OOP 报告

——文件阅读器

目录

1.需求分析

1.1 平台介绍

Windows 10。Windows 10 是美国微软公司所研发的新一代跨平台及设备应用的操作系统。

1.2 IDE 介绍

QT creator。Qt Creator 是跨平台的 Qt IDE，Qt Creator 是 Qt 被 Nokia 收购后推出的一款新的轻量级集成开发环境（IDE）。此 IDE 能够跨平台运行，支持的系统包括 Linux、Mac OS X 以及 Windows。

1.3Complier 介绍

Gcc。GNU 编译器套件（GNU Compiler Collection）包括 C、C++、Objective-C、Fortran、Java、Ada 和 Go 语言的前端，也包括了这些语言的库（如 libstdc++、libgcj）。

1.4 功能

显示文本文件，文件的内容以 ASCII 字符格式和十六进制格式显示，支持以下功能：

1.4.1 文件的打开与关闭；

调用 void openfile(char filename)函数打开文件，这里 filename 是要打开的文件名。文件打开函数执行后，根据相关参数的值判断文件是否顺利打开。如果文件打开失败，则返回一个错误提示。如果文件打开成功，则以 ASCII 显示文件内容。

调用 void closefile()函数关闭文件，并清除 buf 中保留的信息。

1.4.2 文件的翻页；

调用 void turnpage()函数执行文件的翻页的操作，然后读取按键信号进行判断，PgUp 信号对应文件向上翻页操作，如果已经是首页，则不再执行；PgDn 信号对应文件向下翻页操作，如果已经达到文件的末页，则不再执行。

1.4.3 文件内容向上或向下移动一行；

调用 void moveline()函数执行文件内容的上下移动操作，读取上、下按键信号，↑信号对应文件向上移动一行操作，如果已经是文件的第一行，则不再执行；↓信号对应文件向下移动一行操作，如果已经达到文件的最后一行，则不再执行。

1.4.4 在 ASCII 或十六进制显示方式之间切换；

void asc2hex()与 void hex2asc()分别实现 ASCII 到十六进制与十六进制到 ASCII 的转换功能，通过循环并移动指针，逐字地对内容进行转换。

2. 关键设计思路或方法

2.1 文件的打开：

通过槽传递信号，信号包含文件路径和文件名信息，调用 MainWindow 的成员函数 void open()。成员函数 open()中判断文件名是否为空，非空则传递参数给 MainWindow 的公共函数 loadFile()，loadFile()用 QTextStream::setCodec()来设置合适的编码读取文本信息，通过调用 Qt 的文本编辑类的成员函数 QTextEdit::setPlainText() 来显示。为了避免文本被编辑，通过调用 QTextEdit::setReadOnly()函数设置文本内容不可编辑。

2.2 文件的关闭：

通过槽传递信号，调用 MainWindow 的成员函数 void close()。close()通过调用 QTextEdit::setPlainText(NULL)设置文本内容为空，此外再调整标题栏以及状态栏的信息，达到关闭文件的效果。

2.3 文件的翻页

因为启用了 Qt 库中的 QTextEdit 类作为显示控件，而 QTextEdit 类作为显示控件即在文本只读的情况下支持 PageUp、PageDown 的按键操作，所以通过这两个按键就可以实现文件的翻页操作，不再需要对按键进行重载。

2.4 文件内容向上或向下移动一行

跟 2.文件的翻页原因一样，QTextEdit 类作为显示控件即在文本只读的情况下支持 Up、Down 的按键操作，可以实现文件内容向上或向下移动一行的操作。

2.5 在 ASCII 或十六进制显示方式之间切换

通过槽传递信号，调用 MainWindow 的成员函数 void hex()。hex()先进行判断当前的文本内容是否十六进制显示，判断可以通过确认 bool 变量 ishex 的真假来实现。

如果当前文本内容是 ASCII 字符格式，将包含文本内容的字符串类 QString content 转换为字符数组形式，利用 QChar::unicode()逐个获取每个字符的 Unicode 值，再通过 QString 类的 arg()函数将获得的 Unicode 值转换为指定格式的十六进制显示，然后附加到用于保存十六进制信息的字符串类上。接着调用 QString::insert()函数，在十六进制中间插入一些地址信息、回车、换行，改善十六进制显示的格式。最后通过调用 QTextEdit::setPlainText()显示转换得到的十六

进制信息，设置 ishex 为真，实现 ASCII 到十六进制的转换。

如果当前文本内容是十六进制格式，设置 ishex 为假，调用 QTextEdit::setPlainText()显示 content 中保存的未转成十六进制的 ASCII 信息，实现十六进制到 ASCII 的转换。

