天津大学

《C++程序设计》实验报告

学　　院　国际工程师学院

专　　业　计算机技术

年　　级　2017

姓　　名　杨毅

2018 年　6 月　10 日

实验二 简易记事本

一、实验内容

使用 Qt 设计并实现简易记事本。记事本包括一个 Menu，三个 Toolbar 和一个文本编辑区。

Menu 主要包含四个下拉列表：File, Edit, Format 和 Help。File 下拉列表包含：New（新建）, Open（打开）, Save（保存）, Save As（另存）, Print（打印）, Print Preview（打印预览）和 Exit（退出）。Edit 下拉列表包含：Undo（撤销）, Redo（重做）, Cut（剪切）, Copy（复制）, Paste（粘贴）。Format 下拉列表包含：Font（字体）子列表和 Color（颜色），Font（字体）子列表里包含 Bold（加粗），Italic（斜体）和 Underline（下划线）。Help 下拉列表包含：About（关于）和 About Qt（关于 Qt）。

Toolbar 包含 fileToolBar, editToolBar 和 formatToolBar。

二、代码结构

实现过程中没有添加新的类，所有功能都在 MainWindow 主类中实现。添加的属性和方法如表 1 所示：

属性/方法	功能、作用
QMenuBar *menuBar	菜单栏
QMenu *menuFile	File 下拉列表
QMenu *menuEdit	Edit 下拉列表
QMenu *menuFormat	Format 下拉列表
QMenu *menuFont	Font 子列表
QMenu *menuHelp	Help 下拉列表
QAction *newAct	File 下拉列表中的 New 事件
QAction *openAct	File 下拉列表中的 Open 事件
QAction *saveAct	File 下拉列表中的 Save 事件
QAction *saveAsAct	File 下拉列表中的 Save As 事件
QAction *printAct	File 下拉列表中的 Print 事件
QAction *printPreviewAct	File 下拉列表中的 Print Preview 事件
QAction *exitAct	File 下拉列表中的 Exit 事件
QAction *undoAct	Edit 下拉列表中的 Undo 事件
QAction *redoAct	Edit 下拉列表中的 Redo 事件
QAction *cutAct	Edit 下拉列表中的 Cut 事件

QAction *copyAct	Edit 下拉列表中的 Copt 事件
QAction *pasteAct	Edit 下拉列表中的 Paste 事件
QAction *boldAct	Font 子列表中的 Bold 事件
QAction *italicAct	Font 子列表中的 Italic 事件
QAction *underlineAct	Font 子列表中的 Underline 事件
QAction *colorAct	Format 下拉列表中的 Color 事件
QAction *aboutAct	Help 下拉列表中的 About 事件
QAction *aboutQtAct	Help 下拉列表中的 About Qt 事件
QPlainTextEdit *textEdit	窗口中的文本编辑区
QString fName	文件名
bool isSaved	记录当前编辑文件是否被保存
void createFileActions()	创建 File 下拉列表中的所有事件
void createEditActions()	创建 Edit 下拉列表中的所有事件
void createFormatActions()	创建 Format 下拉列表中的所有事件
void createHelpActions()	创建 Help 下拉列表中的所有事件
void createMenus()	创建主菜单栏
void createToolBar()	创建 Tool Bar
void initialize()	初始化函数
void closeEvent(QCloseEvent *event)	关闭事件触发函数

