'utiliser avec la fonction cb_typefound pour l'utiliser :
static void cb_typefound (GstElement *typefind, guint probability, GstCaps *caps, gpointer data) (flux.c)

→ pour la gestion des options, j'ai écrit un algorithme qui passe en boucle les paramètres d'entrées(arg[i]) et verifie graçe à la fonction strstr si ces paramètres corespondent à une option existente :

```
for(i=1;i<argc;i++){ de 1 à nombre d'argument en paramètre
        if(strstr (argv[i], "volume")!=NULL){ si volume fait partie des parametres...
vol=(double)atof(getExt(argv[i])+1); ...on initialise avec la valeur donné après le =
      }...
```

Les options sont d'abord toutes initialisées avec leurs valeurs par défaut : double vol=1; volume à 1
Les éléments sont ensuite créé normalement avec gst_element_factory_make et insérer dans le pipeline.
Toutefois deux options sont différentes :

- info : donne les informations du média, si cette option est détecté dans les arguments en entrée, on utilise la fonction tag (tag,c) qui prend un flux en paramètre et affiche ces options
- start : (option vidéo seulement) permet de démarrer la vidéo de l'endroit souhaité. Si cette option est détecté, on initialise une variable avec le temps de départ **en seconde.** La gestion des messages gst_bus_add_watch prend cette fois ci en paramètre le pipeline au lieux de loop car une fois le message GST_STREAM_STATUS_TYPE_ENTER détecté, il va falloir modifier le pipeline grâce à gst_element_seek_simple qui permet de partir de l'endroit souhaité.
  De même lorsque le message GST_MESSAGE_EOS est détecté, on n'utilise plus g_main_loop_quit (loop) pour quitter la vidéo mais gst_element_seek avec une valeur de temps à 0 pour relancer la lecture.