1. **C++编程**
   1. 动态加载动态库失败

动态加载Qkzh\_plg.dll,但是Qkzh\_plg.dll依赖A.dll,而A不存在,则总是显示Qkzh\_plg.dll加载失败,用dependence软件就发现原因。

Key:loadlibrary

* 1. 单例

Singleton.h文件

Class Singleton

{

Public:

Static Singleton \* getIns(){return m\_instance;}

Private:

Singleton(){}

Singleton \* m\_instance;

}

Singleton.cpp文件

Singleton \* Singleton::m\_instance = null;

注释：

getIns()函数是静态函数,static并不是修饰返回值对象的,而是表明这是静态函数,所以该函数不能访问非静态成员变量;

getIns()必须是静态的,静态函数可以在类对象未初始化时调用,所以第一个掉getIns后才能初始化该对象实例。

* 1. C语言运行库MTD MDD

程序依赖外部库同时有静态库和动态库，要保证这两个库编译时；C/C++-》代码生成-》代码生成 ，该选项选择C语言运行库版本，要保证外部库编译时，选用的是一样的MDD或者MTD；如下例子会错误：

程序依赖静态库A，动态库B；

A编译时C运行库为MTD，则是静态C语言运行库；

B编译时C运行库为MDD，则是动态C语言运行库；

程序编译时，会同时加载A和B，两个C语言运行库就会同时加载，则会出现

msvcprtd.lib(MSVCP100D.dll) : error LNK2005: "bool \_\_cdecl std::uncaught\_exception(void)" (?uncaught\_exception@std@@YA\_NXZ) 已经在 libcpmtd.lib(uncaught.obj) 中定义

即msvcprtd.lib和libcpmtd.lib这两个c语言运行库冲突。

混用可能产生跨动态库传参数失效。A库调用B库中函数

Fun(list<unsigned char> para)

,A库使用Debug版C运行库,B库使用Release版C运行库, para参数在传递过程中丢失了,一定保证参数一致。

* 1. vector map 防止迭代器失效的问题

Vector:删除后,内存数据是顺序存储,被删元素后面的元素均向前移动,则原始后面迭代器均失效,所以erase返回下一个元素迭代器,使用该新迭代器

Itor = Vecotr.erase(itor)

Map:红黑树存储,删除节点,节点内存不变,只是更改节点指向的数据,所以后面迭代器仍然可用,则利用后++,传递的参数仍然是旧迭代器,这样

map.erase(itor++)

* 1. map基本使用

插入

map<int ,int> m;

m[1] = 1;直接插入

检索

不能使用m[1],如果没1这个key,则会插入一条

应该是m.find(1),返回对象迭代器

删除参见2.4

* 1. 各种链接失败原因
     1. 因编码方式造成的

库文件 1、使用多字节或者Unicode编译未统一 ,即使在项目常规中选择字符集为“未设置”,也需要在预处理器中 去掉 UNICODE宏（QT-addin生成的项目总是在预处理器中加入UNICODE宏,需要去除）,2、c/c++中语言 – 将wchar\_t设为内置类型 这两点与应用程序不一致,会造成无法链接库文件中的函数,应该将两者统一

* + 1. 某些库必须定义宏才能将宏定义为\_\_declspec(dllimport)

#if defined(Q\_WS\_WIN)

# if !defined(QT\_QTWINMIGRATE\_EXPORT) && !defined(QT\_QTWINMIGRATE\_IMPORT)

# define QT\_QTWINMIGRATE\_EXPORT

# elif defined(QT\_QTWINMIGRATE\_IMPORT)

# if defined(QT\_QTWINMIGRATE\_EXPORT)

# undef QT\_QTWINMIGRATE\_EXPORT

# endif

# define QT\_QTWINMIGRATE\_EXPORT \_\_declspec(dllimport)

# elif defined(QT\_QTWINMIGRATE\_EXPORT)

# undef QT\_QTWINMIGRATE\_EXPORT

# define QT\_QTWINMIGRATE\_EXPORT \_\_declspec(dllexport)

# endif

#else

# define QT\_QTWINMIGRATE\_EXPORT

#endif

以上这一段在MFC调用Qt的Qt\_WinMigrate库中,必须加QT\_QTWINMIGRATE\_IMPORT宏其他项目才能使用该库

* 1. c++ 编程规范
     1. 参数 输出在前,输入在后. 保证风格统一

输出参数必填,输入参数有默认值,放到后面可以不填

* + 1. 头文件排序:

本cpp头文件

c++系统文件

qt系统文件

其他库头文件

本项目头文件

* + 1. 头文件按照项目源代码结构安排,不要使用../../../ 这类

使用src/comprehensive/openData.h这种风格.

