



## Qt视频播放器[QMediaPlayer+QVideowidget]

2023-08-15 584 发布于北京

版权

**简介：** 本代码在Window10下运行，利用qMediaPlayer和qvideowidget实现视频文件mp4的播放，并且提供进度显示，还可以通过拖动进度条来变换播放位置

## 参考

Qt实现视频播放器

Qt播放视频报错 DirectShowPlayerService::doRender: Unresolved error code 0x80040266

安装K-Lite 解码器

本代码在Window10下运行，利用 qMediaPlayer 和 qvideowidget 实现视频文件mp4的播放，并且提供进度显示，还可以通过拖动进度条来变换播放位置

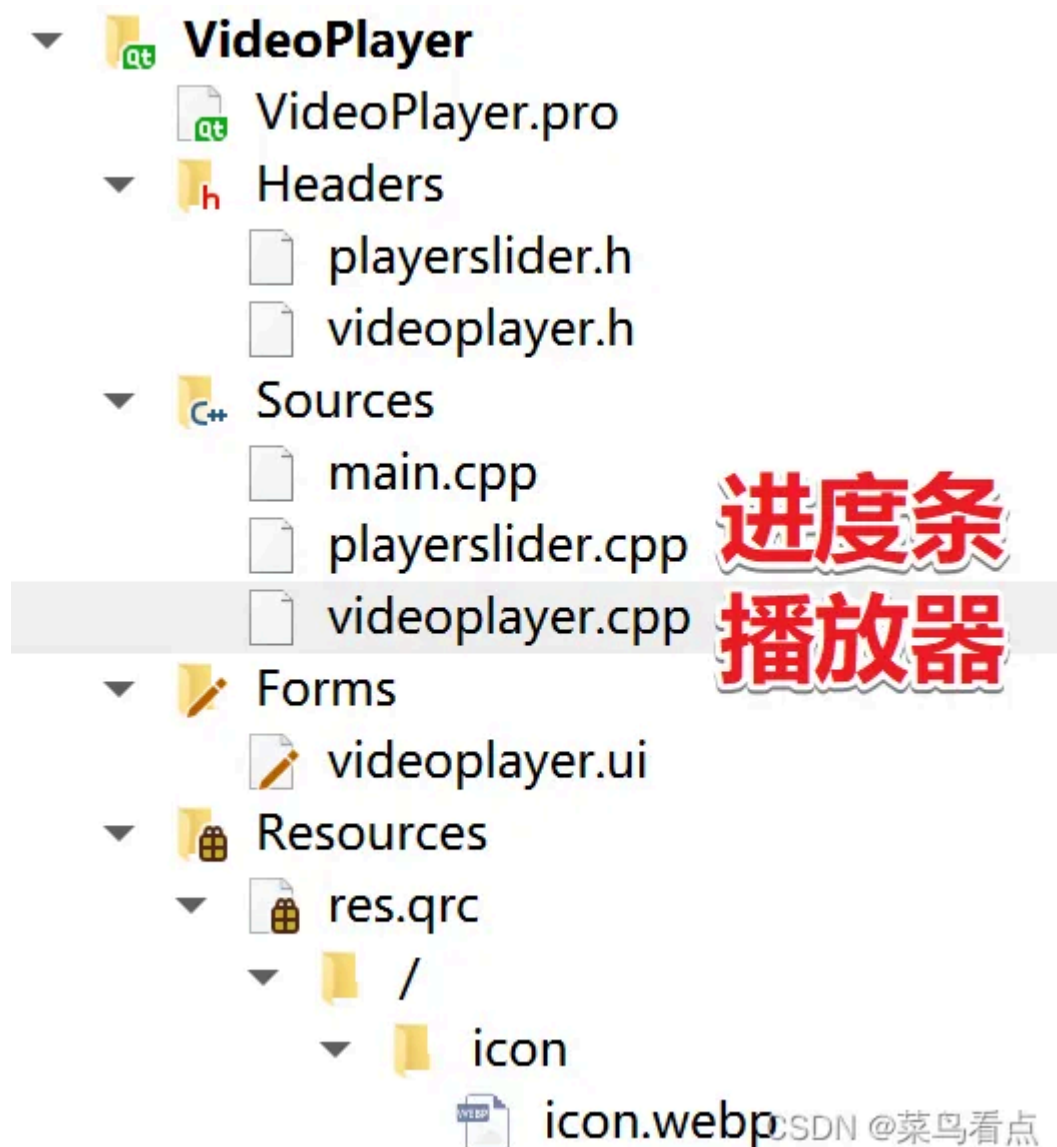
## 一、安装K-Lite 解码器

K-Lite 是一个万能解码器，它可以扩展播放器功能，使其能解码更多格式。

安装K-Lite 解码器

或者在官网[K-Lite官网](#)

## 二、Qt代码结构



## VideoPlayer.pro



```

QT      += core gui
QT      += multimedia      #使用多媒体模块
QT += multimedialogs      #使用QVideoWidget 视频显示组件
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++11 #c++17在使用emit时会出错

# You can make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \
    main.cpp \
    playerslider.cpp \
    videoplayer.cpp

HEADERS += \

```

```
playerslider.h \
videoplayer.h
```

```
FORMS += \
    videoplayer.ui
```

```
# Default rules for deployment.
qnx: target.path = /tmp/${TARGET}/bin
else: unix:!android: target.path = /opt/${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
```

```
RESOURCES += \
```

## main.cpp



```
#include "videoplayer.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    videoplayer w;
    w.show();
    return a.exec();
}
```

## videoplayer.h 播放器



```
#ifndef VIDEOPLAYER_H
#define VIDEOPLAYER_H

#include <QWidget>
#include <QtMultimedia>
#include <QVideoWidget> // 视频显示组件窗口
#include "playerslider.h"

QT_BEGIN_NAMESPACE
