**加载程序图标**

1.先找到一个ico文件当图标，文件不要过大，命名为app.ico。

2.查看帮助文档Setting the Application Icon

3.在工程目录下创建一个myapp.rc以记事本打开写入下边的一句话

IDI\_ICON1 ICON DISCARDABLE "app.ico"

4.然后在创建的工程.pro中最下面写入RC\_FILE = myapp.rc

**数据库名称**

在创建的工程.pro 中加入QT+=sql

QMYSQL //mysql数据库

QSQLITE //sqlite数据库

QOICQ //oracle数据库

**QMessageBox**

QMessageBox::information(this,tr("系统提示"),tr("请先输入密码再登录！"),QMessageBox::*Ok*);

QMessageBox::warning(this,tr("系统提示"),tr("请先输入密码再登录！"),QMessageBox::*Ok*);

**设置固定大小**

this->setMinimumSize(400,500);

this->setMaximumSize(400,500);

**设置标题**

this->setWindowTitle("连接SQLITE数据库");

**槽连接函数**

QObject::connect(ui->LISTBUTTON,SIGNAL(clicked()),this,SLOT(showData()));

**初始化窗口显示大小**

this->resize(QSize(900,600));

**去掉标题栏**

this->setWindowFlags(Qt::*FramelessWindowHint*);

**程序显示在屏幕中间**

QDesktopWidget \*desktop=QApplication::desktop();

this->move((desktop->width()-this->width())/2,(desktop->height()-this->height())/2);

**改变程序工具栏的文字字体大小**

在main.cpp 中的这句代码下QApplication a(argc, argv);

添加

QFont f("ZYSong18030",16);

setFont(f);

**改变程序图标**

this->SetWindowIcon(QIcon(":/images/icon.png"));

**设置背景色，在程序的构造函数中加入**

setPalette(QPalette(Qt::*darkCyan*));//设置背景色

setAutoFillBackground(true);

setWindowTitle("高度计"); //写标头

**画圆角按钮**

actionSave->setStyleSheet("QToolButton{border:2px groove cyan;border-radius:10px;color:gray;}QToolButton:hover{background-color:rgb(0,130,150);}QToolButton:pressed{background-color:rgb(85,170,255);}");

**退出系统**

qApp->quit();

**保存文件对话框**

QString fileName=QFileDialog::getSaveFileName(NULL,tr("保存文件"),".","Excel(\*.xlsx \*.xls)");

**将linEdit中的文本以密码形式显示**

ui->lineEditPassWord->setEchoMode(QLineEdit::*Password*);//设置为加密形式

**设置背景色与父窗口一样**

minButton->setStyleSheet("background-color:transparent");

closeButton->setStyleSheet("background-color:transparent");

**将ByteArray中的数据写到TextEdit中换行输出：**

QString strTemp;

strTemp.prepend(temp.toHex());

QString str;

for(int i=0;i<strTemp.size();i+=28)

{

str=strTemp.mid(i,28);

str.append("\r\n");

ui->textEditRead->insertPlainText(str); //写入信息

}

**把TextEdit中的数据写入到文件中换行输出：**

QString fileName=QFileDialog::getSaveFileName(this,tr("保存"),QDir::currentPath());

if(fileName.isEmpty())

{

QMessageBox::warning(this, tr("系统提示"), tr("请输入文件名"));

return;

}

QFile \*file = new QFile;

file->setFileName(fileName);

bool fileOk=file->*open*(QIODevice::*WriteOnly* | QFile::*Text*); //以只写方式打开

if(fileOk)

{

QTextStream out(file);

QString strTemp=ui->textEditRead->toPlainText();

QString str;

for(int i=0;i<strTemp.size();i+=28)

{

str=strTemp.mid(i,28); //每28个字符为一行

str.append("\r\n"); //加入换行符

out<<str; //写入文件

}

file->*close*();

delete file;

}

else

{

QMessageBox::warning(this,tr("系统提示"),tr("保存失败"));

}

**添加启动动画图片**

QApplication a(argc, argv);

QPixmap pixmap("radar.png"); //加载工程目录下图片

QSplashScreen splash(pixmap);

splash.show(); //显示图片

a.processEvents();