2.6 文本字体格式的设置

利用 QFont 类设置字体，保存文本的字体信息。通过槽传递信号，调用不同的改变格式的函数，比如加粗、斜体、下划线、字体、大小等等，用 QFont 类中的设置函数，改变字体信息。最后调用 QTextEdit 类中的 setFont()函数，根据前面得到的字体信息设置文本字体格式。

3.详细设计

3.1 文件操作模块设计说明

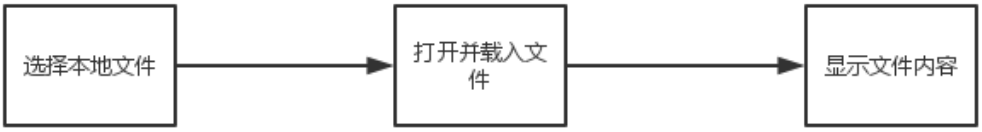
3.1.1 模块描述

此模块是用户用来打开文件所需的功能模块，实现的功能有打开文件和关闭文件。

3.1.2 功能

IPO 图如下：

输入　　处理　　输出



3.1.3 性能

用户可以打开保存在本地的任一文件，以 Unicode 格式显示，随后可以转换为十六进制显示。最后能关闭文件

3.1.4 输入项

名称	标识	类型和格式	输入方式
本地文件	file_Name	QString	菜单栏点击操作
已打开的文件	cur_File	QString	菜单栏点击操作

3.1.5 输出项

名称	标识	类型和格式	输出方式
文件内容	Typeface	QString	文本框

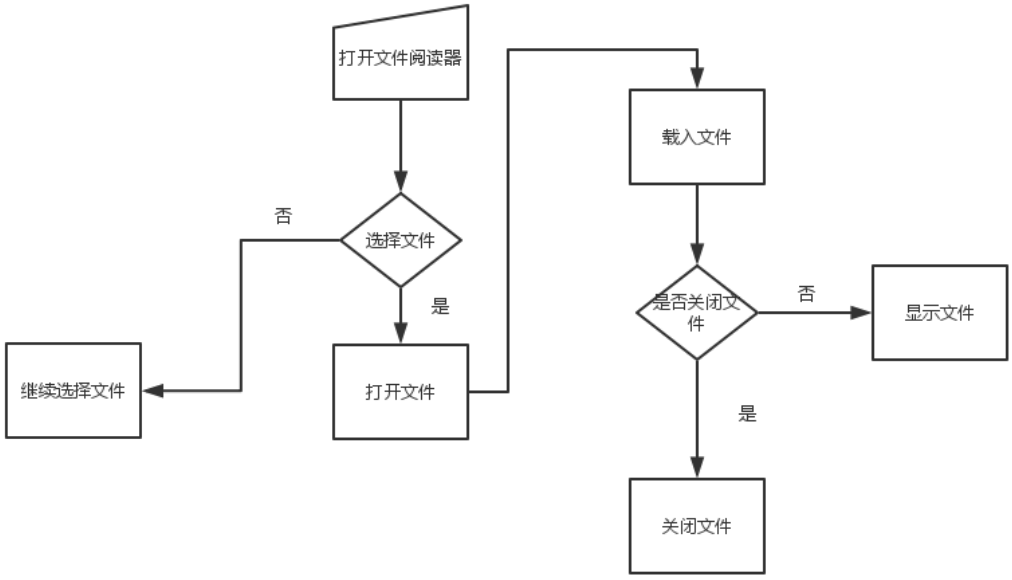
3.1.6 设计方法（算法）

当用户点击“文本阅读器”这一应用程序时，触发 OpenFile 函数，打开文件后调用 LoadFile 函数载入文件，最后调用 SetCurrentFile 函数设置当前文件打开的一些属性。

