文章编号：1005-8451（2017）11-0041-05

基于GStreamer的音视频播放器的设计

左登超[1]，左登峰[2]，刘永康[1]
（1.中车青岛四方车辆研究所有限公司，青岛 266031；
2.内蒙古银行总行 科技信息部，呼和浩特 010010）

摘 要：目前，铁路客车大多使用Windows系统进行播放器的设计。Windows系统容易感染病毒，并且在铁路客车经常断电复位的情况下，很容易造成系统崩溃。为了解决上述问题，文章提出了在Linux系统下，使用GStreamer框架进行播放器的构建并给出系统运行效果图。经装车验证，在GStreamer框架下构建的音视频播放器的播放效果清晰、稳定，开发周期短，实用性较强。

关键词：Linux；GStreamer；QT；音视频播放器
中图分类号：U285：TP39 **文献标识码：**A

Audio and video player based on GStreamer

ZUO Dengchao[1], ZUO Dengfeng[2], LIU Yongkang[1]
(1. CRRC Qing Dao Sifang Rolling Stock Research Institute, Qingdao 266031, China;
2. Department of Technology Information, Head Office of Bank of Inner Mongolia, Hohhot 010010, China)

Abstract: At present, Windows system was used to design the player for most of the railway passenger cars. But the Windows system is vulnerable to viruses and can easily cause a crash when a passenger car is often powered off and reset. In order to solve these problems, this article proposed the construction of the player using GStreamer framework based on the Linux system, gave the system running effect diagram. The verification results showed that the audio and video player got clear and stable playback effect with short development cycle and strong practicability under the GStreamer framework.

Keywords: Linux; GStreamer; QT; audio and video player

目前，铁路客车上大多使用 Windows 系统进行播放器的设计。在向娱乐系统拷贝节目时，Windows 系统容易感染病毒，并且在铁路客车经常断电复位的情况下，很容易造成系统崩溃。为了解决上述问题，本文提出了在 Linux 系统下，使用 GStreamer 框架进行播放器的构建。使经常断电复位对 Linux 系统不会造成影响，并且 Linux 系统不易感染病毒。GStreamer 框架在设计时采用了非常灵活的体系结构，并提供了各种各样的预定义媒体处理模块，简化了在 Linux 下开发多媒体应用的过程。

1 GStreamer框架

GStreamer 是基于插件的流媒体应用程序开发框架。插件提供了各种各样的多媒体数字信号编解码器。开发者利用 GStreamer 提供的插件、API 接口及数据结构，可以编写一个适用于特定环境下的音视频播放器[1]。文章主要分析 GStreamer 框架构建播放器的基本思想，为后面构建播放器奠定理论基础。

1.1 元件

元件是 GStreamer 框架中最基本的类对象。可以创建一系列的元件并把它们链接起来，使数据流在被连接的各元件之间进行传输，从而创建一个管道来完成一个特殊任务，例如：媒体播放或者录音[2]。每个元件都有一个特殊的函数接口，有些接口可以用来读取文件、解码文件；有些接口则只是输出相应的数据到具体的设备上，例如：声卡设备。GStreamer 默认安装了很多有用的元件，通过使用这些元件，可以构建一个具有多种功能的播放器。

1.2 衬垫

当链接元件时，衬垫就是元件的输入输出，用来在元件之间协商链接和数据流动。一个衬垫可以看作元件上的一个“插口”或者“端口”[3]。通过衬垫，数据可以流出或者从其他元件流入。衬垫有特定

收稿日期：2017-05-20
作者简介：左登超，工程师；左登峰，工程师。

的数据处理能力，一个衬垫能够限制数据流类型的通过。只有在两个衬垫允许通过的数据类型一致时，元件之间的链接才被建立。数据类型使用了一个叫做 caps negotiation 的过程，在衬垫之间进行协商。