MainWindow w;

w.show();

splash.finish(&w);

return a.exec();

**关于使用QtableWidget**

//设置列表控件等宽显示

QHeaderView \*headerView=ui->tableWidgetData->horizontalHeader();

headerView->setSectionResizeMode(QHeaderView::*Stretch*);

//设置为不可编辑

ui->tableWidgetData->setEditTriggers(QAbstractItemView::*NoEditTriggers*);

//设置单行选中

ui->tableWidget->setSelectionBehavior(QAbstractItemView::*SelectRows*);

ui->tableWidget->setSelectionMode(QAbstractItemView::*SingleSelection*);

ui->tableWidget->setAlternatingRowColors(true); //设置表格颜色一行灰色一行白色

//纵向隐藏序号

headerView=ui->tableWidgetData->verticalHeader(); headerView->setHidden(true);

ui->tableWidgetData->setRowCount(6); //设置显示几行网格

//设置表头颜色为天蓝色表格的字体颜色为黑色

ui->tableWidgetData->horizontalHeader()->setStyleSheet("QHeaderView::section{background:skyblue;}");

ui->tableWidgetData->setStyleSheet("QTableWidget{border:2px groove cyan;border-radius:0px;color:black;}");

//网表格中添加数据

ui->tableWidgetData->setItem(indexTarget-1,4,new QTableWidgetItem(QString::number(areaTarget,'f',1)));

**设置字体**

void Dialog::paintEvent(QPaintEvent \*)

{

QFont font(“Arial”,20,QFont::Bold,true); //设置字体的类型，大小，加粗，斜体

font.setUnderline(true); //设置下划线

font.setOverline(true); //设置上划线 font.setCapitalization(QFont::SmallCaps); //设置大小写 font.setLetterSpacing(QFont::AbsoluteSpacing,5); //设置间距

QPainter painter(this); painter.setFont(font); //添加字体

QRectF ff(100,100,300,200); painter.drawRect(ff); painter.setPen(QColor(Qt::red)); painter.drawText(ff,Qt::AlignCenter,”yafeilinux”);

}

**按键响应**

void LoginDialog::*keyPressEvent*(QKeyEvent \*k)

{

if(k->key()==Qt::*Key\_Enter*){}

}

**获取加载图片的宽高，并保持宽高比**

void Dialog::paintEvent(QPaintEvent \*)

{

QPainter painter(this);

QPixmap pix; pix.load(“images/linux.jpg”);

painter.drawPixmap(0,0,100,100,pix);

qreal width = pix.width(); //获得以前图片的宽和高

qreal height = pix.height();

//将图片的宽和高都扩大两倍，并且在给定的矩形内保持宽高的比值

pix = pix.scaled(width\*2,height\*2,Qt::KeepAspectRatio); painter.drawPixmap(100,100,pix);

}

**提问对话框，点击是否消息响应**

int ok = QMessageBox::question(this,tr("提示"),tr("确定修改吗？"),QMessageBox::Yes|QMessageBox::No);

if(ok == QMessageBox::Yes){}

if(ok == QMessageBox::No){}

**重写eventFilter(Qobject\*watched,Qevent\*event)实现在控件上面绘图以及鼠标双击响应事件**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*重写函数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ui->widgetPaint->installEventFilter(this);

bool MainWindow::*eventFilter*(QObject \*watched, QEvent \*event)

{

if(watched==ui->widget&&event->type()==QEvent::*Paint*)

{

paintImage();

}

return QWidget::*eventFilter*(watched,event);

}

要现在构造函数中对你所要画图的控件注册监视对象

ui->widget->installEventFilter(this);//注册监视对象

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*画图函数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void MainWindow::paintImage()

