【文本编辑器】二、文件操作功能（下）

原创 Qt 学习 Qt 学习 2020-02-14

本篇是《【文本编辑器】二、文件操作功能（上）》的姊妹篇，介绍“文件”主菜单其余功能的实现，包括新建文件、打开文件、文件打印与预览以及将文档导出为 PDF 格式。

文本编辑器“文件”主菜单功能实现的完整代码可在后台回复「**文本编辑器-文件**」获得下载链接。

本篇目录

运行环境：

win 10 + Qt 5.12.5 + Qt Creator 4.10

1. 新建文件

在 "textedit.h" 文件中添加“新建”文件的槽函数声明，代码如下：

```
1  private slots:
2      void fileNew();
```

本节介绍两种新建文件的方法，第一种方法的**步骤**是：

i. 判断文本编辑器中的文件是否保存，若未保存，则弹出保存提醒窗口；

ii. 若已保存，则将应用程序中文本编辑框中的内容先清除，并重新设置标题条显示，完成“新建”文件操作。

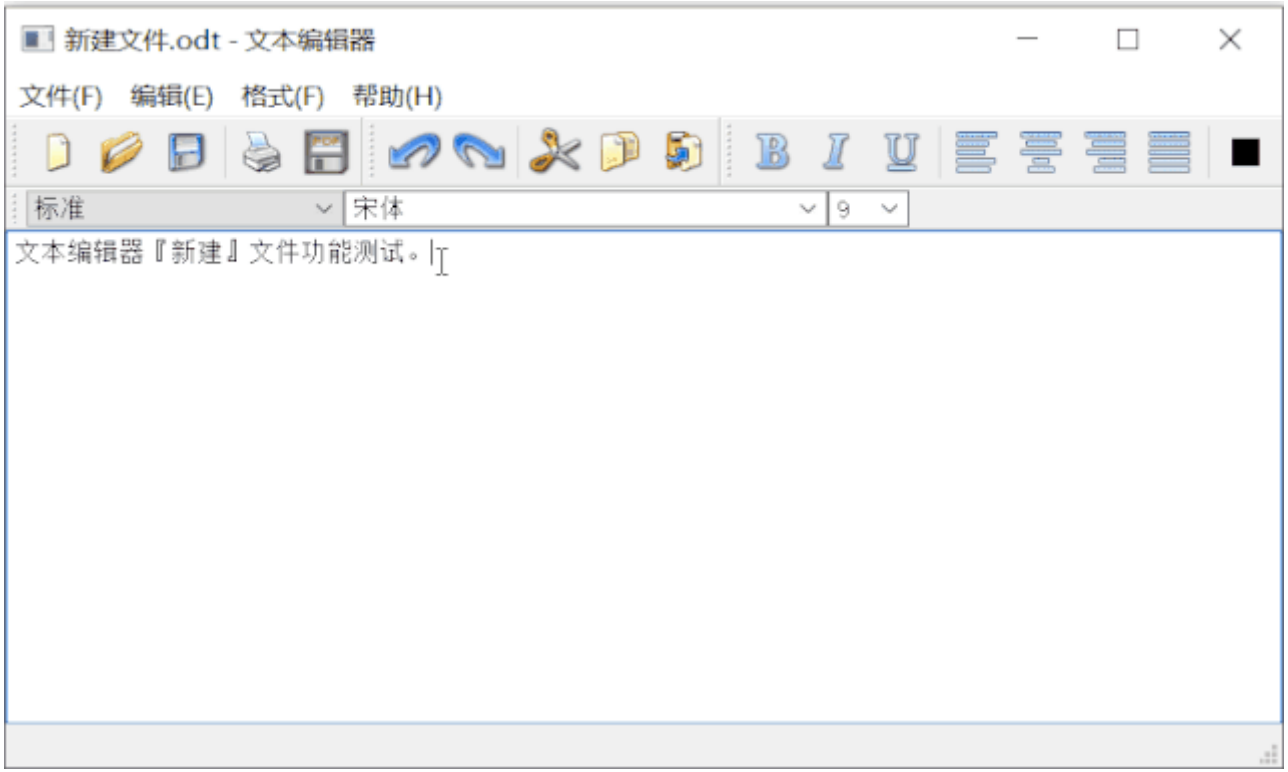
第一种方法的代码如下：

```
1  void TextEdit::fileNew()
2  {
3      if (maybeSave()) {
4          textEdit->clear();
5          setCurrentFileName(QString());
6      }
7  }
```