数据流建立起链接的元件之间流动。数据向元件以外流出可以通过一个或者多个 source 衬垫，元件接收数据是通过一个或者多个 sink 衬垫来完成的。source 元件（接收数据）和 sink 元件（输出数据）分别有且仅有一个 sink 衬垫或者 source 衬垫。数据在这里代表的是缓冲区 (Buffers) (GstBuffer 对象描述了数据的缓冲区信息) 和事件 (Events) (GstEvent 对象描述了数据的事件信息)[4]。

1.3 箱柜和管道

一个箱柜是一个容器，可以容纳一堆元件。箱柜是元件的子类，可以像控制一个元件一样控制一个箱柜，可以为应用程序屏蔽很多的复杂性 [5]。例如，通过改变某一个箱柜的状态来改变它所包含的所有元件状态。箱柜也可以将来自其所包含的元素的总线消息向前传递。(例如 ：error messages，EOS messages)。

一个 Pipeline 是一个顶层的 Bin，为应用程序提供了一个软件层面上的总线并且为其所包含的元件管理同步 [3]。当将 Pileline 设置为 PAUSED 或者 PLAYING 状态时 , 数据流动将会开始并且媒体处理也将会发生。此时，Pipelines 在一个分离线程中运行直到外界停止此管道的或者管道达到数据流的末尾。

1.4 通信

GStreamer 框架为应用程序和管道之间的通信和数据交换提供了一些机制。

Buffers 是在元件之间传递流的对象，Buffers 从 sources 传到 sinks(顺流)。

Events 是在元件之间或者从应用程序到元件之间传递的对象。Events 可以顺流也可以逆流。顺流 Events 可以同数据流同步。

Messages 是被元件发送到管道消息总线上的对象，在总线上它们将会被应用程序收集。在一个安全线程中，Messages 用来传递元件产生的像 errors, tags，state changes, buffering state, redirects 等的信息到应用程序。

Queries 允许应用程序从管道中请求诸如像 duration 或者 current playback position 的信息。

管道通信流，如图 1 所示。

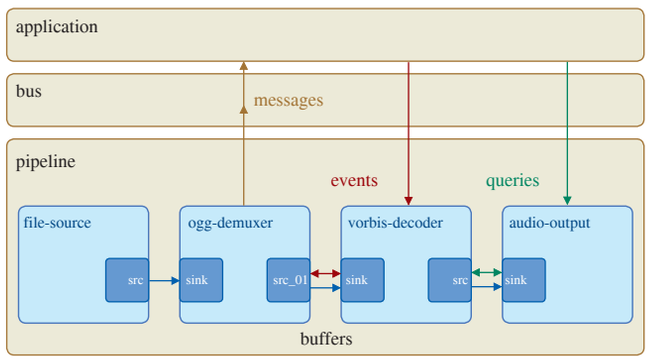

图1 管道通信流

2 基于GStreamer的音视频播放器设计

2.1 硬件系统搭建

本系统硬件平台，CPU 采用 ARM Coretex A9 Dual，1G RAM，1G ROM，运行经裁剪的适应目标板的 Linux 操作系统 ；外围硬件接口包括 SATA、CF 卡接口、USB、RS232、VGA、Audio 接口、电源接口。可以支持外接硬盘、外接 CF 卡、外接 VGA 显示器等，满足基本的音视频播放功能。

2.2 音频播放器设计

2.2.1 操作步骤

在本系统中，音频播放器处理 mp3 音频文件。构建音频播放器的操作步骤如下所示 ：

（1）根据需求定义 GStreamer 框架中的类对象 ：

GstElement*m_Pipeline; // 定义管道元件

GstBus *m_Bus; // 定义总线

GstElement*m_Audiosink; // 定义音频输出元件

GMainLoop m_Loop;// 定义循环

gdouble m_Volume; // 定义音量。

（2）调用 gst_init 方法对 GStreamer 框架进行初始化操作 ：

gst_init_check(NULL, NULL, NULL))。

（3）调用 gst_element_factory_make 方法创建一个管道元件对象 ：

m_Pipeline=gst_element_factory_make("playbin2","player")。

（4）调用 gst_element_factory_make 方法创建一

个音频输出元件对象：

m_Audiosink=gst_element_factory_make("alsa-sink","audiosink")。

（5）调用 gst_pipeline_get_bus 方法，从管道中获取总线对象，并对总线设置消息回调函数：

m_Bus=gst_pipeline_get_bus(GST_PIPELINE(m_Pipeline))。

gst_bus_add_watch(m_Bus,bus_call,&bus_data);