1. **VS开发环境**
   1. 重构-重命名带来的缺陷

重构-rename时,不会改变头文件开始的

#ifndef TARGETSELECT\_H

#define TARGETSELECT\_H

则,下次使用重复使用该类名,因为宏TARGETSELECT\_H被定义,则新的文件用于不会被定义,则编译器总是提示 找不到新的文件中的类。

* 1. VS编辑操作技巧
     1. 行编辑
     2. Alt + shift +T 交换光标所在行和下一行
     3. 代码模板

一段常用代码,选中后放入工具箱,下次点击即可直接使用

* + 1. Ctrl + i渐进式搜索

输入ctrl+I,按照状态栏提示进行搜索,再次按ctrl + i搜索下一个,非常好用的文档内字符匹配搜索工具

* + 1. 不同标签页直接的切换

按住,Ctrl + alt + 向下键,列出当前所有打开的cpp和h文件标签,输入文件名自动匹配,实现不用鼠标在标签页直接切换

Ctrl +tab键,也可实现在最近使用的几个标签页直接切换

* + 1. Alt + shift +S检索所有符号

exec –cmd .ab cd. 检索包含exec的符号,排除cmd字符串 以ab开头 以cd结尾, 后面的控制可不加 ,还可以勾选仅显示类名,检索出的结果上面可以直接右键菜单操作

* + 1. Alt + shift +f查看所有引用

有一个Clone Result,可以在新Tab页保存当前结果,查另一个对象的引用时,不会丢失该结果

* + 1. Alt + 上下 跳到下一个函数,跳到上一个函数
    2. Change Signature更改函数的名字,参数顺序等,会将调用该函数的地方同步修改。
    3. Encapsulate Field 给变量自动提供Get和Set函数

输入函数变量,右键 Encapsulate Field,自动提供Get和Set函数

* + 1. Alt + shift +] alt +shift +[

智能选择光标前后区域, 多次按]表示扩大,[表示减小,Alt + ] 一般选择一个函数

* + 1. Tools中,可以将类声明中函数自动排序
    2. 让vs重新解析，清空下面的路径即可

C:\Users\Administrator\Local Settings\Application Data\VisualAssist

* + 1. 成员变量输入,自动加入\_

成员变量输入时,按m后按shift键自动加入\_,从而免去输入shift+-的麻烦

* + 1. 查看引用时,显示读取的

在 va find references results结果中,D显示读取变量的,W显示写入变量的。

* 1. 调试时，应用代码更改

设置：c/c++常规 调试信息格式 用于“编辑并继续”的程序数据库 (/ZI)

* 1. 多进程编译

多进程同时编译选项，在库文件的 属性 - c++ - 命令行 加入 /MP

四核可以保证四个进程同时编译，提高一倍以上的速度。

* 1. Va snippet好用的快捷缩写

$end$光标位置

$MONTH$/$DAY$/$YEAR$ 日期

Class 关键词

class $Class\_name$

{

public:

$Class\_name$();

~$Class\_name$();

protected:

$end$

private:

}; 用户自己定义Class\_name,弹出界面让用户输入

* 1. 文件中相对路径的问题

Cpp中写相对路径的问题. Vs的exe程序需要在 “调试””工作目录”设定$(OutDir),这样程序中,qss配置中的资源的相对路径均相对该地址,保证双击程序也可以执行.

不建议将vcproj路径设定为基准路径,因为发布后,没有vcproj,用户在其他文件夹双击exe执行,则变为相对exe,路径自然错误

 Cpp中路径最好使用appDataDirAbsPath类似函数,获取exe路径,再取相对位置

保证双击exe执行也正常.

Cpp文件中, QFile styleFile("../../../Data//ComprehensiveStatus//Config//myStyle.qss");

字符串中路径相对于工程配置vcproj文件,在编译期,编译器根据工程当前路径,填补完整该路径.