在`setupFileActions()`函数中，添加“新建”动作事件关联函数

```
1  connect(actionNew, &QAction::triggered, this, &TextEdit::fileNew);
```

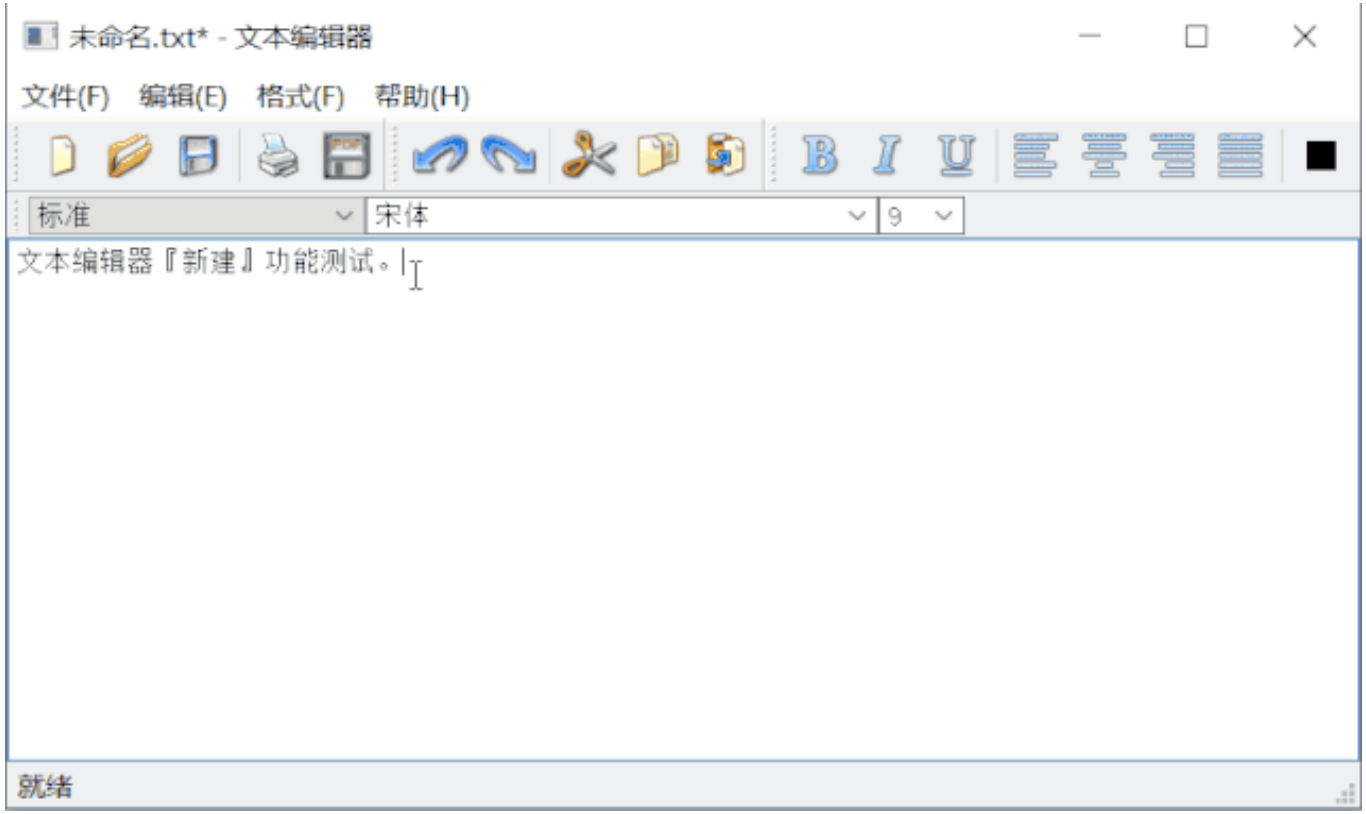
运行程序，单击『新建』按钮，察觉不出文本编辑器有变化，因为新建文件前后文本框中都是空白；输入任意文本并保存，单击『新建』按钮，得到“新建”文件功能的效果如下：