（6）定义（5）中的消息回调函数：

static gboolean bus_call(GstBus *bus,GstMessage *msg,gpointer data)。

在该消息处理函数中，主要捕获 GST_MESSAGE_EOS 消息与 GST_MESSAGE_ERROR 消息。分别处理文件播放完毕与播放时遇到错误的情况。

2.2.2　Qt程序处理流程

将播放器嵌入到 Qt 程序中，通过 Qt 界面，可以对播放器进行播放、暂停、停止、快进、快退、音量调节、播放进度的查询操作。Qt 程序处理流程，如图 2 所示。

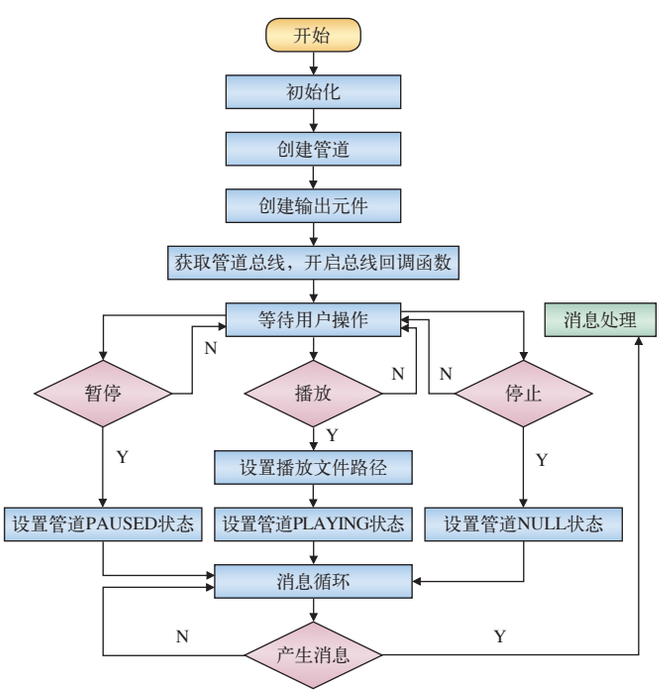

图2　Qt音频播放器的处理流程

（1）播放

a. 调用 gst_element_set_state 对管道设置播放状态。

gst_element_set_state(m_Pipeline, GST_STATE_PLAYING)；

b. 若设置成功，则使播放器进入循环状态，g_main_loop_run(bus_data.m_Loop)。

（2）暂停

调用 gst_element_set_state 对管道设置暂停状态。

gst_element_set_state(m_Pipeline, GST_STATE_PAUSED)。

（3）停止

调用 gst_element_set_state 对管道设置空。

gst_element_set_state(m_Pipeline, GST_STATE_NULL)。

（4）快进

a. 调用 gst_element_query_position 查询管道当前的播放时间；

b. 将当前播放时间加上一定的值；

c.gst_element_seek 接口函数设置指定的播放时间，从而实现快进操作。

（5）快退

a. 调用 gst_element_query_position 查询管道当前的播放时间；

b. 将当前播放时间减去一定的值；

c.gst_element_seek 接口函数设置指定的播放时间，从而实现快退操作。

（6）音量调节

a. 设置当前音量 m_Volume 的值；

b. 调用 g_object_set 接口对播放器设置音量，

g_object_set(m_Pipeline, "volume", m_Volume, NULL)。

（7）时间查询

调用接口 gst_element_query_position 进行播放进度的查询。

gst_element_query_position(m_Pipeline,&fmt,&pos))。

音频播放的效果，如图 3 所示。

2.3　视频播放器设计

2.3.1　操作步骤

在本系统中，视频播放器处理 DVD 文件。构建视频播放器的操作步骤，如下所示：

（1）根据需求定义 GStreamer 框架中的类对象；

图3 音频播放运行效果图

GstElement* m_Pipeline; // 定义一个管道元件

GstElement*m_Audiosink; // 定义音频输出元件

GstElement*m_Videosink; // 定义视频输出元件

