Qtopia程序中文化方法研究*

严 博 欧庆于 吴晓平

(海军工程大学信息安全系 武汉 430033)

摘 要 Qtopia作为基于 Qt/Embedded的第一个嵌入式窗口环境,是目前一种主流的嵌入式 GUI系统的解决办案。分析了基于 Qtopia应用程序中文化的原理,并阐述了程序中文化的一般方法。

关键词 Qt/Embedded Qtopia 中文化

中图分类号 TP311

Research on Chinese Localization of Qtopia Application

Yan Bo Ou Qingyu Wu Xiaoping

(Department of Information Security, Naval University of Engineering, Wuhan 430033)

Abstract As the first Embedded GUI environment based on Qt/Embedded, Qtopia is one of the most popular embedded GUI implements. This paper introduces the theory of Chinese Localization of Qtopia application, and a generic method of it's Chinese localization is described in details in the end.

Key words Qt/Embedded, Qtopia, chinese localization

Class Number TP311

1 引言

Qtopia是 Trolltech公司基于 Qt/Embedded开发的第一个嵌入式窗口环境,是为基于嵌入式 Linux的 PDA、智能电话和其他移动设备设计的一个全面并且可用户化的应用程序平台和用户界面[1]。Qtopia为开发者提供了一组面向对象的 API,简化了嵌入式 GUI应用程序的开发。此外,这些 API与 Qt下是一致的,因而 Qt应用程序可以在 Qtopia平台上运行。

为了适应嵌入式系统的特点,Qtopia自身结构及其开发过程都有其特殊性。

一方面,由于嵌入式系统的存储空间有限以及系统的使用环境和功能定制的需求不同,要求其上所运行的嵌入式软件在体积小的同时也应具备较高的执行效率。

另一方面,随着国内的 Qtopia用户数量日益增多[2],为了适应国内用户群的特性,开发出的 Qtopia程序必须经过中文化。因此,Qtopia程序的中文化在整个嵌入式系统图形界面的开发过程中的重要性日益增加。

2 中文化过程

Qt/Embedded与 Qtopia中所有与用户交互的函数都是使用 QString来传递字符串,而 QString内部都是使用 unicode编码的,因此如果开发人员在程序中直接人工输入中文字并把它赋给诸如标签之类的控件,则显示的肯定是乱码[3]。另外在中文化 Qtopia程序之前,首先必须将 Qtopia环境中文化,这样中文化后的 Qtopia程序才能在 Qtopia中正确的显示汉字。此外,如果准备使用感应翻译的方法实现中文化,编写 Qtopia程序时,需将代码中所有用户可见的字符串使用 tr( )标记出来。

2.1 Qtopia环境的中文化过程

* 收稿日期:2008年 3月 2日,修回日期:2008年 3月 28日
基金项目:海军工程大学校基金 (编号:HGDJJ07020)资助。
作者简介:严博,男,硕士研究生,研究方向:信息安全。欧庆于,男,硕士,讲师,研究方向:嵌入式系统。吴晓平,男,教授,博士生导师,研究方向:系统工程、信息安全。

中文化 Qtopia环境和 Qtopia程序需要安装包括工具 linguist、lupdate、和 lrelease在内的 Qt 3. x,此外由于 Qtopia默认使用的是 helvetica字体,这种字体并不支持中文显示,所以需要安装附加的字体文件,或使用 Qt/Embedded自带的 unifont字体。

在开始中文化前,需要设置相应的环境变量并重新配置 Qtopia,使得 QPEDIR、QTDIR、DQTDIR分别指向 qtopia、qt2、dqt,并为中文化配置相应的语言环境条件。

```
export QPEDIR = /usr/local/qtopia - free - 2.2.0/qtopia
export QTDIR = /usr/local/qtopia - free - 2.2.0/qt2
export DQTDIR = /usr/local/qtopia - free - 2.2.0/dqt
./configure - languages "zh_CN"
```

而后,在 QPEDIR/qtopia/i18n/zh_CN 中增加. directory文件

```
[Translation]
File = QtopiaI18N
Context = zh_CN
[Desktop Entry]
Name[ ] = zh_CN
```

. directory起到了语言描述的作用,它定义了该语言的名称和呈现给用户的名称。

然后,运行命令 make lupdate,根据 Qtopia语言配置,提取代码中 tr( )标记的所有字符串文字,生成服务器特定的后缀为. ts的翻译源文件。而 nct_lupdate则是为不包含在源码中的用户可见字符串产生相应的翻译文件,这些字符串大多数包含在 config和 desktop文件中, config文件用来处理应用程序配置信息的存储与读取,而 desktop文件则与桌面显示密切相关。

为了将默认字体由 helvetica变为 unifont,需要修改 i18n/Zh_CN下的 QtopiaDefaults. ts文件,将代码:

```
<name > FontMap < /name >
<message >
  ……
  < source > Small, helvetica, 10 < /source >
  < translation type = "unfinished" > < /translation >
< /message >
```

改为

```
<name > FontMap < /name >
<message encoding = "UTF - 8" >
  < source > Small, helvetica, 10 < /source >
  < translation > Petit, unifont, 11 < /translation >
< /message >
```

最后,用 linguist工具翻译 i18n/Zh_CN下所有的. ts文件并利用 lrelease工具生成. qm文件。将 Zh_CN文件夹拷贝至 QPEDIR/image/opt/Qtopia/i18n路径下,重启 Qtopia,可以发现界面已经国际化成中文环境。使用 Qt/Embedded自带的 unifont字体,存在汉字显示大小不等的问题,可以通过使用其它字体如文泉驿,文鼎等字体来取得更好的显示效果。

