

almslfniswd

flowingwind

博客园 首页 新随笔 联系 订阅 管理

# QT5 文件读写操作

## QFile Class

### 1. read读文件

- 加载文件对象 `QFile file("文件地址");`
- 打开加载的文件`file.open(打开方式);`
- 操作文件
- 关闭打开的文件`file.close();`




```
void Widget::on_pushButton_clicked()
{
    QFile file("L:/qtpro/_qtApp/text/t.txt");
    file.open(QIODevice::ReadOnly | QIODevice::Text);
    QByteArray t = file.readAll();
    ui->text_r->setText(QString(t));
    file.close();
}
```




### 2. wirte写文件

- 以纯文本的形式读取要保存文件到QString对象 `//ui->text_e->toPlainText();`
- 创建QFile 对象保存文件
- 打开QFile对象
- 写入文件操作
- 关闭打开的文件;



```
void Widget::on_pushButton_2_clicked()
{
    QString e = ui->text_e->toPlainText();
    QFile file("L:/qtpro/_qtApp/text/e.txt");
    file.open(QIODevice::WriteOnly | QIODevice::Text);
    file.write(e.toUtf8());
    file.close();
}
```



### 细节优化处理

- `read`文件添加读取文件选择项 `QFileDialog::getOpenFileName();`
- 打开文件是否成功的判断;
- 按行读取文件, 可控制读取行数与每行字符数;
- `write`文件创建保存路径`QFileDialog::getSaveFileName();`

### 公告

昵称： 疯颠研究者  
园龄： 1年11个月  
粉丝： 6  
关注： 0  
[+加关注](#)

<	2019年1:			
日	一	二	三	
1	2	3	4	
8	9	10	11	
15	16	17	18	
22	23	24	25	
29	30	31	1	
5	6	7	8	

### 搜索

### 我的标签

- c++(1)
- QT(1)
- unicode(1)
- 编码(1)
- 控制台编程 贪吃蛇 c/c
- 乱码(1)

### 随笔分类

c/c++(15)



```
void Widget::on_pushButton_clicked()
{
    QFile file;
    QString f = QFileDialog::getOpenFileName(this, QString("选择文件"), QString("/"),QString("TEXT(*.txt)"));
    file.setFileName(f);
    if(file.open(QIODevice::ReadOnly | QIODevice::Text))
    {
        QByteArray t ;
        while(!file.atEnd())
        {
            t += file.readLine();
        }
        ui->text_r->setText(QString(t));
        file.close();
    }
}

void Widget::on_pushButton_2_clicked()
{
    QString e = ui->text_e->toPlainText();
    QFile file;
    file.setFileName(QFileDialog::getSaveFileName(this, QString("保存路径"), QString("/"),QString("TEXT(*.txt)"));
    file.open(QIODevice::WriteOnly | QIODevice::Text);
    file.write(e.toUtf8());
    file.close();
}

```



各编码转换

```
QString -> QByteArray      QString.toUtf8();

QByteArray -> std::string  QByteArray.toStdString();

std::string -> char *      string.data();
```

常用静态函数：

```
QFileDialog::getOpenFileName()    //获取指定文件路径名返回QString
QFileDialog::getExistingDirectory() //获取指定路径返回QString
QFileDialog::getSaveFileName()    //获取指定保存路径名返回QString
```

辅助配合使用的类：

QFileInfo class

获取文件信息；



QFileInfo类用于读取文件的属性信息

```
QFile file(f);
QFileInfo info(file);

qDebug() << info.exists();
qDebug() << info.isFile();
qDebug() << info.isReadable();
qDebug() << info.isWritable();
qDebug() << info.created();
qDebug() << info.lastRead();
qDebug() << info.lastModified();
qDebug() << info.path();
qDebug() << info.fileName();
qDebug() << info.suffix();
qDebug() << info.size();
```

cmd控制台(2)

javaScript(5)

php

python

qt/c++(5)

QT笔记(6)

环境配制(2)

零碎知识点(7)

数据结构与算法(6)

随笔档案

2018年2月(18)

2018年1月(17)

2017年12月(5)

最新评论

1. Re:QT5 Thread线程

新方法：1.创建继承Ob  
在线程中实现的方法在/  
线程中实例化A对象a,再  
类对象b 3.通过a.move  
将a对象的实...

2. Re:编码(ACSII uni  
输出中文乱码深入分析  
  
收藏了！

3. Re:编码(ACSII uni  
输出中文乱码深入分析  
  
如果编辑编译和执行都  
不搞那么多 就方便好用  
不然一个编码搞死人



## 二、文本流与数据流

QT中将文件分为文本文件和数据文件，文本文件内容是可读的文本字符，数据文件的内容是二进制数据。

QFile直接支持文本文件和数据文件的操作，主要函数接口如下：

- qint64 read( char \* data, qint64 maxSize) //数据流读取
- QByteArray read( qint64 maxSize) //文本流方式读取
- QByteArray readAll() //文本流方式读取
- QByteArray readLine()//文本流方式读取
- qint64 write(const char \* data, qint64 maxSize)
- qint64 write(const QByteArray & byteArray)

为了简化文本文件和数据文件的读写操作，QT提供了QTextStream和QDataStream辅助类。QTextStream可将写入的数据全部转换为可读文本，QDataStream可将写入的数据根据类型转换为二进制数据。

**QTemporaryFile**是QT中的临时文件操作类，用来安全创建全局唯一的临时文件，QTemporaryFile对象销毁时对应的临时文件将被删除，临时文件的打开方式为QIODevice::ReadWrite,临时文件常用于大数据传递或者进程间通信场合。



```
QTemporaryFile tempFile;  
if( tempFile.open() )  
{  
    tempFile.write("D.T.Software");  
    tempFile.close();  
}
```