GstElement* m_Source; // 定义文件源元件

GstElement* m_Videodecodec;// 定义视频解码器

GstElement* m_Audiodecodec; // 定义音频解码器

GstBus *m_Bus ; // 定义总线

gulong m_WinID ; // 定义播放窗口 ID 号

gdouble m_Volume ; // 定义音量。

（2）调用 gst_init 方法对 GStreamer 进行初始化操作 ；gst_init (NULL,NULL))。

（3）调用 gst_pipeline_new 方法创建管道对象 ：m_Pipeline=gst_pipeline_new("audio-player")。

（4）调用 gst_element_factory_make 接口分别创建文件源元件、音视频解码器、音视频输出元件。

m_Source = gst_element_factory_make("file-src","file-source") ；

m_Videodecodec = gst_element_factory_make ("vpudec", "videodecodec") ；

m_Videosink = gst_element_factory_make ("mfw_v4lsink", "videosink") ；

m_Audiodecodec = gst_element_factory_make ("ac3parse", "audioparsers") ；

m_Audiosink = gst_element_factory_make ("alsasink", "audiosink")。

（5）调用 gst_pipeline_get_bus 方法，从管道中获取总线对象，并对总线设置消息回调函数，

m_Bus=gst_pipeline_get_bus(GST_PIPELINE(m_Pipeline)) ；

gst_bus_add_watch(m_Bus,bus_call,&bus_data);

（6）定义（5）中的消息回调函数。

static gboolean bus_call(GstBus *bus，GstMessage *msg,gpointer data) ；在该消息处理函数中，主要捕获 GST_MESSAGE_EOS 消息与 GST_MESSAGE_ERROR 消息。分别处理文件播放完毕与播放时遇到错误的情况。

2.3.2 Qt程序处理流程

将视频播放器嵌入到 Qt 程序中，通过 Qt 界面，可以对视频播放器进行播放、暂停、停止、快进、快退、音量调节、播放进度的查询操作。Qt 程序处理流程，如图 4 所示。

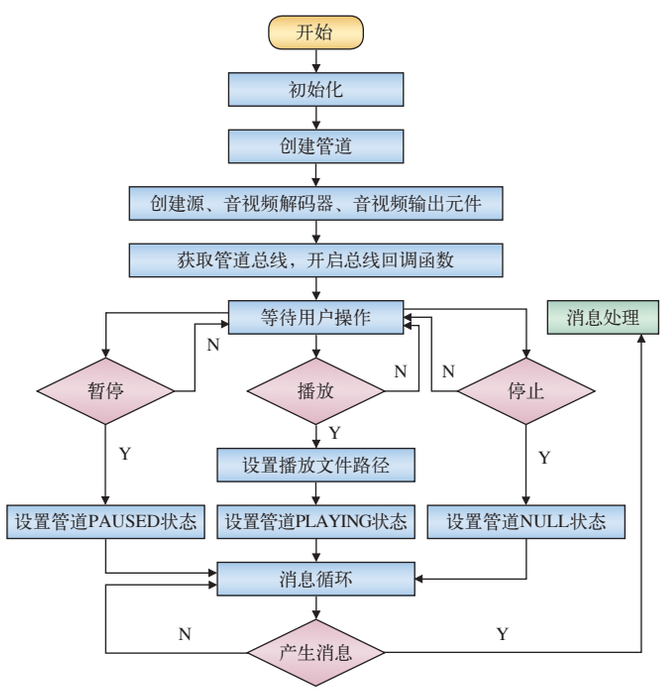

图4 视频播放器的处理流程

（1）播放

调用 gst_element_set_state 对管道设置播放状态，并指定播放窗口 ID。

gst_element_set_state(m_Pipeline, GST_STATE_PLAYING);

