Compiling Phonon on Linux

by Prashanth N Udupa (prashanth.udupa@gmail.com)

One of the most wonderful news items in the recent times was that Trolltech has opened up its Phonon backends to the KDE community.

> **Oslo, Norway and Tuebingen, Germany, 13 December - 2007 - Trolltech®, the leading cross-platform software development company, and KDE e.V. today announced a technical collaboration on the development of Phonon, a cross-platform multimedia framework that makes it simple for programmers of all experience levels to incorporate multimedia functionality into their applications.**

I was smitten by Phonon ever since I heard about it in Kevin's talk at FOSS.IN 2007. Phonon is the new multimedia engine in KDE that brings video and sound capabilities to KDE and Qt applications. Trolltech has decided to bundle Phonon with Qt 4.4; which means that native Qt applications on Windows and Mac can also make use of it to provide audio and video functionality.

In this article I am going to explain how you can

1. Download Phonon and the relevant backends from the KDE SVN
2. Compiling Phonon on Linux (against Qt 4.3.x and not all of KDE 4).
3. Compiling the Phonon_GStreamer backend (against t 4.3.x and not all of KDE 4).
4. Writing simple applications in Qt that make use of Phonon.

Ok, so here we go.

Setting up your computer for compiling Phonon

To compile Phonon you need the following packages

1. Qt 4.3.1 or later.
2. Any recent compiler (GCC 4.1.2 or later).

In the rest of this article I will assume that you have downloaded Qt 4.3.x sources and compiled and installed it on your computer. This means that the Qt libraries are available in /usr/local/Trolltech/Qt-4.3.x/ directory.

You would ofcourse need a working Internet connection ;)

Compiling Phonon

Phonon is essentially a multimedia framework. Like all frameworks it is made of a core module that establishes the framework (as in the rules and regulations for the module) and a bunch of backend modules that actually implement the framework for a specific platform. In this section we are only compiling Phonon-the-framework and not its backend for Linux.

Checking out Phonon sources

You can checkout the source code for Phonon as a anonymous user from anonsvn.kde.org. Start

konsole, create a folder called Phonon in your home directory and move into that directory. Now execute the following command.

```
[user@localhost]# svn co svn://anonsvn.kde.org/home/kde/trunk/KDE/kdelibs/phonon/
```

This will checkout the core Phonon libraries into /home/user/Phonon/phonon directory.

Creating phonon.pro for use with QMake.

KDE uses CMake as its official build system. While CMake is a great build system generator tool; I prefer to explain the use of QMake in this article because my focus is to get you to use Phonon with just Qt. So lets compile Phonon in Qt style :).

To understand the library dependencies and include paths for Phonon take a look at the CMakeLists.txt file in /home/user/Phonon/phonon.

```
FIND_PACKAGE(Alsa)
ALSA_CONFIGURE_FILE(${CMAKE_CURRENT_BINARY_DIR}/config-alsa.h)

# Those apply to all subdirs
include_directories(${KDE4_KDECORE_INCLUDES} ${CMAKE_BINARY_DIR}/solid)

add_subdirectory(tests)
add_subdirectory(examples)
add_subdirectory(experimental)
add_subdirectory(platform_kde)
add_subdirectory(libkaudiodevicelist)
if(NOT MINGW)
  # ICE in outputdevicechoice.cpp:262
  add_subdirectory( kcm )
endif(NOT MINGW)
########### next target ###############

set(phonon_LIB_SRCS
       objectdescription.cpp
       ..............
       iodevicestream.cpp
       )

kde4_add_library(phonon SHARED ${phonon_LIB_SRCS})
target_link_libraries(phonon ${QT_QTDBUS_LIBRARY} ${QT_QTCORE_LIBRARY} ${QT_QTGUI_LIBRARY})
set_target_properties(phonon       PROPERTIES       VERSION       ${GENERIC_LIB_VERSION}       SOVERSION
${GENERIC_LIB_SOVERSION} )
install(TARGETS phonon  DESTINATION ${LIB_INSTALL_DIR})

########### install files ###############

install( FILES
       phonon_export.h
       ..........
       DESTINATION ${INCLUDE_INSTALL_DIR}/phonon)
install( FILES  phononbackend.desktop  DESTINATION  ${SERVICETYPES_INSTALL_DIR} )
install( FILES org.kde.Phonon.AudioOutput.xml DESTINATION ${DBUS_INTERFACES_INSTALL_DIR} )
```

I have compressed the file list on purpose. Anyways; you will notice that the include file dependencies for Phonon are ${KDE4_KDECORE_INCLUDES} and ${CMAKE_BINARY_DIR}/solid. Lets drop these include file dependencies for now because I assume that you dont have KDE4 compiled on your computer.