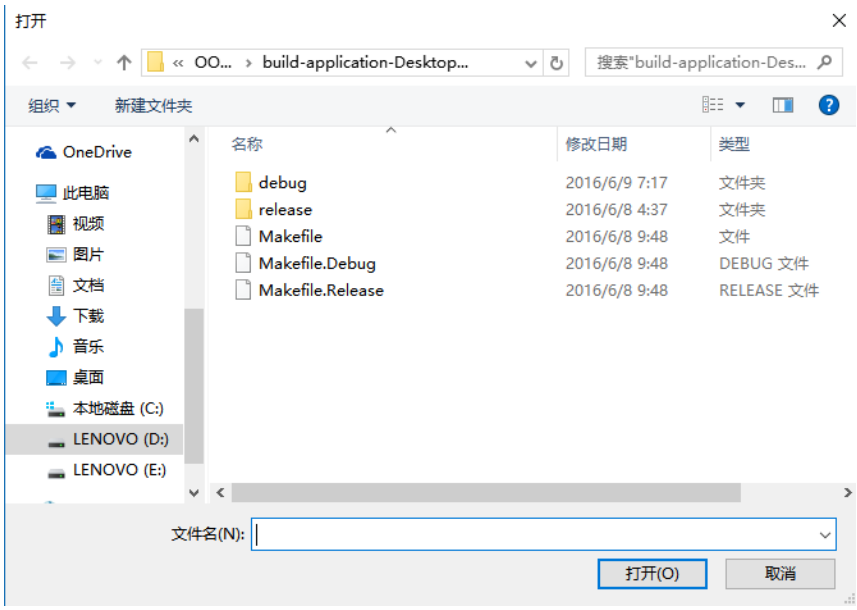
```
void MainWindow::OpenFile()//open the file we choose
{
    QString file_Name = QFileDialog::getOpenFileName(this);
    if (!file_Name.isEmpty())
        LoadFile(file_Name);
}

void MainWindow::CloseFile()//close current file
{
    textEdit->setPlainText(NULL);
    SetCurrentFile("文本阅读器");
    statusBar()->showMessage(tr("就绪"), 2000);
}
void MainWindow::LoadFile(const QString &file_Name)//load file
{
    QFile file(file_Name);
    if (!file.open(QFile::ReadOnly)) {
        QMessageBox::warning(this, tr("文本阅读器"),
                             tr("无法读取文件 %1:\n%2.")
                             .arg(QDir::toNativeSeparators(file_Name),
file.errorString()));
        return;
    }

    QByteArray data = file.readAll();
    QTextStream in(&data);
    in.setAutoDetectUnicode(false);
    in.setCodec("utf-8");
    content = in.readAll();
#ifndef QT_NO_CURSOR
    QApplication::setOverrideCursor(Qt::WaitCursor);
#endif
    textEdit->setPlainText(content);
#ifndef QT_NO_CURSOR
    QApplication::restoreOverrideCursor();
#endif
    isHex=false;
    SetCurrentFile(file_Name);
```

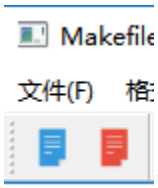
```
        statusBar()->showMessage(tr("文件已打开"), 2000);
        file.close();
}
```

3.1.7 流程逻辑



3.1.8 接口





蓝色表示打开文件，红色表示关闭文件。

3.1.9 测试计划

单独测试。
测试要求可以打开任意本地文件，并能关闭。

输入数据	预期结果
本地文件	成功打开
已打开的本地文件	成功关闭

3.2 字体设置模块设计说明

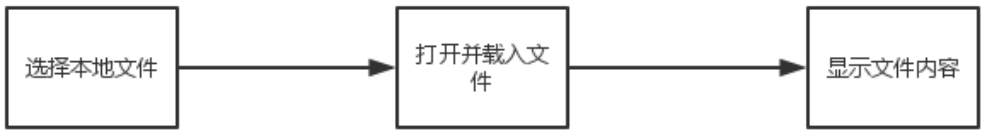
3.2.1 模块描述

此模块是用户打开文件后对文件的字体进行设置的一个功能模块，可以设置是否加粗、是否斜体、是否加下划线，另外可以设置字体种类、字体大小和字体颜色，此外背景颜色也可以设置。