干扰项：有时候配置文件编码与程序不一致如 配置文件编码 UTF-8,程序为GBK,则读入配置文件的中文为乱码,误以为文件路径不对。

1. **QT知识**
   1. QSetting

读注册表,使用

new QSettings("HKEY\_LOCAL\_MACHINE\\SOFTWARE ",QSettings::NativeFormat);

构造函数中使用的双反斜杠,取值时使用的单正斜杠

setting->value("KEY\_OraDb11g\_home1/ORACLE\_HOME")

读ini文件使用

QSettings settings("myapp.ini",QSettings::IniFormat);

* 1. Qt的界面属性设置
     1. Qt::Tool

永远在父窗口的顶层显示。用于包括一些工具按钮,通常边框会小一点。

* 1. Qobject的特性

metaObject() 返回元对象，其中存储了类名、各属性、信号等，特殊特性都依赖于该元对象

connectNotify() 有信号连接到该对象时可进行通知。

objectName() 每个对象都有自己的名称,默认为空

findChildren() 根据名称(objectName)寻找该对象的子对象

event() 事件处理，可以通过installEventFilter过滤事件

事件对象为QEvent，有一个标志位表示接受还是忽略该消息。

继承自QObject的类无复制构造函数，每一个对象的特性是唯一的

* + 1. 智能指针

QSharedPointer,共享指针,类似boost中的share\_ptr,但是不能用于有父子关系的界面qt类,因为父类释放时会调用子类的释放函数, QSharedPointer再次释放子类就出错了。

QScopedPointer出了局部范围自动删除指针,防止程序在一对大括号之间运行时突然奔溃,指针未释放的情况。

* 1. 关闭窗口

QWidget::setVisible()设置窗口显隐，show，hide，sethidden都是马甲，方便函数。

QWidget::close() 发送QCloseEvent事件，隐藏窗口，如果设置Qt::WA\_DeleteOnClose标志位则释放该对象，destroyed()信号会发送，用来将对象指针设为NULL

Widget::closeEvent ( QCloseEvent)，点击关闭按钮或者调用close槽会发送QCloseEvent事件并调用该函数

非模态对话框关闭时释放对象并置指针为空的方法：

w->setAttribute(Qt::WA\_DeleteOnClose);关闭时会删除对象，发送destroyed信号

connect(w,SIGNAL(destroyed()),this,SLOT(SetPointerNull()));槽函数中将w指针置为NULL

QDialog::done() 和close类似，隐藏窗口，结束消息循环

Dialog关闭时，除非是确定按钮，exec函数返回QDialog::Accepted，其他情况均返回Dialog::Rejected

* 1. QString的使用技巧
     1. Right;取右侧的子串
  2. Model/view delegates
     1. QitemEditorFactory中有item常见类型的delegate,不够可以使用registerEditor注册。

QitemEditorFactory根据model的setdata(QVariant)中, Qvariant实际传入的类型来进行判断该使用哪种编辑控件,所以时间Qdatetime类型会自动使用QdateTimeEdit控件。同时,对于int类型,可以自己注册registerEditor类型,对于用int表示的经纬度使用特殊处理,其他使用默认处理。

* + 1. QmodelIndex

View和delegate通过QmodelIndex在model中定位信息,但是获取每一个信息是临时变量,因为Model中的信息随时在变化,需要保存永久的model中的位置信息使用QpersistentModelIndex。

Tree每一个节点下面带一个表格坐标系,使用index(row, column, parent)表示。Parent即为Tree当前节点的QmodelIndex对象。

Views和delegates通过model index访问数据项,Model.model()返回model指针。获取model index时需要考虑model的三个属性：行,列,父项,顶层项的父项为QModelIndex()。每一个数据项包含多个角色role,

