基于Qt的软件汉化界面技术的研究

王家华,程顺顺
(西安石油大学 计算机学院,陕西 西安 710065)

摘 要:随着国内用户群的增长,专业软件的汉化问题受到关注。针对此问题,探讨了基于Qt的图形用户界面软件汉化方法及问题。并通过具体的项目实践,详细论述了软件汉化使用的方法及问题。提出了应用Qt设计器对GUI界面进行说明,本地用户对软件进行汉化,运用Qt Linguistg国际化工具进行软件汉化三种方法。介绍了动态语言切换功能,实现用户界面不同语言之间的转换。对汉化过程中出现的问题,提出不同的解决方法。实践表明,使用上述方法可以解决软件的汉化问题,使软件界面实现中文化。

关键词:Qt;GUI;字符串;汉化
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2011)03-0146-04

Research on Qt-Based Software Interface Technology of Chinese Localization

WANG Jia-hua, CHENG Shun-shun
(School of Computer Science, Xi'an Shiyou University Xi'an 710065, China)

Abstract: With the increasing of user groups of domestic, professional software localization issues have more concern. For this problem, study methods and issues based on the Qt graphical user interface software localization. Through the specific practice projects, discussed the methods and problems which use localization software in detail. Proposed for applying the Qt designer explain GUI interface, a local user use the methods about the software Chinses localization, use Qt Linguistg international tool to carry on the software Chinese locational. And introduces the dynamic switch language function, implement between different language's switch with user interface. It is proposed some different solution when the Chinese localization's process appear problems. Practice shows that using the above method can solve the problem of software localization, make the software interface to achieve the Chinese localization.

Key words: Qt; GUI; string; Chinese localization

0 引 言

Qt是Trolltech公司开发的给予标准框架的图形应用程序,它使用"一次编写,随处编译"的方式为开发跨平台的应用程序提供了一个标准的C++应用程序开发框架[1]。Qt支持的平台有微软操作系统、苹果机OS以及Linux操作系统[2]。它以其作为一个跨平台框架著称,拥有强大的应用程序编程接口[3],提供给应用开发者大部分的功能,来完成建立合适的、高效的图形界面程序和后台执行的应用程序,并提供了一种面向对象的可扩展性能和基于组件的编程模式[4]。

汉化主要是针对使用Qt开发的图形用户界面程序,使用MFC编写的C++应用程序的图形界面的可移植性相对较差。在对使用Qt编写的应用程序图形界面进行汉化时,相对就比较容易,并且Qt为世界上其它文字系统都提供了广泛的支持,可以实现在不同语种之间进行切换。

1 汉化技术使用的方法

目前,使用的大多数专业软件都是来源于国外,在操作过程中总是会出现一些不可避免的问题,而存在这些问题的大部分原因是语言问题。而随着国内用户群的增长,应用软件的汉化问题就引起了广泛的关注[5]。

1.1 Qt设计器

Qt设计器是Qt自有的设计GUI界面的工具,它提供一种可视化的设计方法,方便程序员的应用[6]。使用可视化的设计方法比手工编码显得更自然、更快速,并且也希望能通过可视化方法,对那些手工编码所设计的窗体,进行更加快速、容易地测试和修改[7]。

收稿日期:2010-07-09;**修回日期**:2010-10-24
基金项目:国家自然科学基金项目(50874091)
作者简介:王家华(1945-),男,教授,从事油藏描述、储层建模、地质统计学、地质图形可视化、决策分析、风险分析等方法,及其软件系统。

使用Qt设计器绘制好GUI界面后,就可以利用Qt提供的qmake工具和uic编译工具将ui文件编译生成C++源文件,并会提取需要汉化的字符串[8]。根据ui文件的特性,每次程序编译都生成新的界面.h文件,uic会自动为那些发生了改变的窗体重新生成源代码。在修改程序内部的同时将界面进行汉化。

1.2 使用Qt识别中文

在无任何标识的情况下,如果程序中出现中文字符,那么在编译后,这些字符将会以乱码的形式出现在生成的页面中。在Qt应用程序中,窗口直接加载一个QLabel窗口部件以显示一些文本信息。然而,在相对较复杂的GUI界面上,仅仅通过指定窗口部件的父子关系以期达到加载和排列窗口部件的方法是行不通的,最好的办法是使用Qt提供的布局管理器来设计页面[4]。在界面中存在部分需要转换成中文的字符串,而这些字符串又不包含在GUI界面中,那么只有选择单纯地在程序内部进行本地汉化。