表 1 属性方法及其功能简介

三、算法介绍

本次实验没有设计到较为复杂的算法，都是使用 Qt 自带的一些类，以及相关类对应的方法来实现对应的功能。主要用到以下几个类及其方法：

1、QPlainTextEdit 类，本实验文本编辑域类，主要用来实现文本编辑中用户需要的各种功能需求。用到的方法及其功能见表 3-1

方法	功能
mergeCurrentCharFormat()	设置当前 textEdit 的样式
clear()	清空 textEdit
setPlainText()	设置 textEdit 的文本
toPlainText()	获取 textEdit 的文本
height()	获取 textEdit 的高
width()	获取 textEdit 的宽
document()	主要用于文档的打应预览
undo()	撤销

redo()	重做
cut()	剪切
copy()	复制
paste()	粘贴

表 3-1 QPlainTextEdit 表

2、QMenuBar 类，主要用来实现菜单栏功能，用到的方法及其功能见表 3-2

方法	功能
addMenu()	为 menuBar 添加 menu

表 3-2 QMenuBar 表

3、QMenu 类，主要用来实现菜单栏每个菜单的功能，包括 File，Edit，Format 和 Help，以 Format 菜单为例，实现代码如下：

```
menuFormat = menuBar->addMenu(tr("&Format"));
menuFormat->addSeparator();
menuFont = menuFormat->addMenu(tr("&Font"));
menuFont->addAction(boldAct);
menuFont->addAction(italicAct);
menuFont->addAction(underlineAct);
menuFormat->addAction(colorAct);
```

用到的方法及其功能见表 3-3：

方法	功能
addAction()	为 menu 添加 Action
addSeparator()	为 menu 添加一个 separator action
addMenu()	为 separator action 添加子 menu

表 3-3 QMenu 类

4、QAction 类，主要用来实现每个菜单中的 Action，以 File 菜单下的 New(新建)为例，代码实现如下：

```
newAct = new QAction(QIcon("icons/new.png"),tr("&New"), this);
newAct->setShortcut(QKeySequence(tr("Ctrl+N")));
newAct->setToolTip("New");
newAct->setStatusTip(tr("Create a new file"));
connect(newAct, SIGNAL(triggered()), this, SLOT(newFile()));
```

用到的方法及其功能见表 3-4：

方法	功能
setShortcut()	添加快捷键
setToolTip()	添加工具栏提示
setStatusTip()	添加 status 提示

表 3-4 QAction 表

5、QTextcharFormat 类，主要用来设置 textEdit 中的字体的样式,以初始默认样式为例，代码实现如下：

```
void TextEdit::initialize() {
    QTextCharFormat fmt;
    fmt.setFontPointSize(15);
    fmt.setFontFamily("Times New Roman");
    textEdit->mergeCurrentCharFormat(fmt);
}
```

用到的方法及其功能见表 3-5

方法	功能
setFontPointSize()	设置字体大小
setFontFamily()	设置字体类型
setFontWeight()	设置字体是否加粗
setFontItalic()	设置字体是否为斜体
setFontUnderline()	设置字体是否有下划线
setForeground()	设置字体颜色

表 3-5 QTextcharFormat 类

四、UML 图

用例图如图 4-1 所示，为用户提供的功能主要是菜单栏中的几个菜单。包括文本编辑，File 菜单中的新建，打开，保存，另存为，打印，预览和退出。Edit 菜单中的撤销，重做，剪切，复制，粘贴。Format 菜单中的加粗，斜体，下划线和颜色设置。Help 菜单中的关于和关于 Qt。类图如图 4-2 所示。

