Технологии QNX и ЗОСРВ «Нейтрино» в России

Москва, 19 апреля 2016

«Интеграция мультимедийного контента в функциональное ПО для QNX и ЗОСРВ «Нейтрино»

Игорь Рондарев, ООО «СВД Встраиваемые Системы»

СВД Встраиваемые Системы

SWD Software

Задачи

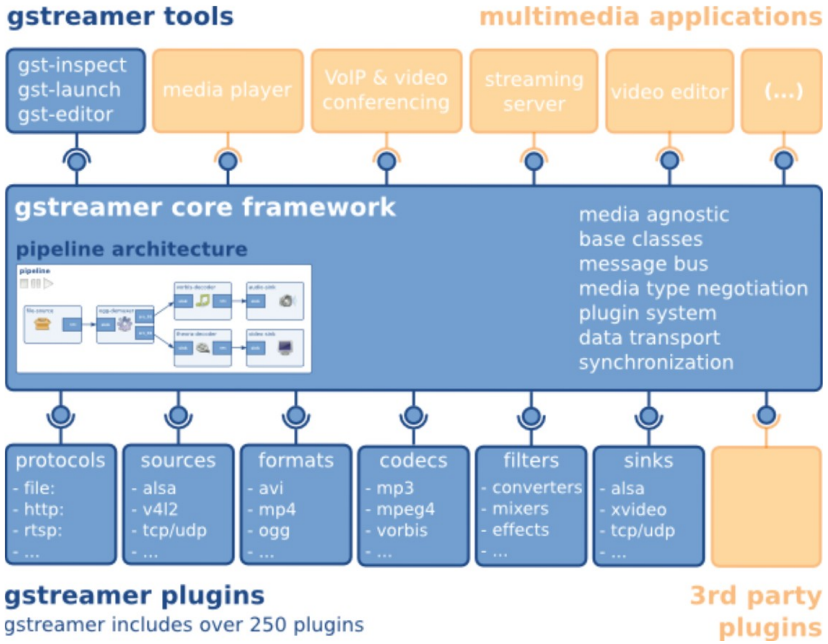
- Интеграция мультимедийного контента в функциональное ПО
  - получение и передача (в т.ч. в потоковом режиме)
  - хранение и обработка (преобразование форматов, фильтрация и т. д.)
  - воспроизведение аудио- и видеоинформации с использованием соответствующих технических средств

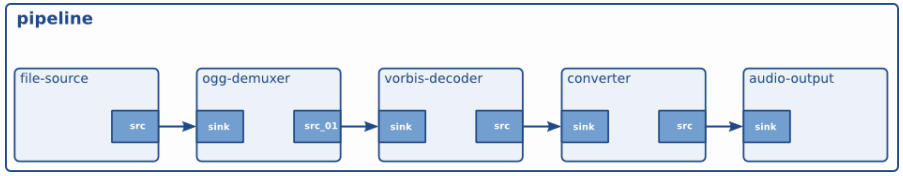
Инструментарий разработчика

- Программные платформы
  - **GStreamer 1.x (**http://gstreamer.freedesktop.org**)**
  - **Qt 4.x/5.x** (http://www.qt.io/)
- Инструментарий разработчика
  - **QNX Momentics IDE (4.7 и 5.0)**
  - **CLI (gcc, gdb и т. д.)**
  - **Qt Creator / Qt Designer**

GStreamer

- Является фундаментальной подсистемой для решения задач, связанных с хранением, обработкой и передачей аудио- и видеоинформации
- Возможности и характеристики:
  - **модульность**
  - большое количество компонентов, имеющихся в составе по умолчанию (**кодеки, фильтры, конвертеры** и т.д.)
  - признание в среде производителей аппаратного обеспечения
  - подробная **документация**
  - отсутствие привязки к какой-либо графической среде
- - основой являются т.н. **конвейеры** (**pipeline**) - цепочки, состоящие из функциональных модулей и их параметров
- Состав: набор библиотек и инструментов





SWD Software

СВД Встраиваемые Системы

Варианты использования GStreamer

Платформа GStreamer позволяет решать широкий круг задач, связанных с мультимедиа-данными (преобразование форматов, потоковая передача и т. д.). При этом возможны различные подходы к его использованию, в т.ч. без применения высокоуровневых API.

- Прототипирование и тестирование
  - консольные утилиты: gst-play, gst-inspect, gst-launch
- Реализация
  - GStreamer C API: gst_init(), gst_pipeline_new() и т.д.
  - Существуют слои адаптации для различных языков программирования (C++, Python, Java и т.д.)