1.2.1 识别汉字的预编译指令

在程序中加入识别汉字的预编译指令:

```
#include <Qtcore/QTextCodec>
```

通过使用适当的QTextCodec来调用setCodec( ),还可以指定其编码方法。QTextCodec是一种可以在Unicode[9](注:Unicode是一种支持世界上绝大多数文字系统的字符编码标准)和给定编码格式之间进行转换的对象,用于指明源代码中出现的字符串的编码方式。

1.2.2 汉字编码方式

在程序中加入能够识别中文的汉字编码,如GB2312、GBK、GB18030、BIG5等[8]。除了采用英语外,Qt对世界上其他文字系统都提供了广泛的支持。如果想将英语转换成其他语言,你只需要加入识别该种语言的编码方式即可。

```
QTextCodec::setCodecForTr(QTextCodec::codeForName("GB2312"));
```

由于QTextStream不允许在开始读取字符之后再改变这个编码格式,因此,考虑到显示编码的问题,正确读取文件的方法是使用适当的编码格式(QTextCodec::codecForName( ))函数获得重新开始读取文件。

1.2.3 程序中的实现

在应用程序中实现汉化字符串,只需要将该字符串使用tr( )函数将其提取,并进行相应的翻译即可。

```
如:QPushButton * choose_button = new QPushButton(QObject::tr("类型选择"));
```

这种汉化方法是存在一定缺陷的,它只识别本地用户需要汉化的文字,一旦将整个程序移植到另外一台机器上,无论这台机器是否配置Qt的环境变量,待程序编译后,在程序中汉化的字符串在界面中都是以乱码形式出现。

1.3 Qt Linguistg工具

Qt还提供了另外一种工具可以帮助程序员完成界面语言的转换,即Qt内置的国际化Qt Linguistg工具来帮助对应用程序进行翻译管理[10]。在一些情况下,如对移植英国英语到美国英语的应用程序,只需要修改少量的单词。但如果将英语移植为中文,就相对复杂一些,这涉及语言的不同、输入法的不同、字符编码的不同和显示方式的不同[4]。

在界面中需要汉化的字符串都会在源程序中出现,使用Qt Linguistg工具进行汉化,必须提取这些字符串。Qt只识别QString类型定义的字符串[11]。如果想让应用程序能够使用多种语言,进行各个语言之间的转换,必须确保每一个用户在程序编译所产生的界面中的字符串都使用了tr( )函数,并且在程序启动的时候,也必须载入一个翻译文件(.qm)。一旦用户可见的字符串全部使用了tr( )调用封装,就需要加载翻译文件[10]。通常是在应用程序的main( )函数中完成的,如:

```
int main( )
{
   QApplication app(argc,argv);
…
   QTranslator * qtappTranslator; = new QTranslator;
   qApp->installTranslator(qtTranslator);
   Qstring path = QApplication::applicationDirPath( )+/"translations"/;
}
```

不能一次完全提取到所有需要转换成中文的字符串。在提取到新的字符串时,进入Dos运行lupdate命令来更新提取到的字符串,即更新应用程序.pro文件。运行命令如下:

lupdate - verbose 应用程序.pro

一旦有了翻译过的.ts文件,就需要把它转换成一个可以用QTranslator使用的二进制.qm文件。如果想为所有的.ts文件生成.qm文件,可以运行lrelease命令来生成,运行命令如下:

lrelease - verbose 应用程序.pro

在所有的翻译完成之后,只需要将翻译完成的.ts文件和生成的.qm文件打包到应用程序的Debug下面,就可以随处移植程序。只要含有.ts文件和.qm文件,你就可以任意地转换语言,不必担心乱码的出现。

2 动态切换语言

应用程序能够在不同语言之间进行切换,满足不同用户的使用需求。能够在不重新启动应用程序的情

况下改变语言,能够这样做是非常完美的事情[10]。让应用程序能够动态切换语言,对于每一个窗口部件或是对话框,将他们需要翻译的字符串放在一个单独的函数(通常称为 retranslateUi( ))中,并且当语言发生改变的时候调用这个函数。由于应用程序在启动的时候,用户不知道会使用哪种语言,所以不需要加载这些翻译文件。当然,当用户需要的时候,就会动态地加载它们。

