



第9章 Qt 5文件及磁盘处理

9.1 读写文本文件

9.2 读写二进制文件

9.3 目录操作与文件系统

9.4 获取文件信息


9.5 监视文件和目录变化





9.1 读写文本文件

9.1.1 QFile类读写文本

(1) 建立一个工程。选择“文件”→“新建文件或项目...”菜单项，在弹出的对话框中选择“项目”组下的“应用程序”→“ Qt控制台应用”菜单项，单击“选择”按钮。

(2) 在弹出的对话框中对该工程进行命名并选择保存工程的路径，这里将工程命名为“TextFile”，单击“下一步”按钮，再次单击“下一步”按钮，最后单击“完成”按钮，完成该文件工程的建立。





9.1.1 QFile类读写文本

(3) 源文件“main.cpp”的具体实现代码如下:

```
#include <QCoreApplication>
#include <QFile>
#include <QtDebug>

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

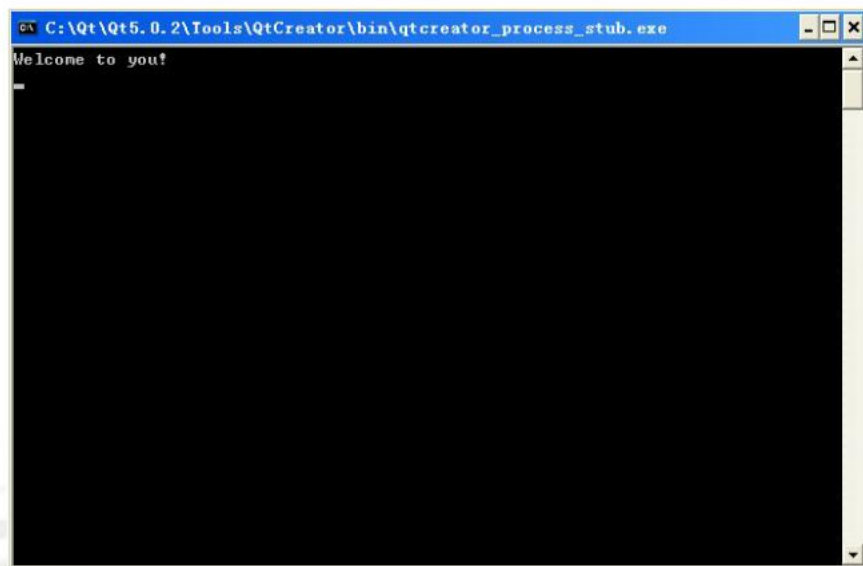
    QFile file("textFile1.txt");
    if(file.open(QIODevice::ReadOnly))
    {
        char buffer[2048];
        qint64 lineLen = file.readLine(buffer, sizeof(buffer));
        if(lineLen != -1)
        {
            qDebug() << buffer;
        }
    }

    return a.exec();
}
```



9.1.1 QFile类读写文本

(4) 选择“构建”→“构建项目”“TextFile”菜单项，首先编辑本例所用的文本文件“textFile1.txt”，保存在项目D:\Qt\CH9\CH901\build-TextFile-Desktop_Qt_5_0_2_MinGW_32bit-Debug目录下，然后运行程序，运行结果如图9.1所示。





9.1.2 QTextStream类读写文本

(1) 源文件“main.cpp”的具体实现代码如下：

```
#include <QCoreApplication>
#include <QFile>
#include <QTextStream>

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    QFile data("data.txt");
    if(data.open(QFile::WriteOnly|QFile::Truncate))
    {
        QTextStream out(&data);
        out<<QObject::tr("score:")<<qSetFieldWidth(10)<<left<<90<<endl;
    }

    return a.exec();
}
```



9.1.2 QTextStream类读写文本

其中，

● **if(data.open(QFile::WriteOnly|QFile::Truncate))**：参数QFile::Truncate表示将原来文件中的内容清空。输出时将格式设为左对齐，占10个字符位置。

● **out<<QObject::tr("score:")<<qSetFieldWidth(10)<<left<<90<<endl**：用户使用格式化函数和流操作符设置需要的输出格式。其中，qSetFieldWidth()函数是设置字段宽度的格式化函数。除此之外，QTextStream还提供了其他一些格式化函数，见表9.1。

函 数 ^❶	功 能 描 述 ^❶
qSetFieldWidth(int width) ^❶	设置字段宽度 ^❶
qSetPadChar(QChar ch) ^❶	设置填充字符 ^❶
qSetRealNumberPercision(int precision) ^❶	设置实数精度 ^❶