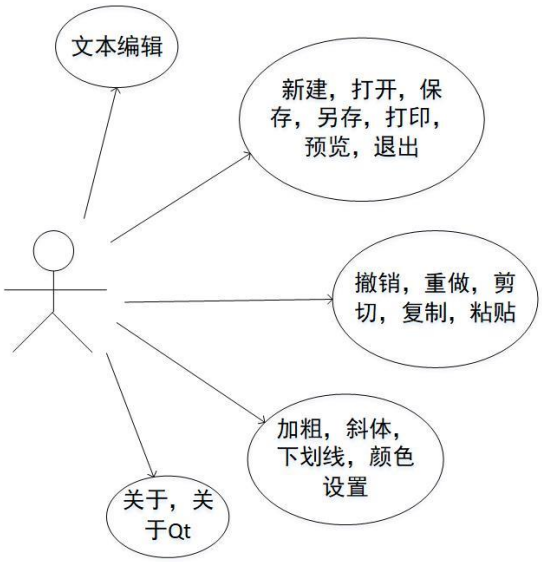

图 4-1 用例图

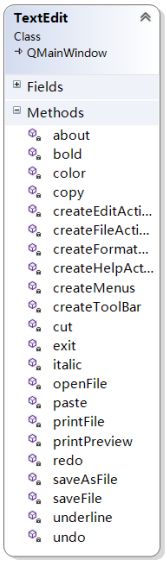

图 4-2 类图

五、信号和槽

1.信号与槽函数的连接

（1）File Menu 中的 connect：

```
connect(newAct, SIGNAL(triggered()), this, SLOT(newFile()));
connect(openAct, SIGNAL(triggered()), this, SLOT(openFile()));
connect(saveAct, SIGNAL(triggered()), this, SLOT(saveFile()));
connect(saveAsAct, SIGNAL(triggered()), this, SLOT(saveAsFile()));
connect(printAct, SIGNAL(triggered()), this, SLOT(printFile()));
connect(printPreviewAct, SIGNAL(triggered()), this, SLOT(printPreview()));
connect(exitAct, SIGNAL(triggered()), this, SLOT(exit()));
```

（2）Eidt Menu 中的 connect：

```
connect(undoAct, SIGNAL(triggered()), this, SLOT(undo()));
connect(redoAct, SIGNAL(triggered()), this, SLOT(redo()));
connect(cutAct, SIGNAL(triggered()), this, SLOT(cut()));
connect(copyAct, SIGNAL(triggered()), this, SLOT(copy()));
connect(pasteAct, SIGNAL(triggered()), this, SLOT(paste()));
```

（3）Format Menu 中的 connect：

```
connect(boldAct, SIGNAL(triggered()), this, SLOT(bold()));
connect(italicAct, SIGNAL(triggered()), this, SLOT(italic()));
connect(underlineAct, SIGNAL(triggered()), this, SLOT(underline()));
connect(colorAct, SIGNAL(triggered()), this, SLOT(color()));
```

（4）Help Menu 中的 connect：

```
connect(aboutAct, SIGNAL(triggered()), this, SLOT(about()));
connect(aboutQtAct, SIGNAL(triggered()), qApp, SLOT(aboutQt()));
```

（5）textEdit 中的 connect：

```
connect(textEdit, SIGNAL(textChanged()), this, SLOT(textChange()));
connect(textEdit, SIGNAL(selectionChanged()), this, SLOT(selectChange()));
```

（6）打印预览 connect：

```
connect(&preview, SIGNAL(paintRequested(QPrinter*)), this,
SLOT(printPreviewSlot(QPrinter*)));
```

2.槽函数执行功能

槽函数	执行功能
void newFile()	新建文件
void openFile()	打开文件
void saveFile()	保存文件

void saveAsFile()	另存为
void printFile()	打印文件
void printPreview()	打印预览
int exit()	退出
void undo()	撤销
void redo()	重做
void cut()	剪切
void copy()	复制
void paste()	粘贴
void bold()	加粗
void italic()	斜体
void underline()	下划线
void color()	设置颜色
void about()	关于
void printPreviewSlot(QPrinter *printer)	打印预览中文档，纸张大小等设置
void textChange()	textEdit 中文档被修改后调用
void selectChange()	textEdit 选中 text 改变调用

六、实验亮点

1.点击窗口关闭按钮或者 exit：

点击窗口关闭按钮或者 exit 时，若文档未保存，则跳出提示框，询问用户是否保存。若文档已经保存，或者文档未进行修改则不会跳出提示框，直接关闭。效果如图 6-1 所示。