Утилиты GStreamer

- **gst-launch**: построение конвейеров без использования API
  - **Запись аудиоданных:**
    - gst-launch audiotestsrc num-buffers=3 samplesperbuffer=8000 wave=saw ! \
      "audio/x-raw, rate=8000, channels=1" ! wavenc ! filesink location=test.wav
  - **Передача потокового видео (формат H.263+):**
    - gst-launch videotestsrc ! avenc_h263p ! rtph263ppay ! udpsink host=<*ip*> port=<*port*>
  - **Воспроизведение видео в текстовой консоли:**
    - gst-launch videotestsrc ! videoconvert ! decodebin ! Cacasink
- **gst-inspect**: получение информации о компоненте
  - gst-inspect <имя_компонента>

Пример использования gst-launch

- RTP*-передатчик (GStreamer 1.x / QNX 6.5 / ЗОСРВ «Нейтрино»)

  # **gst-launch videotestsrc ! avenc_h263p rtp-payload-size = 20 !**
  rtph263ppay ! udpsink host = 192.168.0.1 port=1234

- RTP-приёмник

  # vlc ./testlocal.sdp

  testlocal.sdp :
  c=IN IP4 192.168.153.1
  m=video 1234 RTP/AVP 96
  a=rtpmap:96 H263-1998/90000

* Real-time Transport Protocol

GStreamer API

- Передача потокового видео:

**gst-launch videotestsrc ! \
avenc_h263p \ ! rtph263ppay ! \
udpsink**

=

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

int main (int argc, char *argv[])
{
  GMainLoop *loop;
  GstElement *pipeline, *source, *encoder, *payload, *sink;

  /* Initialisation */
  gst_init (&argc, &argv);
  loop = g_main_loop_new (NULL, FALSE);

  /* Create gstreamer elements */
  pipeline = gst_pipeline_new ("udp-stream");
  source   = gst_element_factory_make ("videotestsrc",  "test-source");
  encoder  = gst_element_factory_make ("avenc_h263p",   "h263p-encoder");
  payload  = gst_element_factory_make ("rtph263ppay",   "h263p-payload");
  sink     = gst_element_factory_make ("udpsink",       "udp-stream");

  /* Set up the pipeline */

  /* we add all elements into the pipeline */
  gst_bin_add_many (GST_BIN (pipeline),
                    source, encoder, payload, sink, NULL);

  /* we link the elements together */
  gst_element_link_many (source, encoder, payload, sink, NULL);

  /* Set the pipeline to "playing" state*/
  g_print ("Now streaming...\n");
  gst_element_set_state (pipeline, GST_STATE_PLAYING);

  /* Iterate */
  g_print ("Running...\n");
  g_main_loop_run (loop);

  /* Out of the main loop, clean up nicely */
  g_print ("Returned, stopping playback\n");
  gst_element_set_state (pipeline, GST_STATE_NULL);

  g_print ("Deleting pipeline\n");
  gst_object_unref (GST_OBJECT (pipeline));
  g_main_loop_unref (loop);