gst_x_overlay_set_xwindow_id(GST_X_OVERLAY(m_PlayerVOB->m_Videosink),ui->movie-

（下转 P53）

下的各个 AF 欠缺令牌子队列的富余令牌分配配额，验证了不同网络状态下 AF 子队列的令牌富余欠缺情况以及 AF 业务流的传输特性对富余令牌分配配额的影响。

参考文献：

[1] 钟　廷 . 基于业务的 TCSN 网络调度算法研究 [D]. 成都：西南交通大学，2016.

[2] 裴子秀 . 基于以太网的列车通信网络性能研究 [D]. 成都：西南交通大学，2014.

[3] 刘　辰 . 基于 DiffServ 模型的层次化 QoS 研究 [D]. 杭州：浙江大学，2003.

[4] 李晓利，郭宇春 QoS 技术中令牌桶算法实现方式比较 [J]. 中兴通讯技术，2007，13（3）：56-60.

[5] 张圆圆，郭裕顺 . 流量监管和流量整形的实现和应用 [J]. 中国水运，2003，10（11）：78-80.

[6] 涂文伟，张　进，张兴明 . 分级统筹令牌参数的流量整形算法 [J]. 计算机应用，2006，26（9）：2175-2177.

[7] 韩　冰 . 网络公平性的区分服务分组标记和队列调度策略研究 [D]. 长春：吉林大学，2007.

[8] 胡　啸 . 基于区分服务网络的改进 RIO-C 算法的研究与仿真 [D]. 沈阳：东北大学，2009.

责任编辑　徐侃春

（上接 P44）

widget->winId())；

若设置成功，则使播放器进入循环状态。g_main_loop_run(bus_data.m_Loop)。

（2）暂停

调用 gst_element_set_state 对管道设置暂停状态。

gst_element_set_state(m_Pipeline, GST_STATE_PAUSED)。

（3）停止

调用 gst_element_set_state 对管道设置空。

gst_element_set_state(m_Pipeline, GST_STATE_NULL)。

（4）快进

a. 调用 gst_element_query_position 查询管道当前的播放时间；

b. 将当前播放时间加上一定的值；

c.gst_element_seek 接口函数设置指定的播放时间，从而实现快进操作。

（5）快退

a. 调用 gst_element_query_position 查询管道当前的播放时间；

b. 将当前播放时间减去一定的值；

c.gst_element_seek 接口函数设置指定的播放时间，从而实现快退操作。

（6）音量调节

a. 设置当前音量 m_Volume 的值；

b. 调用 g_object_set 接口对播放器设置音量，

g_object_set(m_Pipeline, "volume", m_Volume, NULL)。

（7）时间查询

调用接口 gst_element_query_position 进行播放进度的查询；

gst_element_query_position(m_Pipeline,&fmt, &pos))。

3　结束语

利用 GStreamer 框架提供的 API 接口以及数据结构，能够快速在目标系统下开发出满足要求的播放器。目前本系统已在铁路客车上装车验证，经试验表明，在 GStreamer 框架下构建的音视频播放器的播放效果清晰、稳定，开发周期短，实用性较强。

参考文献：

[1] 王　蕊 . 基于 GStreamer 的媒体播放研究 [J]. 电子设计工程，2012（3）：28-30.

[2] 孟凡飞 . 基于 GStreamer 的嵌入式流媒体播放器的设计 [J]. 嵌入式系统应用，2010（7）：31-32.

[3] 陈　玲 . 基于 GStreamer 的 MP3 播放器开发 [J]. 电脑编程技巧与维护，2010（8）：110-112.

[4] 秦端振 . Linux 下基于 Gstreamer 的流媒体播放器设计 [J]. 科学大众（科学教育），2014（3）：149-151.

[5] 张海滨 . 嵌入式高清播放器的设计与实现 [J]. 计算机工程与设计，2010，31（13）：3084-3087.

责任编辑　徐侃春