Also notice that the library dependencies for Phonon is QtDBus, QtCore and QtGui modules. This is good news because we dont have to worry about KDE libs for compiling Phonon.

Taking the above information; lets now create a phonon.pro file that captures the build information for phonon and save it in /home/user/Phonon/phonon.

```
TEMPLATE = lib
CONFIG += shared qdbus
TARGET = phonon

INCLUDEPATH += . ../

SOURCES +=  objectdescription.cpp \
            objectdescriptionmodel.cpp \
            phononnamespace.cpp \
            mediasource.cpp \
            abstractmediastream.cpp \
            streaminterface.cpp \
            mediaobject.cpp \
            medianode.cpp \
            path.cpp \
            effectparameter.cpp \
            effect.cpp \
            volumefadereffect.cpp \
            audiooutputadaptor.cpp \
            abstractaudiooutput.cpp \
            abstractaudiooutput_p.cpp \
            audiooutput.cpp \
            abstractvideooutput.cpp \
            abstractvideooutput_p.cpp \
            backendcapabilities.cpp \
            globalconfig.cpp \
            factory.cpp \
            platform.cpp \
            mediacontroller.cpp \
            videowidget.cpp \
            videoplayer.cpp \
            seekslider.cpp \
            volumeslider.cpp \
            effectwidget.cpp \
            iodevicestream.cpp

HEADERS +=  abstractaudiooutput.h \
            abstractmediastream.h \
            audiooutputadaptor.h \
            audiooutput.h \
            backendcapabilities.h \
            backendcapabilities_p.h \
            effect.h \
            effectwidget.h \
            factory.h \
            factory_p.h \
            globalconfig.h \
            iodevicestream.h \
            mediacontroller.h \
            mediaobject.h \
            objectdescriptionmodel.h \
            seekslider.h \
            videoplayer.h \
            videowidget.h \
            volumefadereffect.h \
            volumeslider.h
```

You might be wondering how I came up with the HEADERS list. QMake uses the HEADERS file list to figure out the header files for which MOC needs to be run. Any Qt developer would know that MOC needs to be run for header files that contain QObject subclasses that declare the Q_OBJECT macro. So I used a simple grep command to figure out the classes that need MOCing and included the files that declare them in HEADERS.

```
[user@localhost]# grep "Q_OBJECT" *
abstractaudiooutput.h:        Q_OBJECT
abstractmediastream.h:    Q_OBJECT
audiooutputadaptor.h:    Q_OBJECT
audiooutput.h:        Q_OBJECT
backendcapabilities.h:        Q_OBJECT
backendcapabilities_p.h:    Q_OBJECT
```

```
effect.h:        Q_OBJECT
effectwidget.h:         Q_OBJECT
factory.h:         Q_OBJECT
factory_p.h:    Q_OBJECT
globalconfig.h:         Q_OBJECT
iodevicestream.h:    Q_OBJECT
mediacontroller.h:    Q_OBJECT
mediaobject.h:         Q_OBJECT
objectdescriptionmodel.h:            Q_OBJECT_CHECK
seekslider.h:    Q_OBJECT
videoplayer.h:    Q_OBJECT
videowidget.h:         Q_OBJECT
volumefadereffect.h:        Q_OBJECT
volumeslider.h:    Q_OBJECT
```

Compiling Phonon

Now that the .pro file is created; start konsole and move into /home/user/Phonon/phonon. Now run the following commands.

```
[user@localhost]# /usr/local/Trolltech/Qt-4.3.1/bin/qmake
[user@localhost]# make
```

After make returns, you will get the phonon shared libraries in the current working directory.

```
[user@localhost]#  ls *.so*
libphonon.so  libphonon.so.1  libphonon.so.1.0  libphonon.so.1.0.0
```

Thats it :). You have the Phonon framework compiled and ready for use on your computer. Ofcourse you cant do much with the framework unless you want to implement a backend. To make use of Phonon to actually play audio and video files you will need to compile an appropriate backend for Phonon.