2.1 加载翻译文件

在 main( )函数中加载这些翻译文件,使用本文2.3中加入翻译文件的方法,如:

```
MainWindow::MainWindow( )
{
...
    qApp->installTranslator( &appTranslator);
    qApp->installTranslator( &qtTranslator);
}
```

其中,appTranslator 对象用于存储用于程序的当前翻译,qtTranslator 对象用于存储 Qt 的翻译。

2.2 调用语言列表

retranslateUi( )函数是 MainWindow 类中所有出现 tr( )调用的地方,是在用户使用动态切换命令改变应用程序语言的时候调用的函数。使用创建菜单函数调用语言列表,在这可以设置应用程序的默认语言,如:

```
void MainWindow::createlanguageMenu( )
{
    languageMenu = new QMenu(this);
...
    QDir qmDir = directoryOf("translation");
...
if (language == "English")
    action->setChecked(true);
}
```

2.3 动态切换语言

设置切换语言的槽函数,用户改变语言时,这一动作就会发出一个信号,该信号就会连接到改变语言的槽函数上[12],如:

```
Void MainWindow::switchlanguage(QAction *action)
{
   QString locale=action->data( ).tostring( );
   QString qmPath = directoryOf("translation").absolutePath( );
   appTranslator.load("callcenter_" + locale,qmPath);
   qtTranslator.load("qt_" + locale,qmPath);
   retranslateUi( );
}
```

当用户从 language 菜单中选择要切换的语言的时候,就会调用改变语言的槽函数。可以为应用程序和 Qt 加载翻译文件,并且调用 retranslateUi( ),由其为主窗口重新翻译所有的字符串。

3 软件汉化存在的问题

3.1 更新翻译文件

在编译好的界面中有部分未翻译的内容。通过使用 lupdata 命令来更新程序中需要转换的字符串,发现这些内容并没有出现在更新的.ts 文件中。在界面中显示的字符串必须用 tr( )函数将其提取,一些字符串并没有使用 tr( )函数,故使用 lupdate 命令更新.ts 文件。而 GUI 绘制的界面在编译时会自动地提取界面中出现的字符串。

3.2 动态控件加载

针对在翻译好的 GUI 界面中存在部分未翻译字符串的问题,文中提出一种解决方案。究其根本这一问题的存在是因为这些字符串并不是在 GUI 界面中原有的,是动态加载进来的控件,它们不是以 GUI 界面的形式出现。这些控件的作用类似于函数的全局属性,一次定义,随处调用。解决这一问题就要从程序入手,查找编写控件的代码。这些控件的定义一般都继承 QWidget 类,使用 QObject::tr( )函数提取字符串,使用 Qt Linguis 进行翻译,重新编译问题解决。

在其他手写代码界面调用动态加载控件时会出现乱码情况。这些控件在 GUI 界面加载时却显示正常。同样是加载,可出现的结果却不一样。这要从控件的全局属性入手解决。

如原有的控件全局属性定义实现如下:

```
const QString GridSelector::no_selection("<-None->");
```

更改后的全局属性定义实现如下:

```
Const QString GridSelector::no_selection
GridSelector::GridSelector(QWidget * parent ,const char * name,
GSTL_project * project),
QComboBox(parent),
Project_view(project)
{
...
no_selector = QObject::tr("<-None->");
...
insertItem(0,no_selector);
}
```

通过上述代码的改写,在不改变其全局属性定义的情况下,在内部实行赋值,进行调用,就不会出现乱码的情况。

3.3 GUI 界面更新动态链接库

应用程序中存在由 GUI 绘制的界面和代码生成的界面。在进行应用程序汉化时,完成 GUI 界面翻译工作后,将译好的界面加入到应用程序中进行编译,会发现存在部分内容是没有翻译过来的。未翻译过来的