  return 0;
}
```

Qt 4.x / 5.x для QNX и ЗОСРВ «Нейтрино»

- Qt4 (актуальная версия — **4.8.7**)
  - находится в стабильном состоянии, выпускаются только bugfix-версии
  - функциональность "заморожена", API хорошо документирован
- Qt5 (актуальная версия — **5.5.1**)
  - ветка активно развивается, добавляются новые возможности
  - портирование ПО под Qt5: http://doc.qt.io/qt-5/portingguide.html

Начиная с версии Qt 4.8.6 реализована поддержка функционирования данных платформ в среде **Photon MicroGUI**

Поддержка мультимедиа в Qt (подсистемы)

- **Phonon**
  - Входит в состав Qt4, является опциональной для Qt5.
  - Легковесная, выполняет базовые функции.
- **QtMultimedia**
  - Является основной в Qt5 (частично входит в состав Qt4)
  - В Qt5 дополнена набором прикладных компонентов QtMultimediaWidgets

Поддержка подсистем в ОСРВ QNX и ЗОСРВ «Нейтрино» (апрель 2016)

	Qt4	Qt5
Phonon	Да	Опционально
QtMultimedia	Да	Да*
QtMultimediaWidgets	Нет	Да*

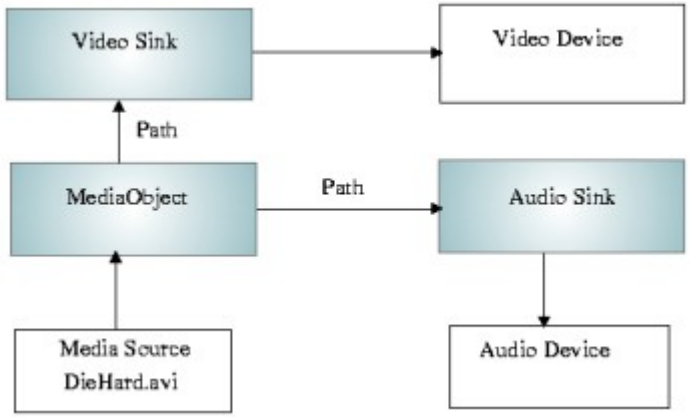
* в процессе тестирования

Phonon

Подсистема платформы Qt, предназначенная для выполнения базовых операций по работе с мультимедиа-контентом.
Входит в состав Qt4 (начиная с версии 4.4), также опционально может быть использована в Qt5.

- Позволяет существенно упростит реализацию следующих типовых операций:
  - получение доступа к контенту (локальному или потоковому)
  - управление процессом воспроизведения контента
  - настройка параметров воспроизведения (громкость, эффекты и т.д.)

- Представляет собой набор классов и методов
  - API: http://doc.qt.io/qt-4.8/phonon-module.html
  - В текущей реализации содержится **около 20 классов**: *Phonon::VideoPlayer, Phonon::AudioOutput, Phonon::VolumeSlider и т.д.*
  - Объекты соединяются между собой путём построения связей, управление осуществляется вызовами соответствующих методов



Использование классов Phonon API

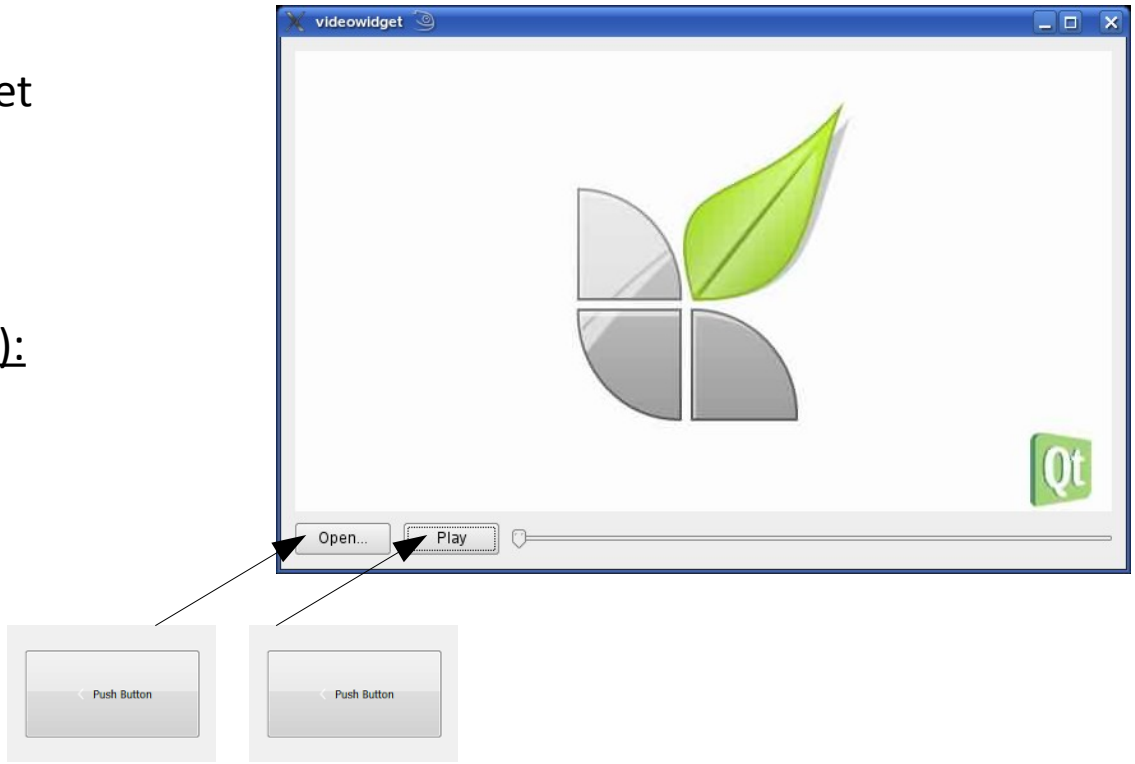
- Базовый способ работы с мультимедиа-данными в пользовательских приложениях, использующих библиотеки Qt 4.x

Разработчик	**Функциональное ПО**
	[Qt Phonon API]
Системное ПО	Phonon library (*libphonon.so*)
	Phonon backend library (*phonon_gstreamer.so*)
	GStreamer Engine (библиотеки)

Использование классов Phonon API (пример)

- Базовый способ работы с мультимедиа-данными в пользовательских приложениях, использующих библиотеки Qt 4.x
- Используемые классы:
  - Phonon::VideoWidget
  - QPushButton
  - QUrl
- qmake Project File (*.pro):