Compiling the GStreamer backend for Phonon.

Trolltech recently committed the code for Phonon backends on Windows, Mac and Linux to the KDE repository. These backends can be found in http://websvn.kde.org/trunk/KDE/kdebase/runtime/phonon.

- On Windows you can make use of the DirectShow9 backend.
- On Mac you can make use of the QuickTime backend.
- On Linux you can make use of the GStreamer backend.

In this section I will explain how you can compile the GStreamer backend for Phonon; and in a next section I will explain how you can use Phonon (with the GStreamer backend) to write audio and video applications.

Setting up your computer for compiling GStreamer Phonon backend

To compile the GStreamer Phonon backend; you will ofcourse need GStreamer and its development headers. Towards this I suggest you install gstreamer010, gstreamer010-devel, gstreamer010-plugins-base and gstreamer010-plugins-base-devel packages. While installing these packages you may run into some dependency issues which you would have to satisfy. But setting all of this up is not a painful process thanks to the exceptionally solid package management systems we have these days on practically all distributions of Linux.

Checking out source code of Phonon backends

Create a directory called backends in the /home/user/Phonon directory. Start konsole and move into the newly created backends directory.

> *I must mention at this point that the the phonon framework checkout creates a backend directory in phonon which is where one might want all the backends to go, but I prefer to checkout the phonon backends into a separate directory.*

Like I mentioned in the previous paragraph, Phonon now has three backends (atleast). You can check out all those backends by executing this command.

```
[user@localhost]# svn co svn://anonsvn.kde.org/home/kde/trunk/KDE/kdebase/runtime/phonon/
```

After the checkout is complete you will notice the following contents in the backend directory.

```
[user@localhost]# ls
CMakeLists.txt  ds9/  gstreamer/  org.kde.Phonon.ServiceRegistry.xml  qt7/  serviceregistry.h  xine/
```

Of all the backends that were checked out; we are interested in the gstreamer backend.

Creating gstreamer.pro for use with QMake

Just like we created phonon.pro to compile the Phonon framework; we will need to create a gstreamer.pro in /home/user/Phonon/phonon/backends/gstreamer directory to compile the GStreamer backend for Phonon. The reasons and procedure for doing so remain the same.

Shown below is the gstreamer.pro that I created for compiling the GStreamer backend. *Please replace "user" with your home directory name*.

```
TEMPLATE = lib
CONFIG += shared plugin
QT     += opengl
TARGET = phonon_gstreamer

# Phonon Include path
INCLUDEPATH += /home/user/Phonon/ \
               /home/user/Phonon/phonon

# GStreamer include paths. How did I come to know of this you ask?
# pkg-config --cflags  gstreamer-0.10
INCLUDEPATH += /usr/include/libxml2 \
               /opt/gnome/include/gstreamer-0.10 \
               /opt/gnome/include/glib-2.0 \
               /opt/gnome/lib/glib-2.0/include

# Link information for linking against Phonon.
LIBS        += -lphonon -L/home/user/Phonon/phonon

# Link information for linking against GStreamer.
# How did I come to know of this you ask?
# pkg-config --libs gstreamer-0.10
LIBS        += -L/opt/gnome/lib -lgstreamer-0.10 -lgstbase-0.10 \
               -lgstcontroller-0.10 -lgstnet-0.10 -lgstcheck-0.10 \
               -lgstdataprotocol-0.10 -lgstaudio-0.10 -lgstvideo-0.10 \
               -lgstinterfaces-0.10 \
               -lgobject-2.0 -lgmodule-2.0 -ldl \
               -lgthread-2.0 -lxml2 -lz -lm -lglib-2.0

# Aggregation of sources from the file
# /home/user/Phonon/backends/gstreamer/CMakeLists.txt
SOURCES +=  audiooutput.cpp \
            backend.cpp \
            devicemanager.cpp \
            effectmanager.cpp \
            gsthelper.cpp \
            mediaobject.cpp \
```

```
            medianode.cpp \
            medianodeevent.cpp \
            abstractvideowidget.cpp \
            videowidget.cpp \
            qwidgetvideosink.cpp \
            message.cpp \
            audioeffect.cpp \
            xvideowidget.cpp

# Listing of header files that should be MOCed.
HEADERS +=  abstractvideowidget.h \
            audioeffect.h \
            audiooutput.h \
            backend.h \
            devicemanager.h \
            effectmanager.h \
            mediaobject.h \
            videowidget.h
```