namespace Ui { class videoplayer; }
QT_END_NAMESPACE
class PlayerSlider;
class videoplayer : public QWidget
{
    Q_OBJECT
```

```

public:
    videoplayer(QWidget *parent = nullptr);
    ~videoplayer();

    bool m_bReLoad; // 已经载入还没设置进度条最大值

private:
    Ui::videoplayer *ui;
    QVideowidget *m_pPlayerWidget; // 视频显示组件
    QMediaPlayer * m_pPlayer; // 媒体播放器类

public slots:
    void OnSetMediaFile(void); // 载入
    void OnSlider(qint64); // 信号内容的位置已更改为位置，以毫秒表示，进度条也要变
    void OnDurationChanged(qint64); // 信号表示内容的持续时间已更改为时长，以毫秒表示
    void OnStateChanged(QMediaPlayer::State); // Player对象的状态已经改变

};

```

## videoplayer.cpp 播放器



```

#include "videoplayer.h"
#include "ui_videoplayer.h"

#include <QFileDialog> // 文件选择窗口
#include <QDebug>
videoplayer::videoplayer(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::videoplayer)
{
    ui->setupUi(this);
    setWindowTitle("播放器");
    setWindowIcon(QIcon(":/icon/icon.webp")); // 图标
    setFixedSize(800, 600); // 设置固定大小

    // 1. 放置播放窗口
    m_pPlayer = new QMediaPlayer; // 媒体播放器类
    m_pPlayerWidget = new QVideowidget; // 视频显示组件窗口
    m_pPlayer->setVideoOutput(m_pPlayerWidget); // 视频输出: 设置视频输出, 允许用户将视频
    ui->verticalLayout->addWidget(m_pPlayerWidget); // 将视频显示组件窗口添加到QVBoxLayout

    // 设置视频小部件是否自动填充背景。true则视频小部件将自动填充背景, 以便在视频播放期间保持
    m_pPlayerWidget->setAutoFillBackground(true);

    // 2. 界面美化
    QPalette qplte; // 调色板
    qplte.setColor(QPalette::Window, QColor(0, 0, 0)); // 透明
    m_pPlayerWidget->setPalette(qplte); // 设置窗口部件的调色板