{

QPainter painter(ui->widgetPaint); //在widget上绘制

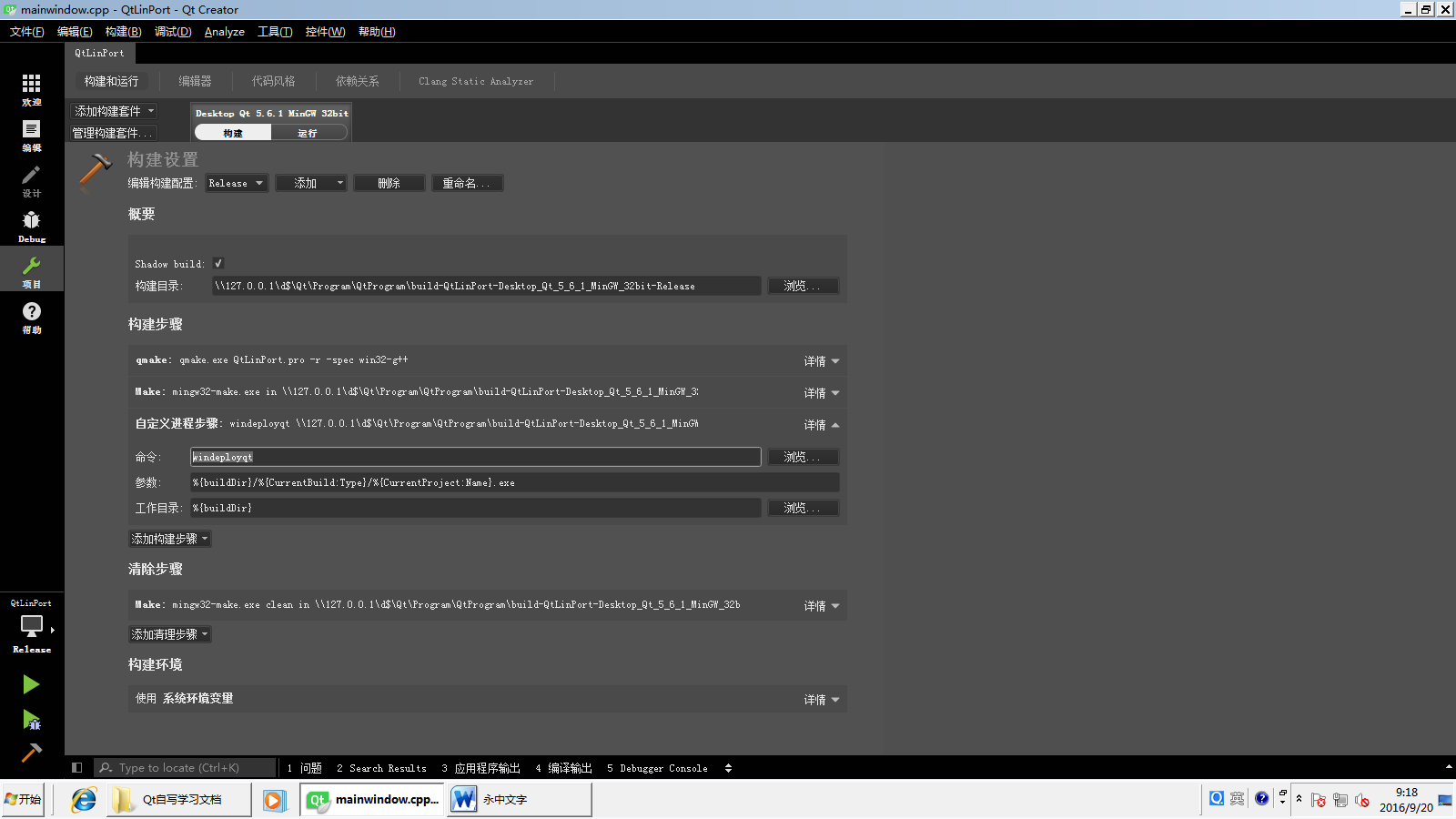
painter.setBrush(Qt::*black*); //设置画刷

painter.setPen(QColor(200,200,200,125));//设置画笔

painter.setPen(QPen(Qt::darkRed,2)); //设置画笔颜色以及粗细

}

**发布应用程序**



命令：windeployqt

参数：%{buildDir}/%{CurrentBuild:Type}/%{CurrentProject:Name}.exe

工作目录：%{buildDir}

**正则表达式的使用**

void MainWindow::regExp()