Compiling the GStreamer backend

Now that the .pro file is created; start konsole and move into /home/user/Phonon/backends/gstreamer. Now run the following commands.

```
[user@localhost]# /usr/local/Trolltech/Qt-4.3.1/bin/qmake
[user@localhost]# make
```

After make returns, you will get the phonon shared libraries in the current working directory.

```
[user@localhost]#  ls *.so
libphonon_gstreamer.so
```

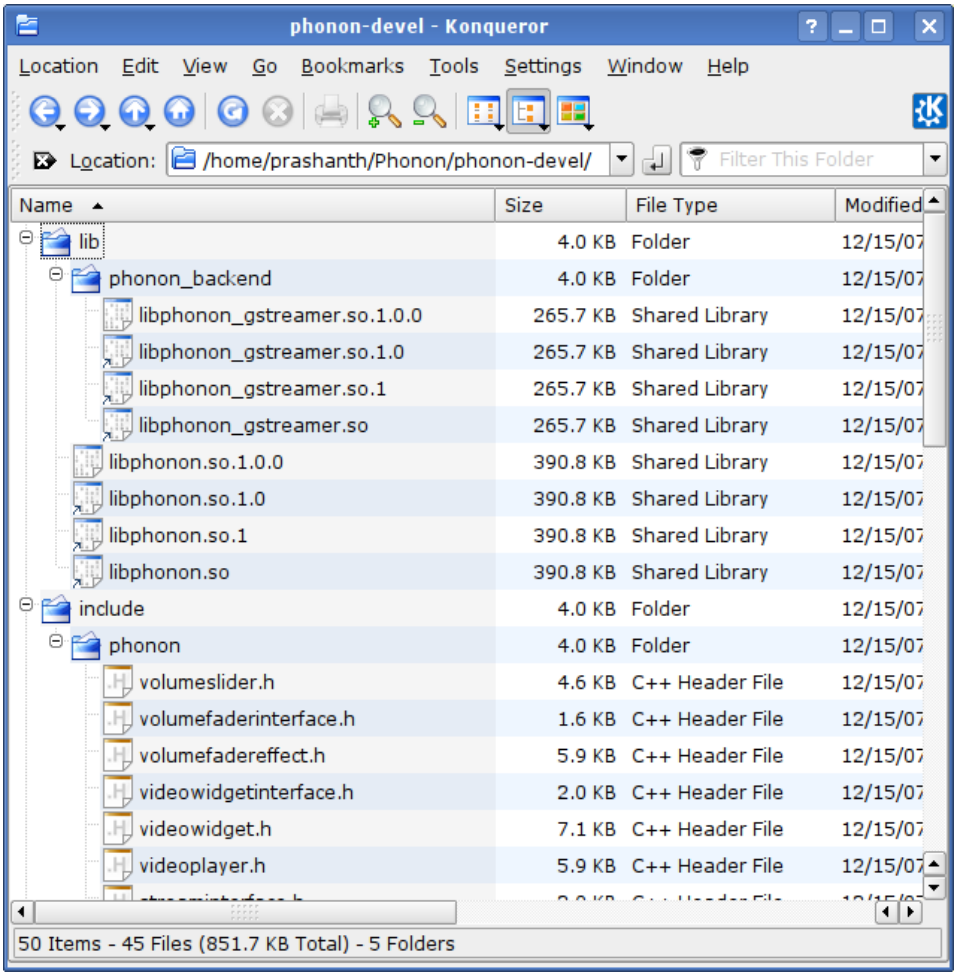
Thats it :). You now have the Phonon framework and the GStreamer compiled and ready for use on your computer.