```

```

// 3. 槽函数
//载入
connect(ui->BtnLoad, SIGNAL(clicked()), this, SLOT(OnSetMediaFile()));
//播放
connect(ui->BtnPlay, SIGNAL(clicked()), m_pPlayer, SLOT(play()));// 对QMedia
//停止
connect(ui->BtnStop, SIGNAL(clicked()), m_pPlayer, SLOT(stop()));// 对QMedia

// Player对象的状态已经改变。
connect(m_pPlayer, SIGNAL(stateChanged(QMediaPlayer::State)), this, SLOT(OnS

ui->BtnStop->setEnabled(false);// 停止：默认不可用
//设置滑块行为
m_bReLoad = true;
ui->slider->setEnabled(false);// 进度条：默认不可用

// 信号内容的位置已更改为位置，以毫秒表示，进度条也要变
connect(m_pPlayer, SIGNAL(positionChanged(qint64)), this, SLOT(OnSlider(qint64)));
// 信号表示内容的持续时间已更改为时长，以毫秒表示，第一次为进度条 的最大值和最小值
connect(m_pPlayer, SIGNAL(durationChanged(qint64)), this, SLOT(OnDurationChanged(qint64)));
// 进度条：鼠标移动位置;setPosition()设置当前播放的位置,实现跳转播放的功能
connect(ui->slider, SIGNAL(sigProgress(qint64)), m_pPlayer, SLOT(setPosition(qint64)));

}

videoplayer::~videoplayer()
{
    delete m_pPlayer;
    delete m_pPlayerWidget;
    delete ui;
}
//载入
void videoplayer::OnSetMediaFile()
{
    QFileDialog dialog(this);//文件选择窗口
    dialog.setFileMode(QFileDialog::AnyFile);//设置文件模式(文件/文件夹)：任意文件，无限制
    QStringList fileNames;
    if (dialog.exec())
        fileNames = dialog.selectedFiles();// 存所有选择的文件

    if(!fileNames.empty())
    {
        qDebug()<<"文件名:"<<fileNames[0];
        m_pPlayer->setMedia(QUrl::fromLocalFile(fileNames[0]));// 设置媒体信息
        m_bReLoad = true;
        ui->slider->setValue(0);//进度条数字在0(开始位置)
        setWindowTitle(fileNames[0]);
    }
}
// 信号内容的位置已更改为位置，以毫秒表示，进度条也要变
void videoplayer::OnSlider(qint64 i64Pos)
{

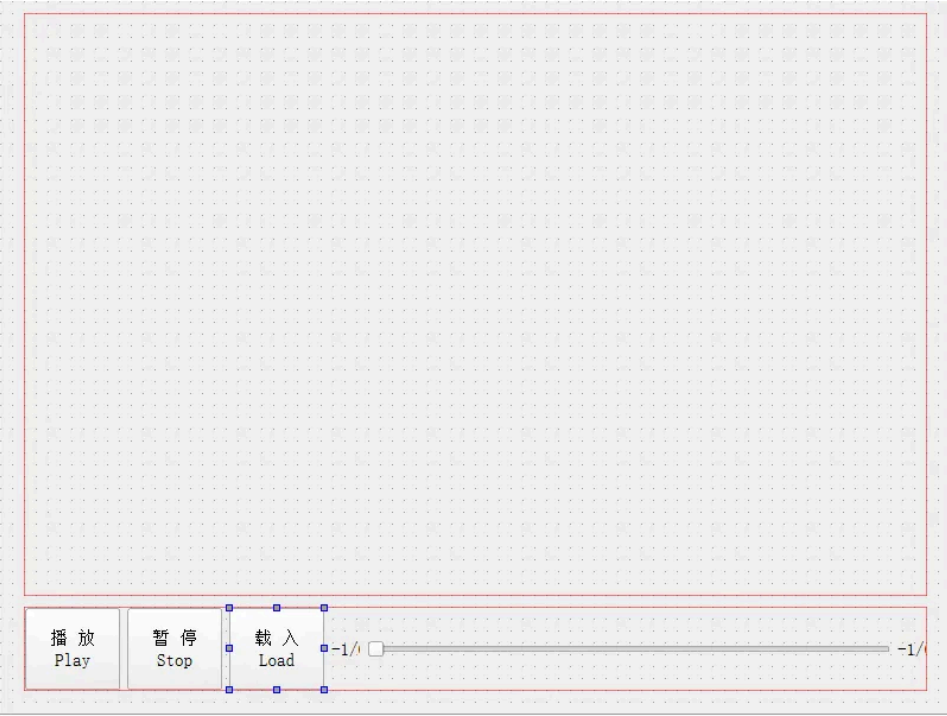
```

```
// 自定义的位置移动==setValue(i64Pos);
ui->slider->setProgress(i64Pos);
}
// 信号表示内容的持续时间已更改为时长，以毫秒表示，进度条 的最大值和最小值
void videoplayer::OnDurationChanged(qint64 i64Duration)
{
    // 第一次i64Duration 为最大值
    if(i64Duration > 0 && m_bReLoad) // 最大值>0并且 已经载入还没设置进度条最大值
    {
        ui->slider->setRange(0, i64Duration);

        ui->minTime->setText(QString::number(0));
        QTime time = QTime::fromMSecsSinceStartOfDay(i64Duration);
        ui->maxTime->setText(time.toString("hh:mm:ss:zzz"));

        m_bReLoad = false; // 不能再设置进度条最大值
    }
}
// Player对象的状态已经改变
void videoplayer::OnStateChanged(QMediaPlayer::State enumState)
{
    if(QMediaPlayer::StoppedState == enumState) // 停止
    {
        ui->BtnPlay->setEnabled(true);
        ui->BtnStop->setEnabled(false);
        ui->slider->setEnabled(false);
    }
    else if(QMediaPlayer::PlayingState == enumState) // 播放
    {
        ui->BtnPlay->setEnabled(false);
        ui->BtnStop->setEnabled(true);
        ui->slider->setEnabled(true);
    }
}
```