问题分为两个部分,一部分是翻译好的GUI界面在编译后的应用程序中没有翻译成中文,另一部分是在已经翻译好的GUI界面中存在未翻译的字符串。

程序在生成解决方案的时候,首先编译的是GUI界面,通过使用uic工具将GUI界面对应的.ui文件转换成C++并且将转换结果存储在.h文件中,并将最终结果生成为.dll动态链接库。在Qt4.x版本中,此版本的文件夹下含有很多在编译程序时所需的动态链接库。这部分未正常显示的GUI界面是通过程序动态加载到主页面的,在编译生成动态链接库时只更新了Debug下面的动态链接库,而Qt4.x下面的库是没有更新的,如果要正常显示内容,就必须保存两者的一致性,故将Debug下面的库拷贝到Qt4.x下面。重新编译程序,GUI界面正常显示。

3.4 字符串强制转化

应用程序中存在部分字符串的定义类型为String,而Qt只识别QString类型定义的字符串,因此需要将String类型的字符串转换成QString类型。因为这两种字符串的定义不一样,在转化过程中必须要进行ASCII字符的转换。而一般的转换形式为:字符串.toAscll( ).data( )。但这种办法有时是行不通的,因为它牵扯到很多使用String类型定义的字符串,而有些字符串如果转换成QString类型程序是会报错的,甚至会引起很多使用这个字符串的语句出现错误。

实现字符串强制转换的代码如下:

```
classStd_scribe : public Scribe( )
{
Public:
std_scribe(std::ostream& os ) : os_(os){}
virtual void write(const std::string& str,const channel * )
{
   os_<<str;
}
Virtual void write(const QString&str,const channel * )
{
   os_str. toStrString;
}
…
}
class QTscribe: public Scribe( )
{
Public:
virtual void write(const std::string&str,const Channel * )
{
QMessageBox::critical(qApp->activeWindow( ),
QObject::tr("An Error Occurred… "),
QString( str. c_str( )),
QObject::tr("OK"));
}
virtual void write(const QString&str,const Channel * )
{
QMessageBox::critical(qApp->activeWindow( ),
QObject::tr("An Error Occurred… "),str,
QObject::tr("OK"));
}
}
```

该方法的思想是:给每一个需要转换的字符串在程序中的定义构造一个虚函数,对它们进行重载,实现它们的多态功能。

4 结束语

目前使用Qt编写的Window产品广泛用于油气资源勘探、自动化电路设计系统等软件。文中总结了Qt中使用汉化的方法,并对其内部自带的工具进行了详细的说明,实现了动态语言切换功能。上述方法是通过具体的实践总结出来的,并针对在实践中出现的问题提出解决方法。文中罗列的方法并不能解决所有的汉化问题,只是提出一种解决方法的思路。

参考文献:

[1] 周育红,闫锋欣. 数字资源跨平台整合系统的设计与实现[J]. 计算机技术与发展,2010,20(1):243-246.

[2] 王存建,张建正. 嵌入式Linux下Qt/Embedded的应用[J]. 计算机技术与发展,2006,16(11):180-182.

[3] 张方辉,王建群. Qt/Embedded在嵌入式Linux上的移植[J]. 计算机技术与发展,2006,16(7):64-66.

[4] 丁林松,黄丽琴. Qt4图像设计与嵌入式开发[M]. 北京:人民邮电出版社,2009.

[5] 严　博,欧庆于,吴晓平. Q top ia程序中文化方法研究[J]. 计算机与数字工程,2008(7):60-63.

[6] 成　洁,卢紫毅. Linux窗口程序设计-Qt4精彩实例分析[M]. 北京:清华大学出版社,2008:303-316.

[7] 芦东昕,周建彬,探振华. 基于Qt/Embedded的控件扩展研究与实现[J]. 计算机技术与发展,2006,16(10):97-100.

[8] 蔡志明,卢传富,李立夏,等. 精通Qt4编程[M]. 北京:电子工业出版社,2008:468-477.

[9] Trolltech. Qt-Cross-Platform C++ Development-Troll2tech[EB/OL]. 2007. http://www. trolltech. com/products/qt/features/index.

[10] Blanchette J,Summerfield M. C++ GUI Qt4编程[M]. 闫锋欣,译. 北京:电子工业出版社,2008:314-326.

[11] Blanchette J,Summerfield M. C++ GUI Programming wiethQt 4[M]. USA: Prentice Hall,2006.

[12] 朱吉佳,蔡家麟. 基于Qt的业务监控界面设计与实现[J]. 计算机技术与发展,2008,18(3):236-242.