## QDataStream Class

数据流操作文件：

```
    创建流对象 QDataStream date;  
  
    int a= xxxx;  
  
    string b = "xxxxxxxxx" ;  
  
    将数据存在流中 date >> a >> b;  
  
    int aa;  
  
    string bb;  
  
    从流中取出数据 date << aa << bb;
```

QDataStream在不同的QT版本中数据流文件格式可能是不同的，如果数据流文件需要在不同版本的QT程序间传递时需要考虑版本问题。

```
void setVersion(int v)  
  
int version() const
```

## QTextStream Class

文本方式操作文件：

```
    创建流对象 QTextStream date;  
  
    date.setCodec();支持对文件读取编码设置(有效解决乱码问题)
```

## QBuffer

QBuffer 类为QByteArray提供QIODevice接口。  
  
目前先理解为一个创建一个缓存文件；

QT中预定义了缓冲区的类QBuffer，可以将缓冲区看成一种特殊的IO设备，文件流辅助类可以直接用于操作缓冲区。QBuffer缓冲区写入和读取的数据必须是同一种数据类型，不能混合多种数据类型。

QBuffer的使用场合：

4. Re:编码(ACSII uni  
输出中文乱码深入分析

顶

5. Re:QT5 Thread线程  
函数里面要加入大量代码  
在里面调用queue 自己

### 阅读排行榜

1. QT5 文件读写操作(4)
2. QT5 Thread线程(3)
3. c++数组的引用(17)
4. QT5线程关闭(1575)
5. c/c++ console(控制台)(929)

### 评论排行榜

1. 编码(ACSII unicod  
出中文乱码深入分析(4)
2. QT5 Thread线程(2)

### 推荐排行榜

1. QT5 文件读写操作(4)
2. QT5 Thread线程(3)
3. c++数组的引用(2)
4. gcc/g++以c++11为基准编译(1)
5. 编码(ACSII unicod  
出中文乱码深入分析(1)

- A、线程间不同类型的数据传递
- B、缓存外部设备中的数据返回
- C、数据读取速度小于写入速度

## 总结:

读写操作主要方法有read();readAll();readline(),write();

## 附录:


### 目录操作

#### 1、QDir

QT中提供了目录操作类QDir, QDir功能如下:


- A、目录分隔符统一使用 '/'
- B、能够对目录进行任意操作 (创建、删除、重命名)
- C、能够获取指定目录中的所有条目 (文件和文件夹)
- D、能够使用过滤字符串获取指定条目
- E、能够获取系统中的所有根目录

QDir使用方法如下:



```
QDir dir;
QString path("../qt/test");
if(!dir.exists())
{
    dir.mkdir(path);
}
else
{
    dir.cd(path);
    QStringList list = dir.entryList();
    for(int i = 0; i < list.count(); i++)
    {
        qDebug() << list[i];
    }
}

//计算文件大小
unsigned int FileSize(QString path)
{
    QFileInfo info(path);
    unsigned int ret = 0;
    if(info.isFile())
    {
        ret = info.size();
    }
    else if(info.isDir())
    {
        QDir dir(path);
        QFileInfoList list = dir.entryInfoList();
        for(int i = 0; i < list.count(); i++)
        {
            if((list[i].fileName() != ".") && (list[i].fileName() != ".."))
            {
                ret += FileSize(list[i].absoluteFilePath());
            }
        }
    }
    return ret;
}
```



#### 2、QFileSystemWatcher

QT中预定义了用于监控文件和目录变化的类QFileSystemWatcher,

QFileSystemWatcher主要功能如下：

- A、能够监控特定目录和文件的状态
  - B、能够同时对多个文件和目录进行监控
  - C、当目录或文件发生改变时触发信号
  - D、通过信号与槽的机制捕捉信号并做出响应
- 通常要使用QFileSystemWatcher需要自定义文件监视类。

分类： QT笔记

好文要顶

关注我

收藏该文

疯颠研究者

关注 - 0

粉丝 - 6

+加关注

4

0

« 上一篇： QT5 Even 事件

» 下一篇： QT5 网络通讯

posted @ 2018-01-23 15:43 疯颠研究者 阅读(42993) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#) ， [访问](#) [网站首页](#)。

- 【推荐】超50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库
- 【推荐】腾讯云热门云产品限时秒杀，爆款1核2G云服务器99元/年！
- 【推荐】阿里云双11返场来袭，热门产品低至一折等你来抢！
- 【活动】京东云服务器\_云主机低于1折，低价高性能产品备战双11
- 【活动】ECUG For Future 技术者的年度盛会（杭州，1月4-5日）



- 相关博文：
- 14.QT-QFile文件,QBuffer缓冲区,QDir目录,QFileSystemWatcher文件系统监视
  - Java中IO流，输入输出流概述与总结
  - Java进阶篇（五）——Java的I/O技术
  - JAVA中的I/O流以及文件操作
  - 文件和文件夹操作
  - » 更多推荐...
- 96秒100亿！ 哪些“黑科技”支撑全球最大流量洪峰？

- 最新 IT 新闻：
- 百年理论预言被证实！我国科学家捕捉到水结冰的关键一瞬

- 腾讯视频因超前点播被起诉 要求赔偿损失500元，网友：干得漂亮！
  - 苹果谷歌亚马逊联手创建开源智能家居标准 设备相互兼容
  - 印度人的一天，中国如影随形
  - 不接入就下架！腾讯宣布：要将32款游戏退市 原因在此
- » 更多新闻...

Copyright © 2019 疯颠研究者

Powered by .NET Core 3.1.0 on Linux