## videoplayer.ui 播放器



The image shows the Qt Designer interface for a video player. The main window contains a large video area and a control bar at the bottom. The control bar includes buttons for '播放' (Play), '暂停' (Stop), '载入' (Load), and a progress slider. The right sidebar displays the Object Inspector and Properties panel for the 'BtnLoad' widget.

**Object Inspector:**

对象	类
videoplayer	QWidget
horizontalLayout	QHBoxLayout
BtnLoad	QPushButton
BtnPlay	QPushButton
BtnStop	QPushButton
maxTime	QLabel
minTime	QLabel
slider	PlayerSlider
verticalLayout	QVBoxLayout

**Properties Panel for BtnLoad:**

属性	值
QObject	
objectName	BtnLoad
QWidget	
enabled	<input checked="" type="checkbox"/>
geometry	[[173, 1), 80 x 69]
X	173
Y	1
宽度	80
高度	69
sizePolicy	[Minimum, Preferred, 0, 0]
水平策略	Minimum
垂直策略	Preferred
水平伸展	0
垂直伸展	0
minimumSize	0 x 0
宽度	0
高度	0

## playerslider.h 自定义进度条



```

#ifndef PLAYERSLIDER_H
#define PLAYERSLIDER_H

#include <QObject>
#include <QSlider>
#include <QMouseEvent>

class PlayerSlider : public QSlider
{
    Q_OBJECT
signals:
    void sigProgress(qint64 i64Pos);

public:
    PlayerSlider(QWidget * parent = 0);

    bool    m_bPressed;
    void    setProgress(qint64);

    void    mousePressEvent(QMouseEvent *e)override;// 按下
    void    mouseMoveEvent(QMouseEvent *e)override;// 移动
    void    mouseReleaseEvent(QMouseEvent *e)override;// 抬起释放