9.1.2 QTextStream类读写文本

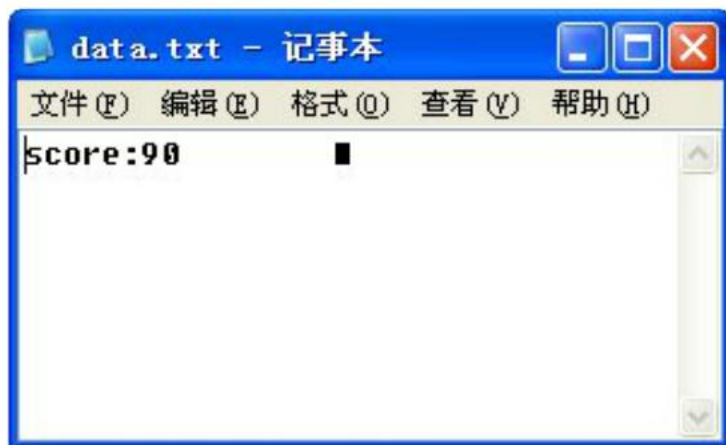
其中，left操作符是QTextStream定义的类似于<iostream>中的流操作符。QTextStream还提供了其他一些流操作符，见表9.2。

操 作 符	作 用 描 述
bin	设置读写的整数为二进制数
oct	设置读写的整数为八进制数
dec	设置读写的整数为十进制数
hex	设置读写的整数为十六进制数
showbase	强制显示进制前缀，如十六进制(0x)、八进制(0)、二进制(0b)
forcesign	强制显示符号(+, -)
forcepoint	强制显示小数点
noshowbase	不显示进制前缀
noforcesign	不显示符号
uppercasebase	显示大写的进制前缀
lowercasebase	显示小写的进制前缀
uppercasedigits	用大写字母表示
lowercasedigits	用小写字母表示
fixed	固定小数点表示
scientific	科学计数法表示
left	左对齐
right	右对齐
center	居中
endl	换行
flush	清除缓冲



9.1.2 QTextStream类读写文本

(2) 运行此程序后，可以看到在项目的D:\Qt\CH9\CH902\build-TextFile2-Desktop_Qt_5_0_2_MinGW_32bit-Debug文件夹下自动建立了一个文本文件“data.txt，”打开后看到的内容如图9.2所示。





9.2 读写二进制文件

(1) 头文件“mainwindow.h”的具体代码如下：

```
#include <QMainWindow>

class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    MainWindow(QWidget *parent = 0);
    ~MainWindow();

    void fileFun();
};
```



9.2 读写二进制文件

(2) 源文件“mainwindow.cpp”的具体代码如下：

```
#include "mainwindow.h"
#include <QtDebug>
#include <QFile>
#include <QDataStream>
#include <QDate>

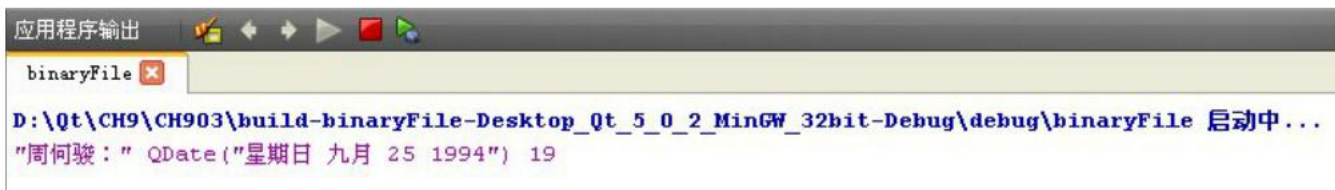
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
{
    fileFun();
}
```

函数fileFun()完成主要功能，其具体代码。



9.2 读写二进制文件

(3) 运行结果如图9.3所示。



```
应用程序输出
binaryFile
D:\Qt\CH9\CH903\build-binaryFile-Desktop_Qt_5_0_2_MinGW_32bit-Debug\debug\binaryFile 启动中...
周何骏: " QDate("星期日 九月 25 1994") 19
```