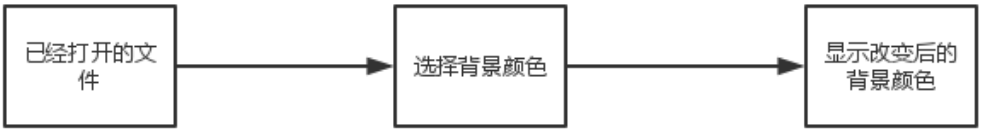
3.2.2 功能

IPO 图如下：

输入　　处理　　输出



输入　　处理　　输出



3.2.3 性能

用户可以设置加粗、斜体、加下划线、字体种类、字体大小、字体颜色、背景颜色。

3.2.4 输入项

名称	标识	类型和格式	输入方式
字体	font	QFont	菜单栏点击操作
背景	BackGroundColorIcon	QPixmap	图像

3.2.5 输出项

名称	标识	类型和格式	输出方式
字体	font	QFont	文本框
背景颜色	BackGroundColorIcon	QPixmap	图像

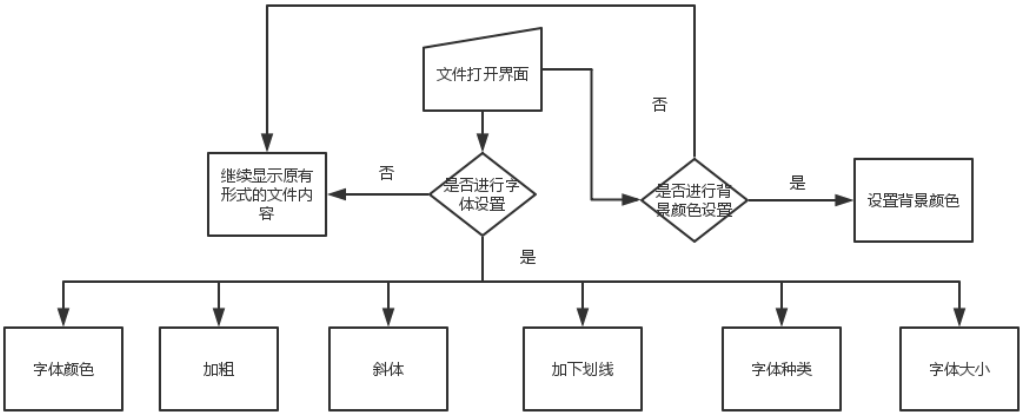
3.2.6 设计方法（算法）

当用户点击相应的字体操作选项时，程序调用相应的字体操作函数进行字体设置；当用户点击背景颜色设定时，程序调用 BackGroundColor 函数进行背景颜色设置工作。

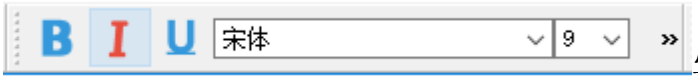
```
void MainWindow::BoldTypeface()//bold the font
{
    font.QFont::setBold(!font.QFont::bold());
    textEdit->setFont(font);
}

void MainWindow::ItaliseTypeface()//italic the font
{
    font.QFont::setItalic(!font.QFont::italic());
```

```
    textEdit->setFont(font);
}

void MainWindow::UnderlineTypeface()//underline the font
{
    font.QFont::setUnderline(!font.QFont::underline());
    textEdit->setFont(font);
}

void MainWindow::FontBox(const QString &family)
{
    font.QFont::setFamily(family);//change family of Font
    textEdit->setFont(font);
}

void MainWindow::SizeTypeface(const QString &ptr)
{
    qreal pointSizeF = ptr.toFloat();
    if (pointSizeF > 0) {
        font.QFont::setPointSizeF(pointSizeF);
        textEdit->setFont(font);
    }
}
void MainWindow::SetFontcolor()//control the color of font
{
    QColor fontcol = QColorDialog::getColor(Qt::white, this);
    QPalette fontpal = palette();
    fontpal.setColor (QPalette::Text,fontcol);
    setPalette (fontpal);
    QPixmap FontColorIcon(16, 16);
    FontColorIcon.fill(fontcol);
    fontcolorAct->setIcon(FontColorIcon);
}

void MainWindow::BackGroundColor()//background color control
{
    QColor BackGroundcol = QColorDialog::getColor(Qt::white, this);
    QPalette BackGroundpal = palette();
    BackGroundpal.setColor (QPalette::Base,BackGroundcol);
    setPalette (BackGroundpal);
    QPixmap BackGroundgColorIcon(16, 16);
    BackGroundgColorIcon.fill(BackGroundcol);
    bgcolorAct->setIcon(BackGroundgColorIcon);
}
```