Preparing a Phonon SDK

So far we have compiled Phonon and its GStreamer backend. To be able to make use of Phonon in your Qt/KDE applications you might want to keep it packaged as a SDK. This basically means that you bring all the required Phonon header files and libraries into a single folder for easy use. Towards this

- create a phonon-devel directory in /home/user/Phonon
- create a include, include/phonon and lib directories within /home/user/Phonon/phonon-devel
- copy the header files (minus the *_p.h files) from /home/user/Phonon/phonon-devel/include/phonon directory into /home/user/Phonon/include/phonon directory.
- copy the phonon and phonon_gstreamer libraries into /home/user/Phonon/phonon-devel/lib and /home/user/Phonon/phonon-devel/lib/phonon_backend directory respectively

The following screenshot is taken from my phonon-devel directory.



The contents of your phonon-devel directory should look like this.

```
/home/user/Phonon/phonon-devel/lib
/home/user/Phonon/phonon-devel/lib/libphonon.so
/home/user/Phonon/phonon-devel/lib/libphonon_gstreamer.so.1
/home/user/Phonon/phonon-devel/lib/libphonon_gstreamer.so.1.0.0
/home/user/Phonon/phonon-devel/lib/libphonon.so.1.0
/home/user/Phonon/phonon-devel/lib/phonon_backend/libphonon_gstreamer.so
/home/user/Phonon/phonon-devel/lib/phonon_backend/libphonon.so.1
/home/user/Phonon/phonon-devel/lib/phonon_backend/libphonon.so.1.0.0
/home/user/Phonon/phonon-devel/lib/phonon_backend/libphonon_gstreamer.so.1.0
/home/user/Phonon/phonon-devel/include
/home/user/Phonon/phonon-devel/include/phonon
/home/user/Phonon/phonon-devel/include/phonon/backendinterface.h
/home/user/Phonon/phonon-devel/include/phonon/videoplayer.h
/home/user/Phonon/phonon-devel/include/phonon/abstractmediastream.h
/home/user/Phonon/phonon-devel/include/phonon/volumefaderinterface.h
/home/user/Phonon/phonon-devel/include/phonon/audiooutputinterface.h
/home/user/Phonon/phonon-devel/include/phonon/effect.h
/home/user/Phonon/phonon-devel/include/phonon/factory.h
/home/user/Phonon/phonon-devel/include/phonon/videowidgetinterface.h
/home/user/Phonon/phonon-devel/include/phonon/effectinterface.h
/home/user/Phonon/phonon-devel/include/phonon/audiooutput.h
/home/user/Phonon/phonon-devel/include/phonon/platformplugin.h
/home/user/Phonon/phonon-devel/include/phonon/addoninterface.h
/home/user/Phonon/phonon-devel/include/phonon/mediaobject.h
/home/user/Phonon/phonon-devel/include/phonon/seekslider.h
/home/user/Phonon/phonon-devel/include/phonon/effectwidget.h
/home/user/Phonon/phonon-devel/include/phonon/objectdescriptionmodel.h
/home/user/Phonon/phonon-devel/include/phonon/effectparameter.h
/home/user/Phonon/phonon-devel/include/phonon/phonondefs.h
/home/user/Phonon/phonon-devel/include/phonon/iodevicestream.h
/home/user/Phonon/phonon-devel/include/phonon/abstractaudiooutput.h
```

```
/home/user/Phonon/phonon-devel/include/phonon/audiooutputadaptor.h
/home/user/Phonon/phonon-devel/include/phonon/volumefadereffect.h
/home/user/Phonon/phonon-devel/include/phonon/objectdescription.h
/home/user/Phonon/phonon-devel/include/phonon/volumeslider.h
/home/user/Phonon/phonon-devel/include/phonon/mediasource.h
/home/user/Phonon/phonon-devel/include/phonon/medianodedestructionhandler.h
/home/user/Phonon/phonon-devel/include/phonon/backendcapabilities.h
/home/user/Phonon/phonon-devel/include/phonon/phonon_export.h
/home/user/Phonon/phonon-devel/include/phonon/streaminterface.h
/home/user/Phonon/phonon-devel/include/phonon/mediacontroller.h
/home/user/Phonon/phonon-devel/include/phonon/path.h
/home/user/Phonon/phonon-devel/include/phonon/videowidget.h
/home/user/Phonon/phonon-devel/include/phonon/medianode.h
/home/user/Phonon/phonon-devel/include/phonon/abstractvideooutput.h
/home/user/Phonon/phonon-devel/include/phonon/globalconfig.h
/home/user/Phonon/phonon-devel/include/phonon/phononnamespace.h
/home/user/Phonon/phonon-devel/include/phonon/mediaobjectinterface.h
```

Cool :). Now you are all set and ready to write simple Qt programs that make use of Phonon.