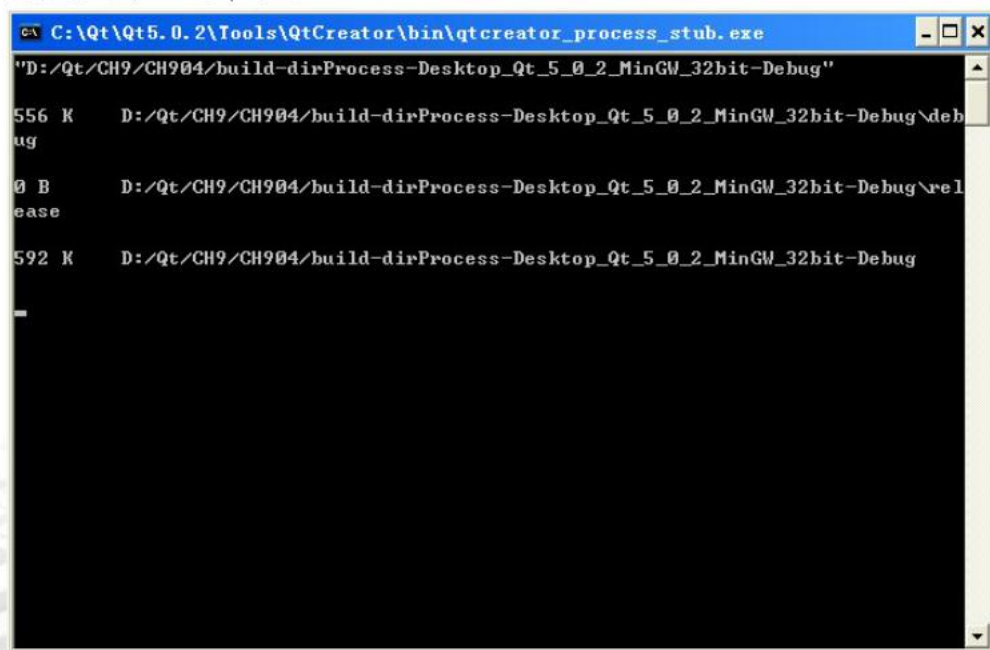


9.3 目录操作与文件系统

9.3.1 文件大小及路径获取

源文件“main.cpp”的具体代码。

运行结果如图9.4所示。



```
C:\Qt\Qt5.0.2\Tools\QtCreator\bin\qtcreator_process_stub.exe
"D:/Qt/CH9/CH904/build-dirProcess-Desktop_Qt_5_0_2_MinGW_32bit-Debug"

556 K    D:/Qt/CH9/CH904/build-dirProcess-Desktop_Qt_5_0_2_MinGW_32bit-Debug\deh
ug

0 B      D:/Qt/CH9/CH904/build-dirProcess-Desktop_Qt_5_0_2_MinGW_32bit-Debug\rel
ease

592 K    D:/Qt/CH9/CH904/build-dirProcess-Desktop_Qt_5_0_2_MinGW_32bit-Debug
```



9.3.2 文件系统浏览

工程FileView.pro的具体内容如下。

(1) 在头文件“fileview.h”中，类FileView继承于QDialog类，[具体代码。](#)

(2) 源文件“fileview.cpp”[的具体代码。](#)

槽函数slotShow()实现了显示目录dir下的所有文件，具体内容如下：

```
void FileView::slotShow(QDir dir)
{
    QStringList string;
    string<<"*";
    QFileInfoList list=dir.entryInfoList(string,QDir::AllEntries,QDir:: DirsFirst);
    showFileInfoList(list);
}
```



9.3.2 文件系统浏览

(3) 运行结果如图9.5所示。





9.4 获取文件信息

下面的例子演示了如何利用QFileinfo类获得文件信息，如图9.6所示（详细内容见代码CH906）。





9.5 监视文件和目录变化

(1) 在头文件“watcher.h”中，类Watcher继承自QWidget类，其具体内容如下：

```
#include <QWidget>
#include <QLabel>
#include <QFileSystemWatcher>

class Watcher : public QWidget
{
    Q_OBJECT

public:
    Watcher(QWidget *parent = 0);
    ~Watcher();

public slots:
    void directoryChanged(QString path);

private:
    QLabel *pathLabel;
    QFileSystemWatcher fsWatcher;
};
```



9.5 监视文件和目录变化

(2) 源文件“watcher.cpp”的[具体内容](#)。

响应函数directoryChanged()使用消息对话框提示用户目录发生了改变，具体实现代码如下：

```
void Watcher::directoryChanged(QString path)
{
    QMessageBox::information(NULL, tr("目录发生变化"), path);
}
```

(3) 运行结果如图9.7所示。