```

```
};  
  
#endif // PLAYERSLIDER_H
```

## playerslider.cpp 自定义进度条

作用：

- 接收QMediaPlaeer设置的进度信息，更新进度条；
- 当用户操作进度条时，不再让进度条响应QMediaPlaeer设置的进度信息；
- 当用户完成对进度条的拖动后，向QMediaPlaeer发送播放位置更新信息。

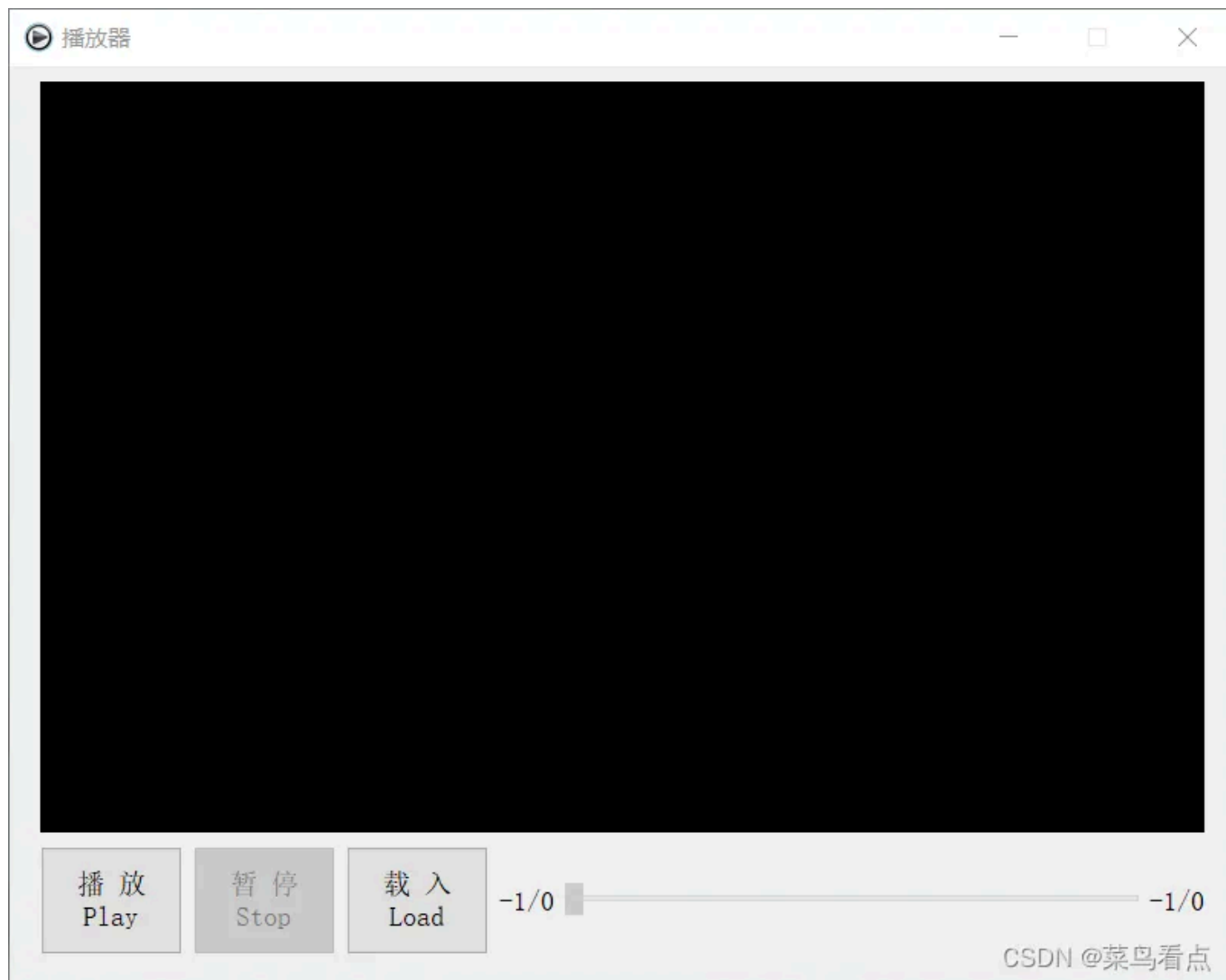


```
#include "playerslider.h"  
  
PlayerSlider::PlayerSlider(QWidget * parent) : QSlider(parent)  
{  
    m_bPressed = false; // 是否允许进度条的进度设置  
}  
// 进度条的进度设置  
void PlayerSlider::setProgress(qint64 i64Progress)  
{  
    if(!m_bPressed)  
        setValue(i64Progress);  
}  
// 按下  
void PlayerSlider::mousePressEvent(QMouseEvent *e)  
{  
    m_bPressed = true; // 用户操作进度条时，不再让进度条响应QMediaPlaeer设置的进度信息；  
    QSlider::mousePressEvent(e); // 必须有这句，否则手动不能移动滑块  
}  
// 移动  
void PlayerSlider::mouseMoveEvent(QMouseEvent *e)  
{  
    QSlider::mouseMoveEvent(e); // 必须有这句，否则手动不能移动滑块  
}  
// 抬起释放  
void PlayerSlider::mouseReleaseEvent(QMouseEvent *e)  
{  
    m_bPressed = false;  
    qint64 i64Pos = (qint64)value();  
    emit sigProgress(i64Pos); // 发出位置改变信号  
    QSlider::mouseReleaseEvent(e); // 必须有这句，否则手动不能移动滑块  
}
```