Using Phonon to play Audio and Video files

Lets now make use of the Phonon framework with its GStreamer backend and write some simple audio and video player programs. The programs that I explain in this article are purely meant to demonstrate how Phonon can be used and not for anything else. To be honest; this is my second day with Phonon so I dont know much about it myself.

> Before you continue you might want to install gstreamer010-plugins-good and gstreamer010-plugins-good-extras packages.

Simple audio player program

Shown below is the listing of a simple audio player program. I really hate it when I read materials where there is a lot of fluff about the architecture of something before they actually show some materials. I usually prefer to see the code first and then figure out the architecture. I hope you like that approach too.

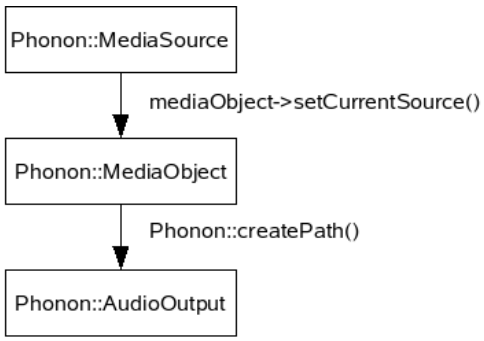
```
#include <QApplication>
#include <QWidget>
#include <QFileInfo>

#include <phonon/mediasource.h>
#include <phonon/mediaobject.h>
#include <phonon/audiooutput.h>

int main(int argc, char** argv)
{
    if(argc != 2)
    {
        qWarning("Usage:\n  %s <input-audio-file-name>", argv[0]);
        return -1;
    }

    static QStringList supportedTypes = QStringList() << "ogg" << "mp3" << "wav";

    QFileInfo fi(argv[1]);
    if( !supportedTypes.contains(fi.suffix()) )
    {
        qWarning("Media type %s not supported by this program yet :(",
                 qPrintable(fi.suffix()));
        return -1;
    }
```

```
    QApplication a(argc, argv);
    a.setApplicationName("Phonon-Audio-Player");

    // The following two lines are not needed if phonon_gstreamer is available in
    // a phonon_backend directory in $QTDIR/plugins
    QCoreApplication::addLibraryPath("/home/user/Phonon/phonon-devel/lib");
    QCoreApplication::addLibraryPath("/home/user/Phonon/phonon-devel/lib/phonon_backend");


    Phonon::MediaSource* source = new Phonon::MediaSource( argv[1] );
    Phonon::MediaObject* mediaObject = new Phonon::MediaObject( 0 );
    Phonon::AudioOutput *audioOutput = new Phonon::AudioOutput(Phonon::MusicCategory, 0);
    Phonon::createPath(mediaObject, audioOutput);

    mediaObject->setCurrentSource(*source);
    mediaObject->play();

    QObject::connect(mediaObject, SIGNAL(finished()), &a, SLOT(quit()));

    int ret = a.exec();

    delete source;
    delete mediaObject;
    delete audioOutput;

    return ret;
}
```

Basically what we did was to create a “pipeline” of sorts in Phonon to load a media file and send the output to an audio output device. The core code is in the following lines

```
    Phonon::MediaSource* source = new Phonon::MediaSource( argv[1] );
    Phonon::MediaObject* mediaObject = new Phonon::MediaObject( 0 );
    Phonon::AudioOutput *audioOutput = new Phonon::AudioOutput(Phonon::MusicCategory, 0);
    Phonon::createPath(mediaObject, audioOutput);

    mediaObject->setCurrentSource(*source);
    mediaObject->play();
```

The above lines of code can be visualized as follows



The media object

- Needs to be told of the source from which it can fetch audio-data input. The media source can be specified as a file name or URL; or in terms of a MediaObject instance.
- Needs to be told of the device into which it should dump the output to. In this case the output is a AudioOutput instance.