具体实现方法如下：

```
int TextEdit::exit() {
    if (!isSaved) {
        int signal = QMessageBox::warning(this, tr("Warning"),
                                          tr("Do you want to save the changes?"),
                                          QMessageBox::Save | QMessageBox::Discard |
                                          QMessageBox::Cancel, QMessageBox::Save);
        switch (signal) {
        case QMessageBox::Save:
            saveFile();
            break;
        case QMessageBox::Discard:
            return 0;
```

```
            QCoreApplication::exit(0);
        case QMessageBox::Cancel:
            break;
        }
        return 1;
    }
    return 0;
}

void TextEdit::closeEvent(QCloseEvent *event) {
    int res = exit();
    if (res) event->ignore();
}
```

当点击了 exit action 之后，调用 exit()槽函数。判断当前文件是否已保存。若文件未保存，则弹出警告 QMessageBox 询问用户是否要保存当前文件，共有三个选择按钮：Save（保存当前文件），Discard（不保存，退出），Cancle（取消，返回当前文件）。

当点击了窗口的关闭按钮后，调用 closeEvent 函数，函数中调用 exit()函数，判断如果用户选择的是 Cancle，则取消关闭当前窗口。

当文档进行修改后，如上文第五点中 textEdit 中的 connect，textEdit 文档发生修改会调用 textChange()槽函数，修改文件状态为为保存即可。（修改 isSaved 值为 false）

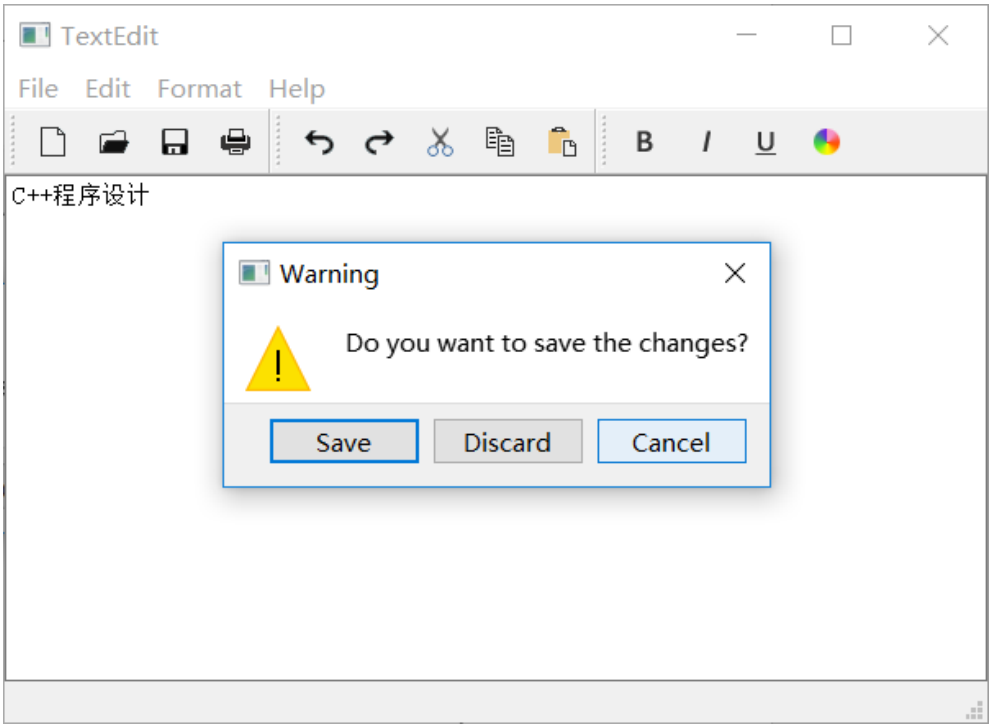

图 6-1 exit 和退出提示框

2.加粗，斜体，下划线功能：

加粗，斜体，下划线功能的实现效果完全仿照 Ofiice Word。ToolBar 中的加粗，斜体，下划线图标的状态（checked 状态或者非 checked 状态）完全取决于鼠标光标所处位置处的字体。如果是选中状态的话，取决于选中文本结尾处字体的格式。效果图如 6-2 到 6-7。