另一种新建文件的思路是不触动当前窗口，直接打开一个新的文本编辑器窗口，其代码如下：

```
1  void TextEdit::fileNew()
2  {
3      TextEdit *newTextEdit = new TextEdit;
4      newTextEdit->show();
5  }
```

运行程序，第二种“新建”文件功能的效果如下：



2. 打开文件

实现打开文件功能之前先要实现加载文件操作，在 "textedit.h" 文件中添加加载函数的声明：

```
1  private:
2      bool load(const QString &f);
```

加载文件操作的**步骤**是：

i. 判断加载文件是否存在，若不存在则返回 `false`；

ii. 以只读模式打开指定的文件，打开成功则将文本数据读入，否则返回 `false`；

iii. 根据输入文本数据的类型设定打开方式；

iv. 设置窗口标题。

加载文件函数的实现代码如下：

```
bool TextEdit::load(const QString &f)
{
    if (!QFile::exists(f))
        return false;
    QFile file(f);
    if (!file.open(QFile::ReadOnly))
        return false;

    QByteArray data = file.readAll();
    QTextCodec *codec = Qt::codecForHtml(data);
    QString str = codec->toUnicode(data);
    if (Qt::mightBeRichText(str)) {
        textEdit->setHtml(str); // 以 html 格式打开文本数据
    } else {
        str = QString::fromLocal8Bit(data);
        textEdit->setPlainText(str); // 以纯文本形式打开文本数据
    }

    setCurrentFileName(f);
    return true;
}
```

在加载文件操作中使用了 `QFile` 类对象，它可以打开指定的文件，并且与 `QByteArray` 类配合使用可以很方便的进行文件的读取与写入操作。此处还需添加头文件：

```
#include <QFile>
#include <QTextCodec>
```