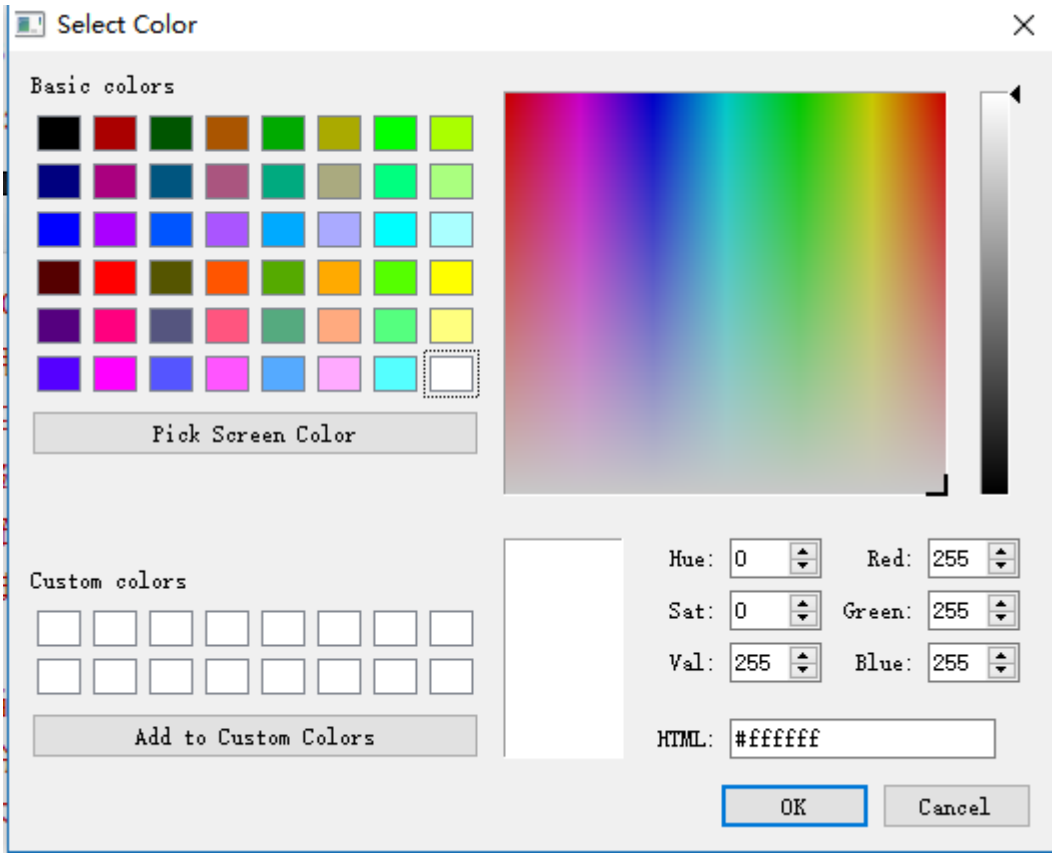
3.2.7 流程逻辑



3.2.8 接口



字体设置功能



背景颜色设置

3.2.9 测试计划

与文件操作模块集成测试。
测试要求可以设置任意字体的格式、任意颜色的背景颜色。

输入数据	预期结果
字体	成功设置字体
背景	成功设置背景颜色

3.3 文件格式设置模块设计说明

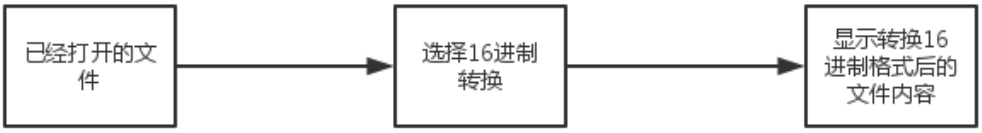
3.3.1 模块描述

此模块是用户打开文件后对文件的格式进行设置的一个功能模块，可以设置是否转换为 16 进制显示。

3.3.2 功能

IPO 图如下：

输入　　处理　　输出



3.3.3 性能

用户可以设置 16 进制格式显示文件内容，另外文件内容格式默认为 Unicode 格式。

3.3.4 输入项

名称	标识	类型和格式	输入方式
文件	file_Name	QString	菜单栏点击操作

3.3.5 输出项

名称	标识	类型和格式	输出方式
文件内容	font	QFont	文本框

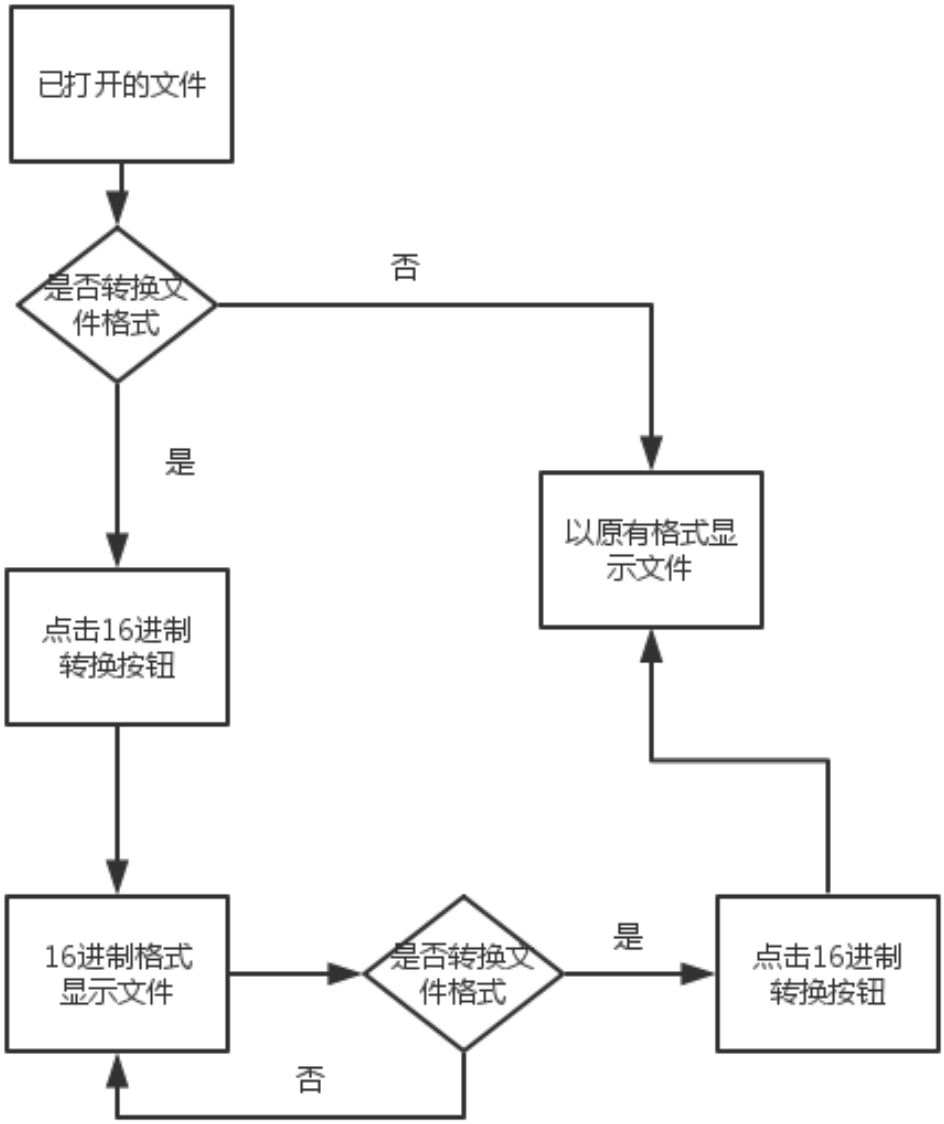
3.3.6 设计方法（算法）

当用户点击相应的 16 进制转换选项时，程序调用 HexTransfer 函数进行文件格式设置。

```
void MainWindow::HexTransfer()
{
    //When show text,we will add some extra void line to
make it clear
    //hence the result may differ initial file littlely.
    //Fomat below:
    //00000000h: xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx\n
    if(!isHex){
        QChar *Char = new QChar[content.count()];
        Char = content.data();
        QString UnicodeFormat;
```

```