自己实现model,1、非层次结构的只要实现rowCount()和data()函数,2、层次结构的还需要实现index()和parent()函数

* + 1. QSqlTableModel::setFilter(qstring);条件语句中不要有where,不然总是执行错误。
  1. Qt的预编译头文件

C++的编译只针对cpp文件，每次将include的头文件展开插入到cpp文件中，所以可以将不变的include头文件作为预编译文件\*.pch。

设置方法：

1. 项目-预编译头- 依次设定：使用预编译文件，预编译文件名，如：myHeader.h
2. 右键myHeader.cpp 预编译头-创建预编译头，（因为只编译cpp文件，所以myHeader.cpp 只需要一句 include “myHeader.h”即可）
3. 在所有cpp的最前面加入 include “myHeader.h”（头文件中不需要，因为编译时展开cpp文件，最前面是myHeader.h的内容，保证头文件中需要的均已经定义）
4. Qt的moc.exe生成的moc\_XXXX.cpp文件因为自动生成,没有包含myHeader.h,可以在工程-属性-C/C++-高级-强制包含myHeader.h,这样moc\_XXXX.cpp代码在第一句就隐式的加载了myHeader.h头文件。（或者在xxx.h的moc.exe的命令行中加入"-fStdAfx.h",新建cpp文件时,勾选使用预编译头,就会在moc.exe的命令行中加入"-fStdAfx.h"）
   1. 鼠标按键一直按着的问题

鼠标漫游地图,一直按住左键.消息如下:

keypressEvent

keyreleaseEvent

keypressEvent

keyreleaseEvent

…

如此循环,但是我们不需要处理keyreleaseEvent,利用isAutorepeat()函数.消息顺序如下:

1 keypressEvent isAutorepeat = false; //此时还没到keyreleaseEvent

2 keypressEvent isAutorepeat = true;

3 keyreleaseEvent isAutorepeat = true;

4 若未松开,回到3 , 若松开, keyreleaseEvent isAutorepeat = false;

所以,只要在keyreleaseEvent 事件处理函数开头写:

if (me->isAutoRepeat())

{

return ;

}

即可保证返回中途不处理释放按键事件.

* 1. 字体绘制

QFont font;

font.setPixelSize(50);

QRect fontBoundingRect = QFontMetrics(font).boundingRect(tr("Qt"));

text.addText(-QPointF(fontBoundingRect.center()), font, tr("Qt"));

addText（相对于字体左下角的偏移量，字体，字符串）;

* 1. styleSheet,qss
     1. 选择符

|  |  |  |  |
| --- | --- | --- | --- |
| 选择符类型 | 例子 | 解释 | 常用场景 |
| \* |  | 通配符，匹配所有情况 |  |
| 类型选择符 | QPushButton | 匹配QPushButton这个类型及子类 |  |
| 类选择 | .QPushButton | 仅匹配QPushButton这个类型 |  |
| 属性选择符 | QPushButton[flat="false"] | 匹配有对象有属性为flat；例子：界面中强调的地方使用黄色，则给按钮，输入框等组件添加同一个属性nameEdit->setProperty("mandatoryField", true);即可 |  |
| ID选择器 | QPushButton#okButton | okButton就是object name设定的名称 | 对单个对象进行定制化 |
| 后代选择器 | QDialog QPushButton | 选择QDialog中的QPushButton包括QDialog子控件和孙子控件的按钮 | 不常用 |
| 子选择器 | QDialog > QPushButton | 前一个特例，仅QDialog的子控件按钮， | 常用 |

::符号，某个控件本身带子控件，（该类控件一般为复合控件），如下拉框的下拉按钮，通过::设置，可以与选择符合用。例子,QComboBox::drop-down { image: url(dropdown.png) }

:符号，状态选择符hover，checked，pressed

* + 1. 属性及属性类型

|  |  |  |  |
| --- | --- | --- | --- |
| 描述 | 值 | 解释 |  |
| border-style | solid:实线  dotted:点线  dashed：虚线 | 边框线型 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

属性类型

|  |  |  |  |
| --- | --- | --- | --- |
| 描述 | 值 | 解释 |  |
| Color | rgb(r, g, b)：红绿蓝  rgba(r, g, b, a)最后是透明度, | 对于顶层窗口,如QDialog,透明效果无效,设为全透明后,会有一层黑色底色,顶层窗口使用setWindowOpacity 函数设置后全部都透明了,非顶层窗口QFrame则有效 |
| qproperty-各种属性 | 例如qproperty-text:面板;将text属性设为“面板” | 通过qproperty-设置所有属性,很重要的功能,扩展了Qss的作用范围见注1 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

注1：所有的property都可以使用,但是

Q\_PROPERTY(QPoint pos READ pos WRITE move DESIGNABLE false STORED false)

表示不存储该字段,该字段信息设置了可能没有用,该字段值依赖其他字段来取。

* + 1. Box model

盒子模型，从内到外，padding 内边距， border 边框，Margin 外边距。默认为0

Padding 允许我们在内容里面挖出一块地方填充其他对象,比如qlabel

Padding-right:20px 则右侧文字部分空出20像素宽,我们可以在这里放置其他窗口部件.