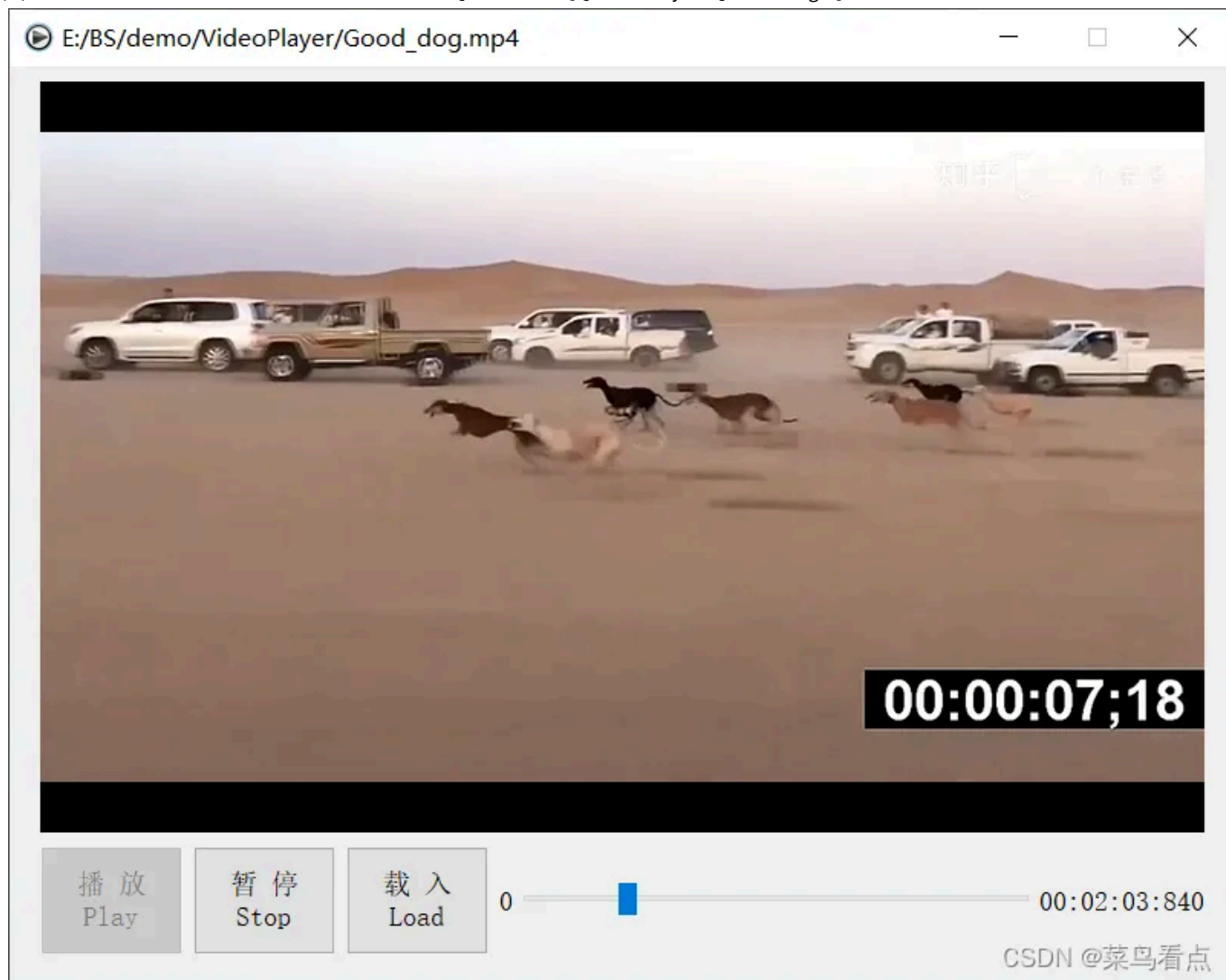


# 效果

- 启动



- 播放



## 源码文件

有道云：[QT视频播放器](#)

## 错误解决

- 报错



```
error: undefined reference to `PlayerSlider::sigProgress(long long)'  
debug/playerslider.o: In function `PlayerSlider::mouseReleaseEvent(QMouseEvent*)':  
.....VideoPlayer/playerslider.cpp:29: undefined reference to `PlayerSlider::sigProgress(long long)'
```

- 解决



这个错误提示是链接器 (Linker) 在链接阶段发现了一个未定义的符号, 即链接器无法找到 ``PlayerSlider::sigProgress(long long)`` 的实现。这种错误通常发生在有函数或变量的声明但没有实现的情况下。



可能原因如下:

1. 在 ``playerslider.cpp`` 中调用了 ``PlayerSlider::sigProgress(long long)`` 函数, 但是没有在 ``playerslider.h`` 文件中声明或者实现该函数。