       for(int i=0;i < content.count();i++){ //change the
char into unicode

UnicodeFormat.append(QString("%1").arg(Char[i].unicode()
,2,16,QChar('0')));
       }
       int j=0;
       UnicodeFormat.insert(0,QString("%1h:
").arg(j++,8,16,QChar('0')));//first line number;
       for(int               i=LINESIGN;i               <
UnicodeFormat.count();i++){//add the line number
          if(0 == (i+1)%LINEMAXCOUNT){
             UnicodeFormat.insert(i,QString("\n%1h:
").arg(j++,8,16,QChar('0')));
             i += SINGLEINSERT ;
          }else               if(0==((i+1)%LINEMAXCOUNT-
LINESIGN)%SETSPACE){
             UnicodeFormat.insert(i,QChar(' '));
          }
       }
       textEdit->setPlainText(UnicodeFormat);//show
unicode
       isHex=true;
    }else{
       textEdit->setPlainText(content);//show  the  plain
text
       isHex=false;
    }
}
```

3.3.7 流程逻辑



3.3.8 接口

 点击此按钮可以从初始的 Unicode 格式转换为 16 进制格式。

3.3.9 测试计划

与文件操作模块集成测试。
测试要求可以使打开文件的格式在 Unicode 格式和 16 进制格式之间转换。

输入数据	预期结果
文件	成功设置文件格式