默认background-image 是从border内部开始的，经过测试不包含border ，qpushbutton默认设置包含一点点宽度border的

Min-height ,max-height同时设置为同一值(见手册stylesheet的reference的width描述),才指定widget的高度,高度是指box model的contents部分，但是函数中height()返回的包含盒子模型的margin在内的所有高度.

* + 1. Border-image 使用

border-image:url(../../../res/TSKJ/zoomIn.png) 2 2 2 2;

border: 2px 2px 2px 2px transparent;

九宫格的边缘和border的宽度一致。

* + 1. 关于 setStyleSheet函数与继承性.

子窗口会继承父窗口的stylesheet 样式**语句,**是否作用取决于选择符的写法.以继承的label类myLabel包含qpushbutton为例.

myLabel->setStyleSheet(“color:red”);

这句等价于:

\*{ color:red } 则子button继承该语句,也变成红色,建议这样写:

myLabel->setStyleSheet(“Qlabel{color:red}”); 这样,不会对子窗口产生影响,

确定选择符后,子窗口不使用父窗口的样式.

* 1. 布局
     1. qgridlayout

Margin在四个方向默认为11个像素，spacing默认7像素

* + 1. 新建布局对象,无需指定父指针

QGridLayout \* layout = new QGridLayout;

在调用setlayout 或者addlayout函数时,会自动添加layout的父指针.

因为一个窗口只能有一个主layout,你在第一个layout创建时指定父指针为当前窗口,则后面主layout就无法成为当前窗口的layout.

* + 1. Layout与qwidget的父子关系

处于Layout中qwidget的父窗口不是layout,而是layout所处的qwidget,qwidget的父窗口只能是qwidget.

* 1. Qlabel的使用

可以使用在qlabel的字符串中填入html风格的字体颜色配置项。

* 1. Xml解析总结

<tag>this is string</tag>

改变this is string的值时，定位到tag之后为node后，需要再使用第一个孩子即：

Node.firstChild().setNodeValue(new string),取孩子，才是this is string的值，然后修改为参数里面的值。

解析常用函数：

elementsByTagName（str） 返回该节点下所有相应名称子节点，好用

firstChildElement（str）寻找该节点下第一个名为str的节点

namedItem和上面类似

qdomnode基类

qdomelement代表一个节点如<tag>this is string</tag>，底下可以包含更多节点

qdomtext代表this is string字符串，这也是一个节点。

* + 1. 将所有子节点组织为map里面

QMap<QString,QString> para\_value\_s;

for (QDomElement node = root.firstChildElement();!node.isNull();node=node.nextSiblingElement())

{

para\_value\_s.insert(node.tagName(),node.text());

}

可以编写一个对象,对应一级xml,包含子对象可以继续展开,或者为根节点,这样可以快速从该对象访问到需要的节点数据。如string = root.member(“first level”). member(“second level”).data();

* 1. 设置插件路径

QStringList l = QCoreApplication::libraryPaths();

QStringList library\_path\_list;

library\_path\_list.append(QCoreApplication::applicationDirPath()+"/sqldrivers");

library\_path\_list.append(QCoreApplication::applicationDirPath());

QCoreApplication::setLibraryPaths(library\_path\_list);

从而动态加载动态库时,可以从自己的路径加载;