Compiling the program

To compile the above program we need a project file that provides all the cues to QMake to generate the make file. The project file looks like this.

```
PHONON_SDK = /home/user/Phonon/phonon-devel
TEMPLATE = app
TARGET = audio_player
DEPENDPATH += .
INCLUDEPATH += . $${PHONON_SDK}/include

LIBS        += -lphonon -lphonon_gstreamer -L$${PHONON_SDK}/lib -L$${PHONON_SDK}/lib/phonon_backend

# Input
SOURCES += Main.cpp
```

After creating the project file you can use a simple qmake, make command pair to compile the program. Once compiled you can invoke the audio_player program and pass to it the complete path of a music file you want to play and hear it play ! :).

> ***Note for people who want to use the example program to play MP3 files****: You will need GStreamer plugins for playing MP3 files. Ofcourse you can figure out how to get that.*
>
> ***Interesting Thought:*** Try passing as parameter to the audio_player program a video file. You will notice that the program still renders the file, but only the audio part.

Simple video player program

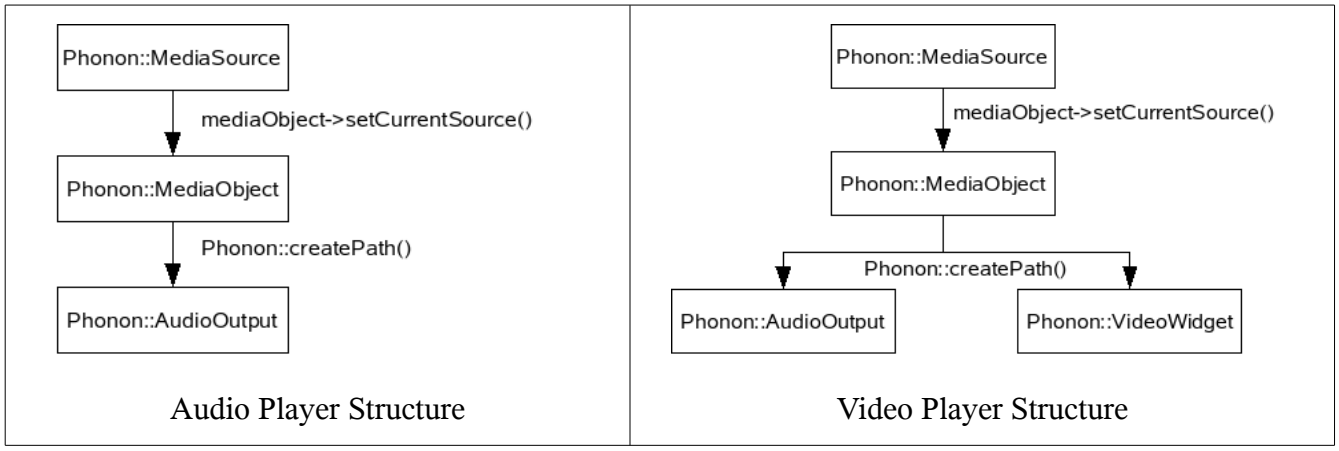
Like I said while explaining the previous program, MediaObject plays media files. While playing the media files it sends

- audio information to a media node that can accept audio data like Phonon::AudioOutput.
- video information to a media node that can show video data like Phonon::VideoWidget.

So by making a small modification to our audio_player program we will be able to view videos as well.

```
Phonon::MediaSource* source = new Phonon::MediaSource( argv[1] );
Phonon::MediaObject* mediaObject = new Phonon::MediaObject( 0 );
Phonon::AudioOutput *audioOutput = new Phonon::AudioOutput(Phonon::MusicCategory, 0);
Phonon::VideoWidget* videoWidget = new Phonon::VideoWidget(0);
Phonon::createPath(mediaObject, audioOutput);
Phonon::createPath(mediaObject, videoWidget);

mediaObject->setCurrentSource(*source);
mediaObject->play();
```

The structure now change is explained in the following block diagrams.



Compile and execute instructions remain the same. Shown below is a screenshot of the video widget

playing the now famous “Experience Ubuntu” video.