{

QRegExp rx;

rx.setPatternSyntax(QRegExp::*RegExp*); //将元字符设置为统配模式下

rx.setCaseSensitivity(Qt::*CaseSensitive*);//大小写敏感

rx.setPattern(QString("^([a-zA-Z0-9\_\.\-])+\@(([a-zA-Z0-9\-])+\.)+([a-zA-Z0-9]{2,4})+$")); //匹配邮箱字符串

if(rx.exactMatch(ui->lineEdit->text()))

{

QMessageBox::about(this,tr("系统提示"),tr("输入格式正确"));

}

else

{

QMessageBox::about(this,tr("系统提示"),tr("输入格式不正确"));

}

}

**截屏函数**

void MainWindow::on\_pushButtonScreen\_clicked()

{

QTime time=QTime::currentTime(); //获取当前事件

QDir \*dir = new QDir;

bool exist = dir->exists("Screen"); //判断是否存在Screen路径

if(!exist)

{

dir->mkdir("Screen"); //如果不存在，则创建该路径

}

//进行截屏，截取当前窗口

QPixmap pix=QPixmap::grabWindow(QApplication::desktop()->winId(),this->x(),this->y(),frameGeometry().width(),frameGeometry().height());

QString str=time.toString("HH点mm分ss秒"); //当前时间

bool pixOk =pix.save(tr("Screen/%1截屏.png").arg(str),"png");

//保存至Screen文件夹下

//提示是否截屏成功

if(pixOk)

{

ui->statusBar->showMessage(tr("截图成功"),3000); //状态栏

}

else

{

ui->statusBar->showMessage(tr("截图成功"),3000); //状态栏

}

}

**去掉字符串中空格**

str.replace(QRegExp("\\s{1,}"),"");

**创建定时器**

this->timer=new QTimer(this); //定义一个定时器

connect(timer,SIGNAL(timeout()),this,SLOT(timeUpdate()));//设置

timer->start(50); //0.05秒调用一次

**explicit显式构造函数**

如Test t1=12； //隐式调用构造函数

Test test=test(); //显式调用构造函数

**创建文件夹**

QDir \*dir = new QDir;

bool exist = dir->exists("Screen"); //判断是否存在Screen路径

if(!exist)

{

dir->mkdir("Screen"); //如果不存在，则创建该路径

}

**设置背景图片并将当前串口设置为圆角矩形**

void MessageDialog::*paintEvent*(QPaintEvent \*event)

{

QPainter painter(this);

painter.drawPixmap(rect(),QPixmap(":/image/background.png"));

QBitmap bitmap(this->size());

QPainter painter2(&bitmap);

painter2.fillRect(bitmap.rect(),Qt::*white*);

painter2.setBrush(QColor(0,0,0));

painter2.drawRoundedRect(rect(),4,4);

setMask(bitmap);

}

**自动加载串口信息**

int portCount=0;

foreach(const QSerialPortInfo&info,QSerialPortInfo::availablePorts())

{

ui->comboBoxPort->addItem(info.portName());

portCount++;

}

if(portCount==0)

{

for(int i=0;i<20;i++)

{

ui->comboBoxPort->addItem(tr("COM%1").arg(i));

}

}

**产生随机数**

qsrand(time(0));//加到构造函数中，设置种子,但每次启动程序时都会按照某种顺序产生随机数

qrand(); //取随机数

qsrand(QTime(0,0,0).secsTo(QTime::currentTime()));//实现设置随机数的初值，而程序每次启动时返回的值都不同，这样就实现了产生不同初始值的功能。

**C++静态成员**

静态函数只能调用静态成员，非静态函数可以任意调用静态成员。

静态成员变量的初始化需要在类外进行初始化！

**错误解决：**

QT在添加新类时，如果添加Q\_OBJECT这个宏时，出现错误 undefined reference to vtable for \*\*\*\*类，

解决办法：点击执行qmake

在使用VS进行编译QT时，可能会遇到中文乱码，在有中文的类的头文件中加入下面这句话，在编译时就不会出现乱码问题。

#if \_MSC\_VER >= 1600

#pragma execution\_character\_set("utf-8")

#endif