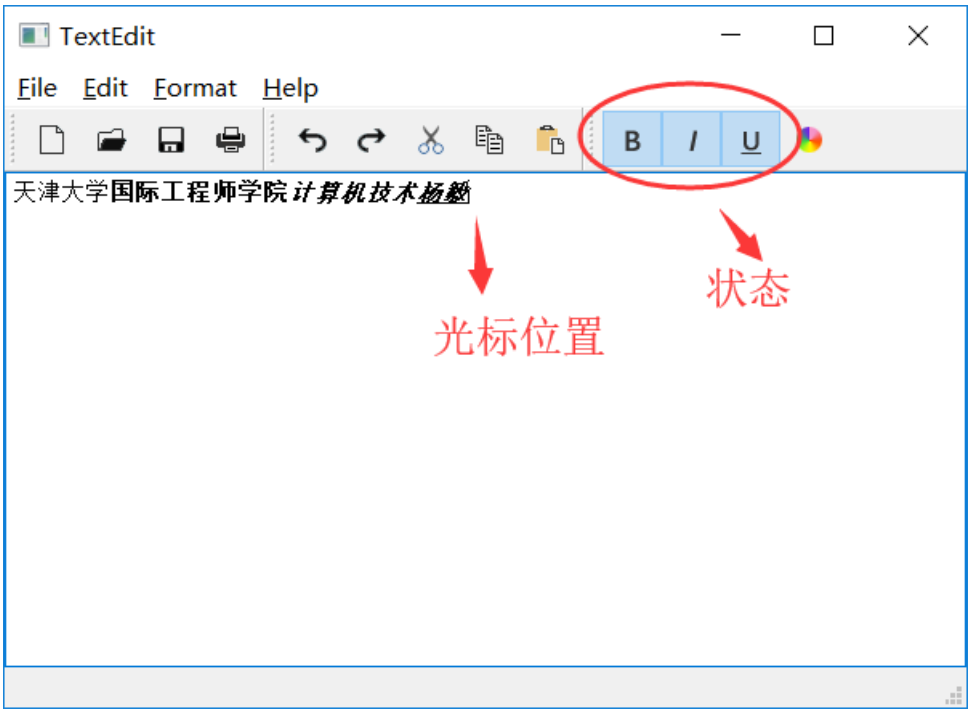

图 6-2

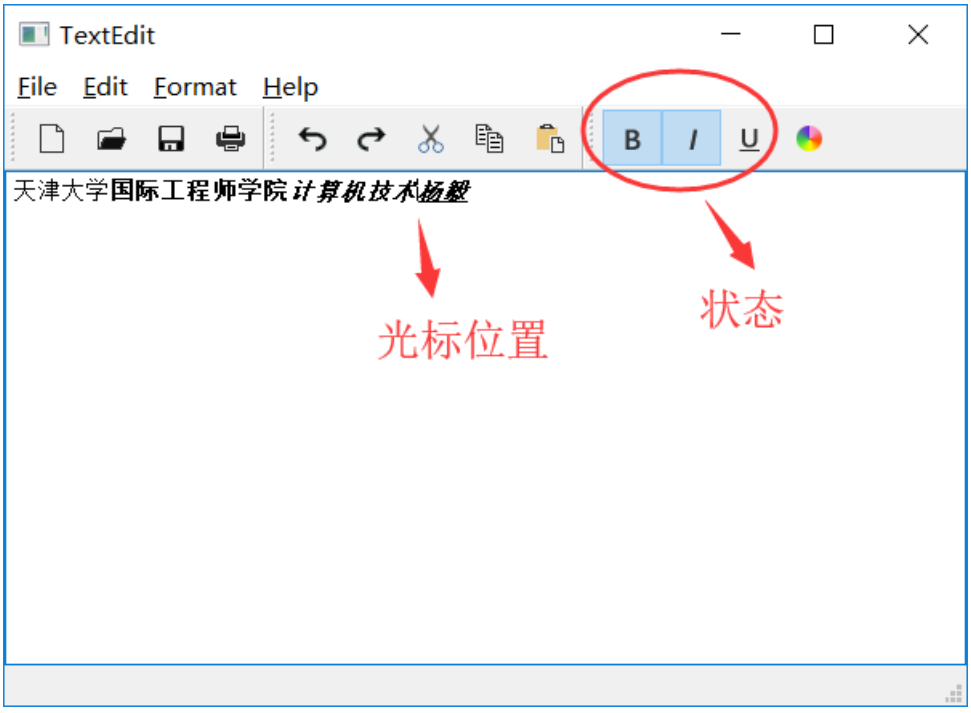

图 6-3

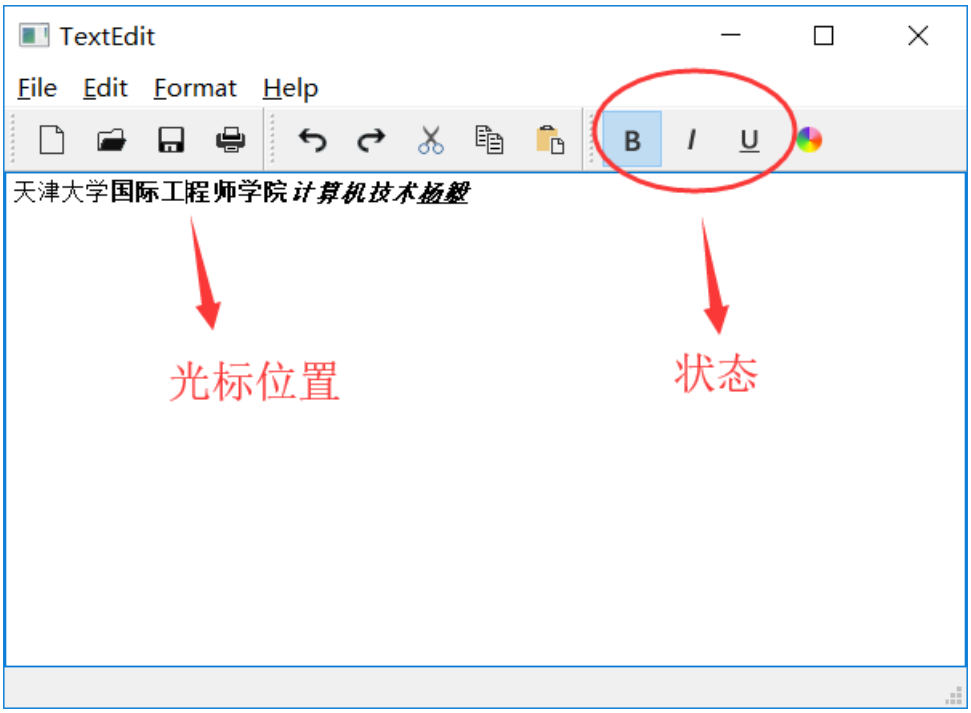

图 6-4

光标所处位置处前一个字“国”为加粗状态。所以当前 B 图标为 checked 状态。I 图标和 U 图标为非 checked 状态。

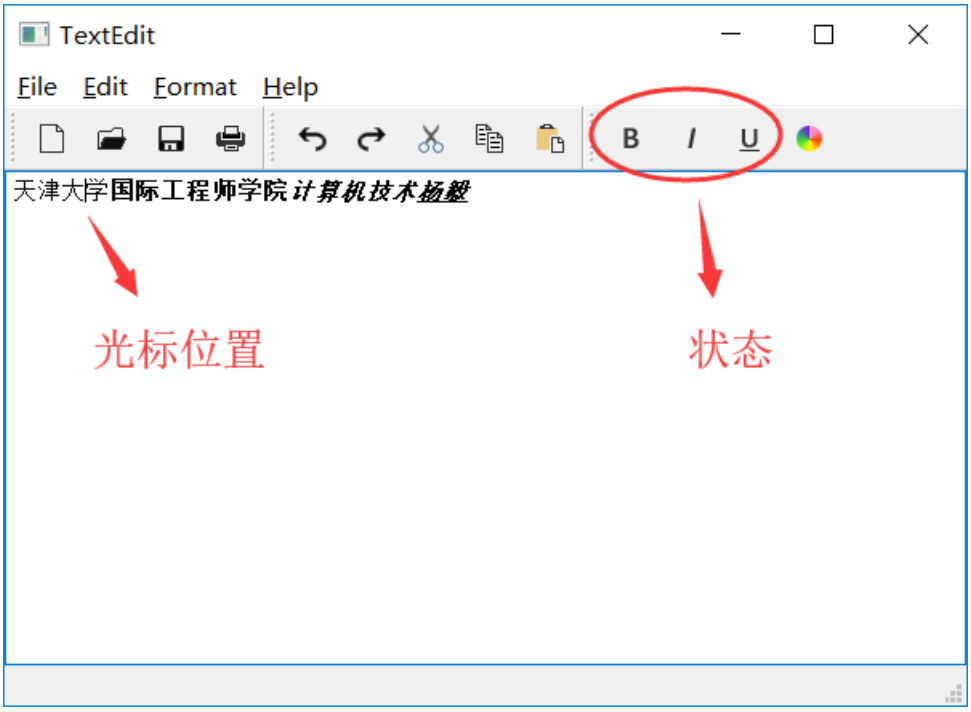

图 6-5

光标所处位置处前一个字为“津”，该字此时为 normal 状态，所以三个图标均为非 checked 状态。

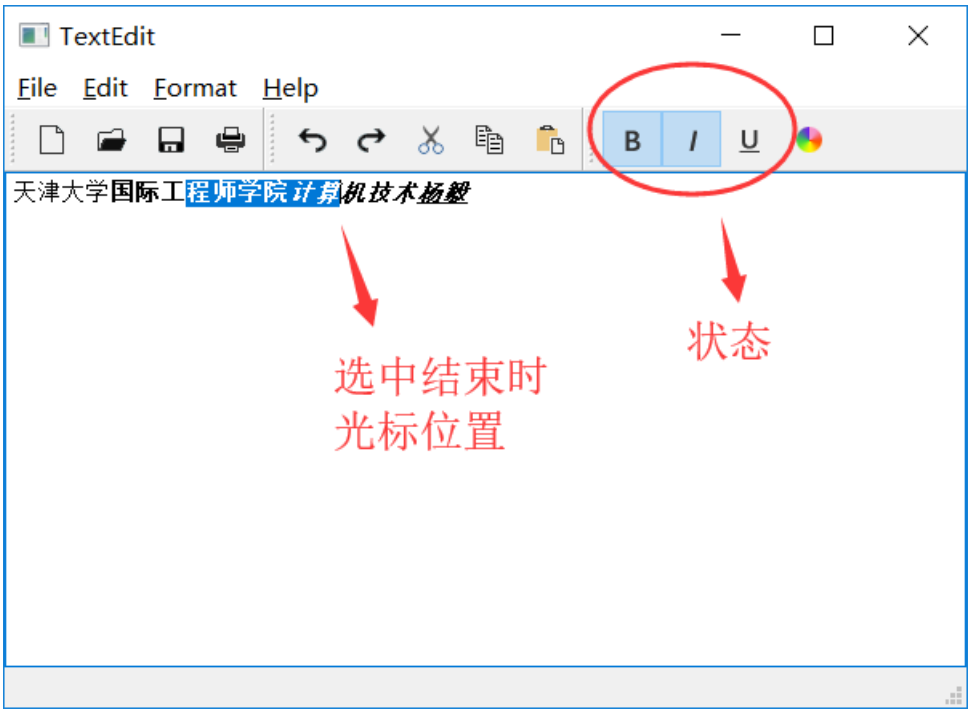