接下来，在 "textedit.h" 文件中声明实现打开功能的函数

```
private slots:
    void fileOpen();
```

打开文件函数实现代码如下：

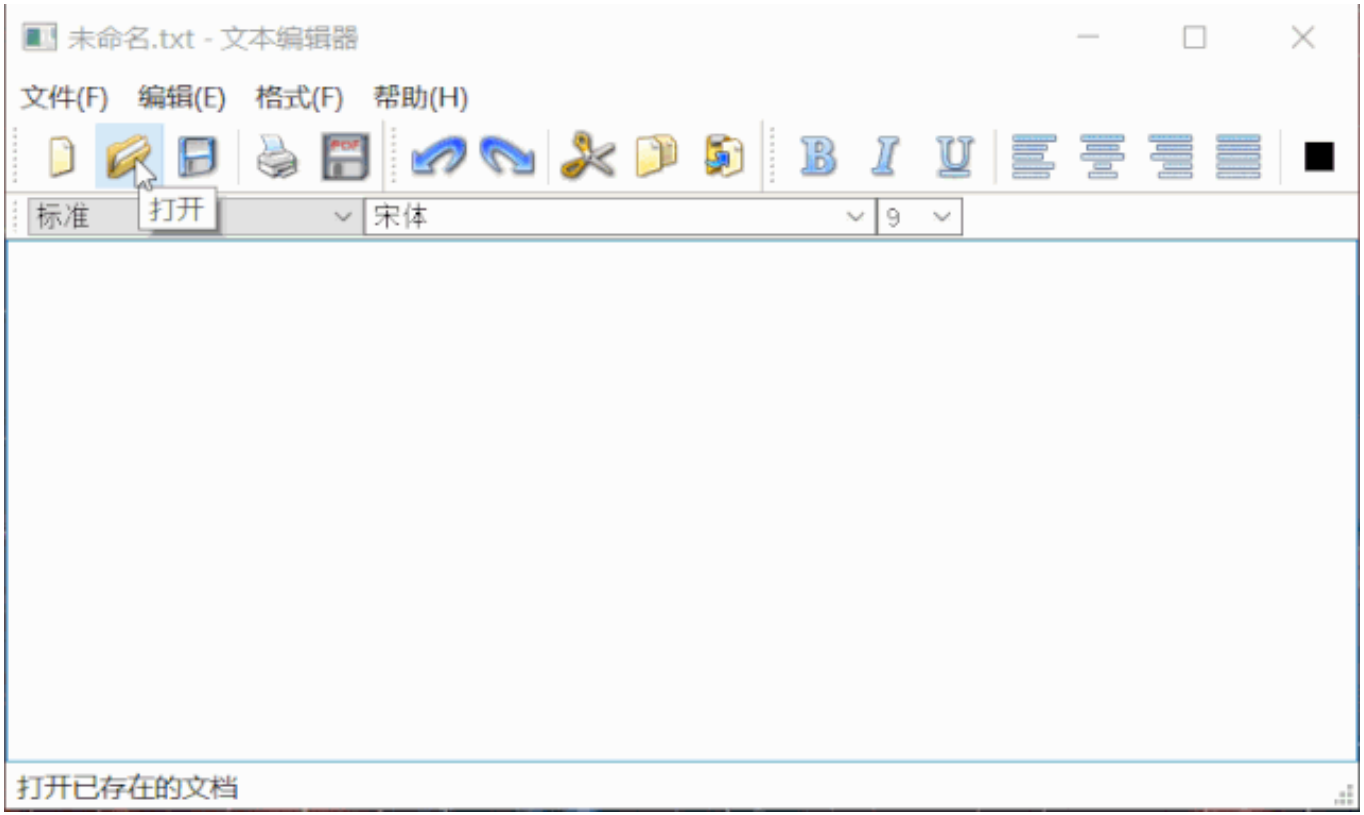
```
void TextEdit::fileOpen()
{
    QFileDialog fileDialog(this, tr("打开文件..."));
    fileDialog.setAcceptMode(QFileDialog::AcceptOpen);
    fileDialog.setFileMode(QFileDialog::ExistingFile);
    fileDialog.setMimeTypeFilters(QStringList() << "text/html" << "text/plain");
    if (fileDialog.exec() != QDialog::Accepted)
        return;
    const QString fn = fileDialog.selectedFiles().first();
    if (load(fn))
        statusBar()->showMessage(tr("已打开 \"%1\"").arg(QDir::toNativeSeparators(fn)));
    else
        statusBar()->showMessage(tr("未能打开 \"%1\"").arg(QDir::toNativeSeparators(fn)));
}
```

通过标准文件对话框打开文本文件，然后将文本文件名（包含文本文件的路径）传给文件加载函数，实现文本文件的打开功能。

在 `setupFileActions()` 函数中，添加“打开”动作事件关联函数

```
1   connect(actionOpen, &QAction::triggered, this, &TextEdit::fileOpen);
```

运行程序，“打开”功能的效果如下：



3. 文件打印与预览

Qt 5 将与打印有关的类放在 `QtPrintSupport` 模块中，因此，需要在项目配置文件 `TextEdit.pro` 中添加支持打印模块：

```
QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
qtHaveModule(printsupport): QT += printsupport # 支持打印模块

CONFIG += c++11
...
```

在 "textedit.h" 文件中添加“打印”“打印预览”函数的声明：

```
private slots:
    void filePrint();
    void filePrintPreview();
    void printPreview(QPrinter *printer);
```