2.2 Qtopia程序的中文化方法

Qtopia中,应用程序主要分为内建应用程序和插件程序,而与用户直接交流的基本上是插件程序,所以对 Qtopia程序的中文化基本上就是对插件程序的中文化。

将 Qtopia程序中文化有两种方式,一种是本地化编程方法,这需要在源程序中使用编码转换工具将字符串转换为本地化的编码方式,例如:

```
QTextCodec * codec = QTextCodec:: codeForName ("GBK"); //定义 GBK编码转换工具 codec
QLabel * label = new QLabel(codec - > toUnicode ("你好")); //对要显示的字符串进行编码转换
```

通过这种方法生成的程序显示的都是经过 GBK编码转换之后的字符串,而不能被其他诸如 KOI18 - R、EUC - JP等编码的字符串所代替,所以只适合提供给本地用户使用,程序的通用性不好,不符合国际化编程的要求。另外 Qtopia中可能未编译进对中文 GBK编码的 textcodec的支持,这时需要修改 $QTEDIR/src/tool/qconfig - qpe. h文件,注释掉 CODEC有关的宏定义 #define QT_NO_TEXTCODEC,增加它对 Unicode字符集的解码的功能[4],再重新编译 Qt/Embedded和 Qtopia,因此并不推荐这种方法。

另外一种是感应翻译的方法,基本原理是通过感应某一特定的翻译文件,然后加载相关翻译到程序界面中,从而实现应用程序的中文化。对 Qtopia环境的中文化就使用了这种方法,程序可以方便的翻译成多种文字,符合国际化编程的要求。

Qtopia中插件程序的装载是通过 PluginLoader类实现的,在 PluginLoader中定义了一个 queryInterface函数,分析该函数就可以知道 Qtopia插件程序感应翻译的过程, queryInterface的部分代码如下。

```
QRESULT PluginLoader:: queryInterface (const QString &name, const QUuid &id, QUnknownInterface * * iface)
{
……
QString type = name;
……
```

```
QStringList langs = languageList();
QStringList qpepaths = global_qtopiapaths();
for (QStringList::ConstIterator qit = qpepaths.begin
(); qit! =qpepaths.end(); ++qit) {
for (QStringList::ConstIterator lit = langs.begin();
lit! =langs.end(); ++lit) {
QString lang = *lit;
QString tfn = *qit+"i18n/"+lang+"/"+type+".
qm";
if (QFile::exists(tfn))
{ QTranslator * trans = new QTranslator(qApp);
if (trans->load(tfn))
qApp->installTranslator(trans);
else delete trans;
}}}
……
}
```

分析以上代码,可知载入插件程序并感应翻译的过程,首先是将插件名和 Qtopia中的语言列表分别赋予字符串变量 type和字符串列表变量 langs,根据插件和所选语言的名称可以得到该插件对应的翻译文件的完整路径 tfn,如果该翻译文件存在,则定义一个负责翻译任务的 QTranslator对象 trans,并在 trans中载入翻译文件的路径 tfn,载入成功,则在程序中加载翻译文件,否则删除 trans。

因此若想使 Qtopia程序感应翻译,需要一个对应的翻译文件 <type>.qm,并应该将该翻译文件放入 i18n/下相应语言的目录中。

为了获得该翻译文件,首先修改应用程序的.pro文件来指定想支持哪些语言[5],例如在 xx.pro文件中添加如下的条目:

```
TRANSLATIONS = xx.ts
```

然后在终端下进入程序源码所在的路径,运行命令 lupdate xx.pro,lupdate工具将提取程序代码中所有使用 tr()标记的字符串文字和应用程序.ui文件中出现的字符串文字,生成 xx.ts空白翻译源文件,之后用 linguist软件翻译此文件并保存,在终端下运行命令 lrelease xx.ts就能够生成此.ts文件对应的、能够被 Qtranslaror所理解的.qm二进制翻译文件,之所以要使用.qm文件而不直接使用.ts文件作为翻译文件,是因为对于设备而言,读取二进制格式的.qm文件比读取 XML格式的.ts文件更有效率。

中文翻译文件需放在 i18n/zh_CN目录下,按照该程序的.pro工程文件中的 TAEGET将.qm二进制翻译文件重命名,并将其复制到 QPEDIR/image/opt/Qtopia/i18n/zh_CN路径下,当在 Qtopia中选择中文显示后,每当装载运行某一插件程序时,Qtopia会自动读取 zh_CN目录下与该程序.pro项目文件的 TARGET同名的.qm文件,并将.qm文件中的翻译信息取代原程序中相应的显示字符,从而实现了程序的中文显示,达到程序中文化的目的。

3 结语

当前嵌入式产品的运用领域越来越广泛,嵌入式产品的开发也向着小型化、多功能发展,此外人们对具备图像界面的嵌入式系统的需求也越来越多,对其图形显示效果的要求也越来越高。Qt/Embedded的出现适应了这个趋势,由于其开放源代码,并且具备丰富的 API接口,优美的图形界面,优良的跨平台特性,因此越来越受到开发者的喜爱,成为目前嵌入式系统的主流 GUI之一。

参 考 文 献

[1]倪继利.Qt及 Linux操作系统窗口设计[M].北京:电子工业出版社,2006

[2]王子强,刘海燕等.Linux下图形用户界面程序的开发与实现[J].计算机应用与软件,2005,22(6):81～83

[3]郑巍,傅建兵,毛银杰.中文化和国际化[J].程序员技术,2002,2:60～62

[4]曹毅,李德玉.基于 Qt/Embedded的嵌入式桌面环境的研究与实现[J].西南民族大学学报(自然科学版)2006,32(6):1270～1274

[5]齐亮[译].C++ GUI QT3编程[M].北京:北京航空航天大学出版社,2006