2. ``PlayerSlider::sigProgress(long long)`` 函数在 ``playerslider.h`` 文件中被声明为信号 (signal), 但是没有在 ``playerslider.cpp`` 中实现, 或者实现了一个不符合声明的函数。
3. ``PlayerSlider::sigProgress(long long)`` 函数在另外的文件中被实现, 但是没有将该文件加入到项目中, 导致链接器无法找到实现。

解决方法:

1. 检查 ``playerslider.h`` 文件中是否声明了 ``PlayerSlider::sigProgress(long long)`` 函数, 并确认该函数的声明是否正确。
2. 检查 ``playerslider.cpp`` 文件中是否正确实现了 ``PlayerSlider::sigProgress(long long)`` 函数, 并确认函数的实现与声明一致。
3. 检查项目文件 (如 ``.pro`` 文件) 中是否正确地包含了所有需要的源代码文件, 并确认没有漏掉任何实现。

如果以上方法仍然无法解决问题, 可能需要进一步检查项目设置、Qt版本等因素。

CSDN @菜鸟看点

## 在 VideoPlayer.pro 文件

把 `CONFIG += c++17` 改为 `CONFIG += c++11`

文章标签: JavaScript

关键词: Qt视频播放器 Qt qmediaplayer Qt qvideowidget

## 评论

登录后可评论

## 相关文章

linux系统中QT里面的视频播放器的实现方法