另外发布Qt程序,参考qt.conf文件,但是上次测试该文件无效。

1. **基础知识**
   1. 日志库Log4cplus
      1. 配置文件编写

#日志根对象,呈现父子结构,子对象集成父对象特性,定义格式 对象名=[日志级别] #appender名称root这个根对象有两个个appender,为ALL\_MSGS,FATAL\_MSGS,对应不同#级别的日志

log4cplus.rootLogger= TRACE,ALL\_MSGS,FATAL\_MSGS

# filelogger是自定义的子日志对象,给特定库或者用户使用的,

log4cplus.logger.filelogger=TRACE,R

#表示不集成父logger对象的特性

log4cplus.additivity.filelogger=FALSE

#appender决定logger的信息写到哪里去,一个appender对应一个日志文件或控制台

log4cplus.appender.ALL\_MSGS=log4cplus::RollingFileAppender

log4cplus.appender.ALL\_MSGS.File=all\_msgs.log

#日志文件大小50MB,可以有5个这样的

log4cplus.appender.ALL\_MSGS.MaxFileSize=50MB

log4cplus.appender.ALL\_MSGS.MaxBackupIndex=5

log4cplus.appender.ALL\_MSGS.layout=log4cplus::PatternLayout

log4cplus.appender.ALL\_MSGS.layout.ConversionPattern=%D{%H:%M:%S %y/%m/%d} - %m %n

log4cplus.appender.FATAL\_MSGS=log4cplus::RollingFileAppender

log4cplus.appender.FATAL\_MSGS.File=fatal\_msgs.log

log4cplus.appender.FATAL\_MSGS.layout=log4cplus::PatternLayout

log4cplus.appender.FATAL\_MSGS.layout.ConversionPattern=%D{%H:%M:%S %y/%m/%d} - %m %n

#控制只有FATAL这个级别的日志才写入到fatal\_msgs.log日志文件

log4cplus.appender.FATAL\_MSGS.filters.1=log4cplus::spi::LogLevelRangeFilter

log4cplus.appender.FATAL\_MSGS.filters.1.LogLevelMin=FATAL

log4cplus.appender.FATAL\_MSGS.filters.1.LogLevelMax=FATAL

log4cplus.appender.FATAL\_MSGS.filters.1.AcceptOnMatch=true

log4cplus.appender.FATAL\_MSGS.filters.2=log4cplus::spi::DenyAllFilter

#自定义日志对象的appender,输出到R.log中

log4cplus.appender.R=log4cplus::RollingFileAppender

log4cplus.appender.R.File=R.log

#log4cplus.appender.R.MaxFileSize=5MB

#log4cplus.appender.R.MaxFileSize=500KB

#log4cplus.appender.R.MaxBackupIndex=5

#这是日志输出格式布局控制

log4cplus.appender.R.layout=log4cplus::PatternLayout

log4cplus.appender.R.layout.ConversionPattern=%D{%H:%M:%S %y/%m/%d} - %m %n

* + 1. UNICODE版本与多字节版本

中文操作系统使用vs开发环境,编译到exe中的字符均为GBK编码,所以必须使用非UNICODE版本的,否则写入文本对象会死掉,见：

if(!out.good()) {

getErrorHandler()->error( LOG4CPLUS\_TEXT("file is not open: ")

+ filename);

return;

}

但是考虑以后切换英文操作系统,后果未知,建议日志用英文编写

* + 1. 阅读笔记
    2. 未进main函数之前就生成了root对象,

class \_static\_log4cplus\_initializer {

public:

\_static\_log4cplus\_initializer() {

log4cplus::initializeLog4cplus();//初始化函数

}

* + 1. } static initializer;利用创建静态对象时创建
  1. QGIS使用
  2. QGIS修改坐标系,切割部分地图数据

打开shp文件;图层名a

右键另存为，

1、坐标系选择WGS84（只要打开过正确的坐标系，这里有记录，

或者右键正确图层，点击“将图层坐标系设为项目的坐标参考系”，就有正确参考系可以候选）；

2、选择范围，点击图层范围或者地图视图范围，保证“东南西北”是逻辑合理的范围内。

切割部分地图数据

1、切换到编辑状态

2、鼠标状态为 选择，可以选择按多边形状态选择；

3、选择区域后，右键图层另存为，注意勾选"save only selected feature"

* 1. 正则表达式

元字符

. 除换行符外任意字符

\w 字母,数字,汉字

\s 匹配任意的空白符，包括空格，制表符(Tab)，换行符

\d 匹配数字

\b 匹配单词的开始和结束,占位符,

^ 字符串的开始

$ 字符串的结束

限定符,

\* 重复0次到任意次

+ 重复一次或更多

? 重复0次或一次

{n} 重复n次

[] 匹配括号中任意几个。

* 1. 代码中有中文字符,如何做到跨平台编译、运行、编译

Windows平台下编译时

Vs2010（包括vs2008）对于cpp文件代码的中文注释,如果是UTF-8带签名,则以UTF-8解析,如果是UTF-8不带签名,则以控制面板中系统语言解析（这时想编译过,系统语言需要选择【英语美国】即UTF-8字符集,如果系统为【中国】,则以GBK解析,则会出现奇怪的编译错误）,