其实现代码如下：

```
void TextEdit::filePrint()
{
    QPrinter printer(QPrinter::HighResolution); // 新建打印对象，并设置打印模式
    QPrintDialog *dlg = new QPrintDialog(&printer, this);
    if (dlg->exec() == QDialog::Accepted)
        textEdit->print(&printer);
    delete dlg;
}

void TextEdit::filePrintPreview()
{
    QPrinter printer(QPrinter::HighResolution);
    QPrintPreviewDialog preview(&printer, this); // 新建打印预览对话框
    connect(&preview, &QPrintPreviewDialog::paintRequested, this, &TextEdit::printPreview);
    preview.exec(); // 执行打印预览操作
}
```

```
17
18  void TextEdit::printPreview(QPrinter *printer)
19  {
20      textEdit->print(printer);
21  }
```

其中，

- `QPrinterDialog *dlg = new QPrintDialog(&printer, this);`
  Qt 提供的标准打印对话框，为打印机的使用提供了一种方便、规范的方法。

- `if (dlg->exec() == QDialog::Accepted)`
  判断打印对话框显示后用户是否单击『打印』按钮，若单击『打印』按钮，则相关打印属性将可以通过创建 `QPrinterDialog` 对象时使用的 `QPrinter` 对象获得；若单击『取消』按钮，则退出打印操作。

- `textEdit->print(printer);`
  可将文本编辑框中的文档打印到给定打印机的便捷打印功能。等效于直接在文档上调用 `print` 方法，除了此函数还支持 `QPrinter::Selection` 作为打印范围。

在这里还需要在源文件中引入头文件：

```
1  #include <QPrinter>
2  #include <QPrintDialog>
3  #include <QPrintPreviewDialog>
```

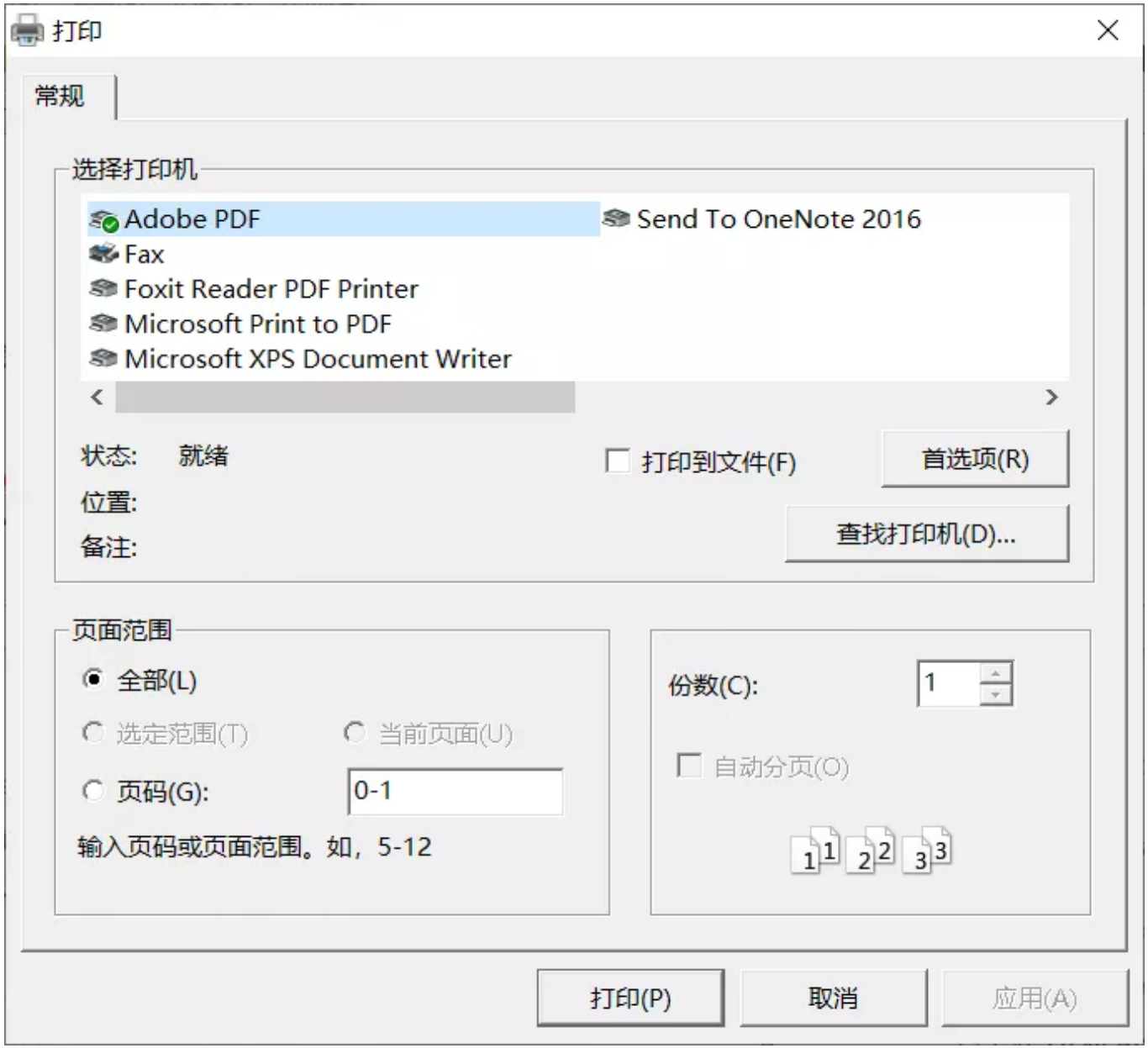
在 "textedit.h" 文件中添加 `QPrinter` 类的前置声明

```
1  class QPrinter;
```