```
QT += phonon
target.path = /tmp
INSTALLS += target
```



Qt Multimedia

Набор компонентов и API, появившийся в Qt 4.x и получивший дальнейшее развитие в Qt 5.x и и позволяющий выполнять операции управления, преобразования и отображения мультимедиа-данных:

API	Назначение
QtMultimedia	Низкоуровневый программный интерфейс
QtMultimediaWidgets	Набор графических компонентов (виджетов)

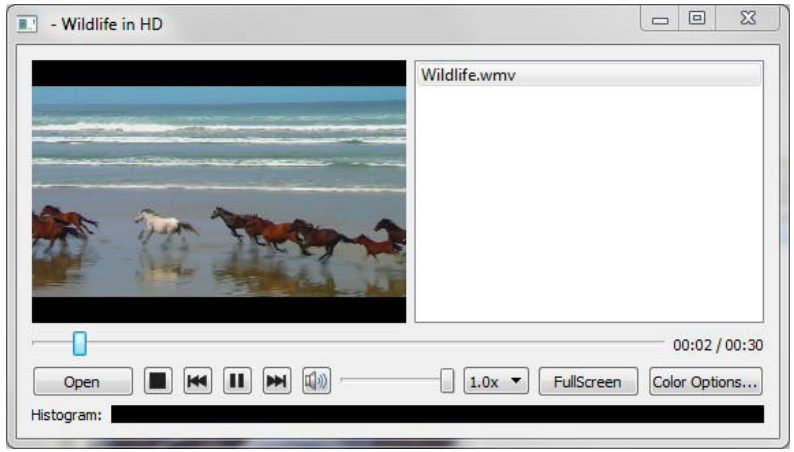
Пример использования в приложениях (фрагмент):

```
playlist = new QMediaPlaylist;
playlist->addMedia(QUrl(""));
playlist->addMedia(QUrl("http://example.com/movie2.mp4"));
playlist->setCurrentIndex(1);

player = new QMediaPlayer;
player->setPlaylist(playlist);

videoWidget = new QVideoWidget;
player->setVideoOutput(videoWidget);
videoWidget->show();

player->play();
```



SWD Software

СВД Встраиваемые Системы

Использование классов QtMultimedia API

- Основной способ работы с мультимедиа-данными в пользовательских приложениях, использующих библиотеки Qt 5.x

Разработчик	**Функциональное ПО**
	[QtMultimedia/QtMultimediaWidgets API]
Системное ПО	QtMultimedia libraries (*libQtMultimedia*.so*)
	GStreamer support plugins (libgst*.*so*)
	GStreamer Engine (библиотеки)

Способы интеграции мультимедийного контента в функциональное ПО

- Использование GStreamer API и консольных утилит
- Использование классов Phonon API
- Использование классов QtMultimedia API

Отладочный вывод

Переменные окружения, позволяющие управлять выводом отладочной информации:

- **GStreamer** (http://gstreamer.freedesktop.org/data/doc/gstreamer/head/gstreamer/html/gst-running.html)
  - GST_DEBUG - низкоуровневый отладочный вывод
- **Qt**
  - QT_DEBUG_PLUGINS - отладочный вывод подсистемы плагинов Qt
- **Phonon** (https://community.kde.org/Phonon/Environment_Variables)
  - PHONON_DEBUG – общая информация
  - PHONON_BACKEND_DEBUG – отладочный вывод backend-плагинов Phonon
  - PHONON_SUBSYSTEM_DEBUG – низкоуровневый отладочный вывод подсистем (в т.ч. Gstreamer)

Спасибо за внимание

Игорь Рондарев
Инженер-программист

+7 (812) 346-89-56
support@kpda.ru

www.kpda.ru
www.swd.ru

SWD Software

СВД Встраиваемые Системы