图 6-6

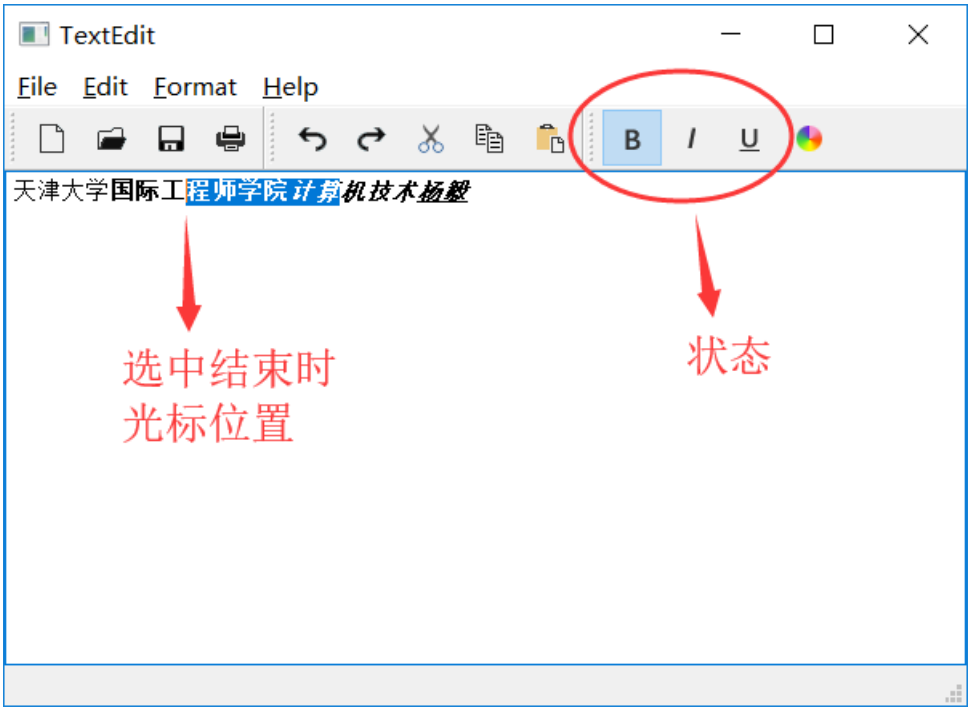

图 6-7

具体实现方法代码如下：

```
connect(textEdit, SIGNAL(selectionChanged()), this, SLOT(selectChange()));
void TextEdit::selectChange() {
    QTextCharFormat fmt = textEdit->currentCharFormat();
    int weight = fmt.fontWeight();
    bool italic = fmt.fontItalic();
```

```
    bool underline = fmt.fontUnderline();
    if (weight == 75) boldAct->setChecked(true);
    else boldAct->setChecked(false);
    if (italic) italicAct->setChecked(true);
    else italicAct->setChecked(false);
    if (underline) underlineAct->setChecked(true);
    else underlineAct->setChecked(false);
}
```

首先，当光标所处位置发生变化时，textEdit 会发出 selectionChanged()信号，然后与之对应的槽函数 selectChange()进行接收。在该函数中，首先获取光标所处位置处文字的 format。然后根据这个 format 对“加粗”，“斜体”，“下划线”的 checked 状态进行设置。只需要设置成对立状态就可以了。

七、总结讨论

首先感谢孙老师本学期的耐心教导。经过本次实验，我对 Qt 的使用更为熟练。此次实验的所有控件都是通过代码实现的，所以是我使用 Qt 以来的一次新的尝试，之前使用 Qt 控件都是通过 Qt designer 完成的。较实验一而言，本次实验更偏重的是对 Qt 一些新的类的了解与应用。比如 QMenu 类，QAction 类，QPrinter 类等。让我认识了很多 Qt 自带的新的类，更加体会到 Qt 的强大之处。更重要的是，对 C++语言更为熟悉，编程能力进一步加强。