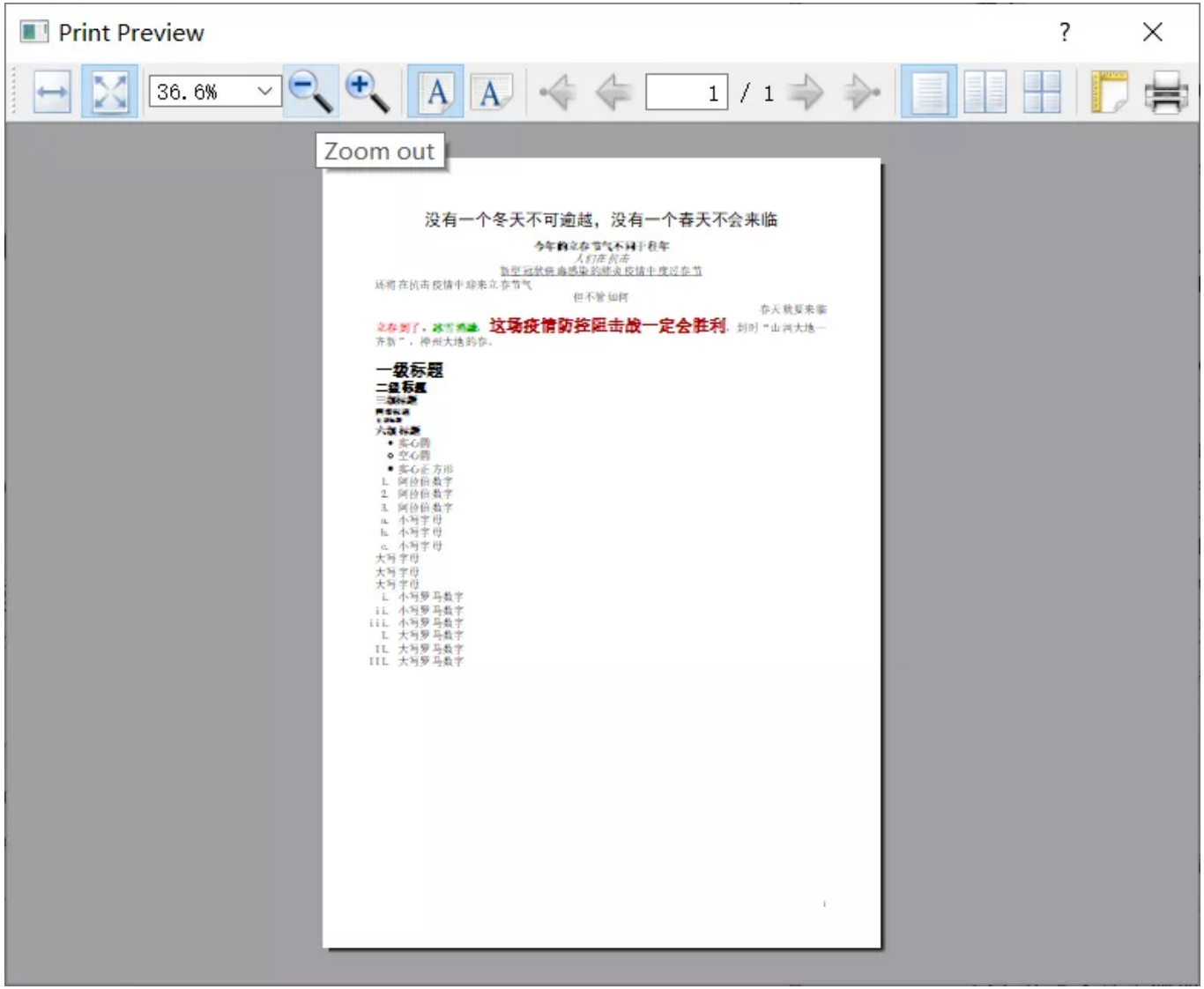
在 `setupFileActions()` 函数中，添加“打印”“打印预览”动作事件的关联函数

```
1  connect(actionPrint, &QAction::triggered, this, &TextEdit::filePrint);
2  connect(actionPrintPreview, &QAction::triggered, this, &TextEdit::filePrintPreview);
```