Linux平台下,操作系统只认UTF-8,所以编译带签名和无签名UTF-8的文件均没有问题,GBK编码的完全无法编译。

运行时,生成的exe执行程序中的字符,在vs2010以下版本时,是以操作系统的语言存储的,（即使代码是UTF-8的,中文操作系统下,编译成的执行文件中”xx”也是以GBK存储,根据setCodecForCStrings的原理,需要设定setCodecForCStrings（QTextCodec::codecForName("GBK")）;

总结：有中文,代码文件编码为带签名UTF-8, setCodecForCStrings为GBK

附录：UTF-8带签名无签名区别,就是文件开头是否表示 该文件以UTF-8编码,多3个字节而已。UTF-8是国际标准,UTF-8带签名是微软力推,也是国际标准,linux也支持。

Ref：Qstring与中文问题。

* 1. 字符编码
     1. UNICODE  和\_UNICODE宏

操作系统API有两套,本地字符集的（多字节-每个字符用char表示,如果中文则多个char）,UNICODE字符集（UNICODE,每个字符用wchar\_t表示,wchar\_t在windows统一为2字节）,定义了UNICODE宏就要用wchar\_t表示字符,鉴于我们代码中都是char,所以使用本地字符集,即多字节。

|  |  |  |  |
| --- | --- | --- | --- |
| 对比 | 多字节 | UNICODE |  |
| \* | Char | Wchar\_t |
|  | Ifstream | wifstream |
|  | ofstream | wofstream |
|  | string | wstring |
|  |  |  |
|  |  |  |
|  |  |  |

* + 1. 基础知识

如果cpp文件中字符格式为UTF-8的,则在控制面板-区域语言-管理-非Unicode程序语言。。

中需要选择英语美国,这样编译器才会将cpp文件中注释以UTF-8的格式去解析。

QTextCodec \*textCodec = QTextCodec::codecForName("GB18030");

分别设置本地，tr函数，qstring的编码方案

QTextCodec::setCodecForLocale(textCodec);

QTextCodec::setCodecForTr(textCodec);

QTextCodec::setCodecForCStrings(textCodec);

用处：Qstring a(“中文”)，qstring内部存储字符使用unicode编码，但是代码里面“中文”这样的字符串编码方式为 vs下文件 –》高级编码方案中显示的，变为qstring时，需要告诉qstring这个编码方案，即QTextCodec::setCodecForCStrings(textCodec);否则，qstring默认为拉丁编码，从而转换后得到乱码。Tr函数原理也类似。setCodecForLocale是进行本地文件读取时,告诉qt本地文件的编码方式。

Unicode、utf-8、GB2312的区别

Unicode和Gb2312是编码方式，前者国际通用，windows操作系统内部也使用该编码，gb2312是我国的汉字编码方案。

Utf-8 （unicode translation format 叫做unicode转换格式）是编码存储标准，规定了unicode的存储方式，英文1字节，希腊字符2字节，汉字3字节，如“汉”的Unicode编码为6c49,utf-8规定用1-3个字节表示编码，6c49属于0800 至FFFF范围内，使用3字节模板

1110XXXX 10XXXXXX 10XXXXXX ，用6c49的比特流代替X部分，得到E6 B1 89 就是”汉”的unicode编码的utf-8表示形式。

代码页的问题

Windows默认使用unicode表达文字，对于各自GB2312,big5等其他编码，提供代码页，从而也支持这些编码，在控制面板，区域语言 ，administrator中选择不用unicode时，使用哪种编码，若选择chinese，则不用unicode时使用gb2312编码。如果全部使用unicode则不涉及该选项。

* 1. Windows不要修改用户名,影响远程登录

修改用户名只是一个别名,用户文件夹名称等还是以前的,远程登录时依然需要使用以前的用户名。Ref：彻底修改Windows系统用户名。