运行程序，点击『打印』按钮，则弹出如下打印对话框



点击『打印预览』按钮，则弹出如下打印预览对话框



4. 输出为 PDF 格式

在 "textedit.h" 文件中添加

```
private slots:
    void filePrintPdf();
```

在源文件中的实现函数：

```
void TextEdit::filePrintPdf()
{
    QFileDialog fileDialog(this, tr("输出为 PDF"));
    fileDialog.setAcceptMode(QFileDialog::AcceptSave);
    fileDialog.setMimeTypeFilters(QStringList("application/pdf"));
    fileDialog.setDefaultSuffix("pdf"); // 设置默认后缀
    if (fileDialog.exec() != QDialog::Accepted)
        return;
    QString fileName = fileDialog.selectedFiles().first();
    QPrinter printer(QPrinter::HighResolution);
    printer.setOutputFormat(QPrinter::PdfFormat);
    printer.setOutputFileName(fileName);
    textEdit->document()->print(&printer); // 打印
    statusBar()->showMessage(tr("输出为 \"%1\"")
                             .arg(QDir::toNativeSeparators(fileName)));
}
```

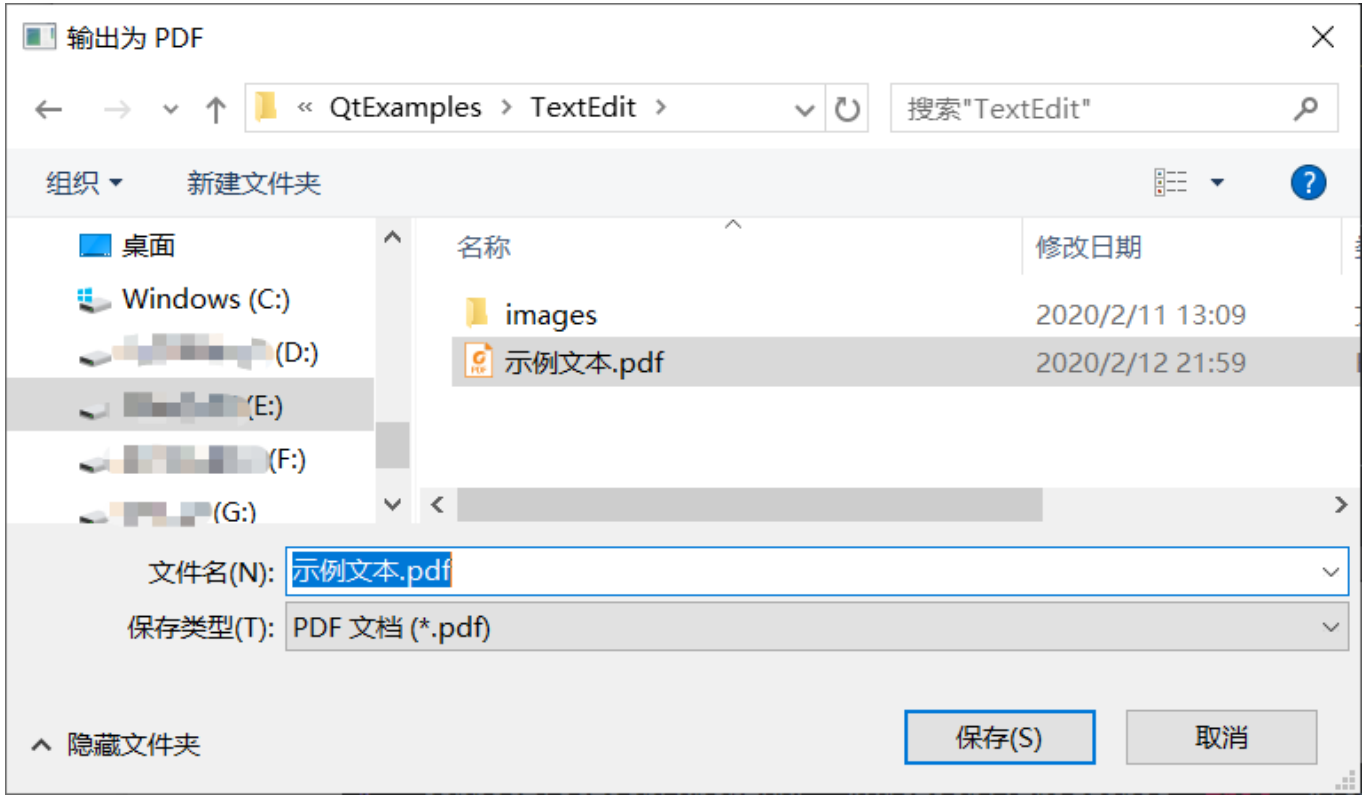
其中，

- `printer.setOutputFormat(QPrinter::PdfFormat);`
  将此打印机的输出格式设置为 `QPrinter::PdfFormat` 。

在 `setupFileActions()` 函数中，添加“输出为 PDF”动作事件的关联函数

```
connect(actionExportPDF, &QAction::triggered, this, &TextEdit::filePrintPdf);
```

运行程序，点击『输出为 PDF』按钮，则弹出如下“输出为 PDF”对话框



小结

本篇与上一篇《【文本编辑器】二、文件操作功能（上）》完成了文本编辑器“文件”主菜单中所有的文件操作功能。

文本编辑器“文件”主菜单功能实现的完整代码可在后台回复「**文本编辑器-文件**」获得下载链接。

相关阅读：

《【文本编辑器】二、文件操作功能（上）》

《【文本编辑器】一、界面设计》

《主要的窗体类和主窗体构成》

《Qt 模块简介》

《信号与槽》

《UI 文件设计与运行机制》