实验四 Qt编写简单的计算器

【实验目的】

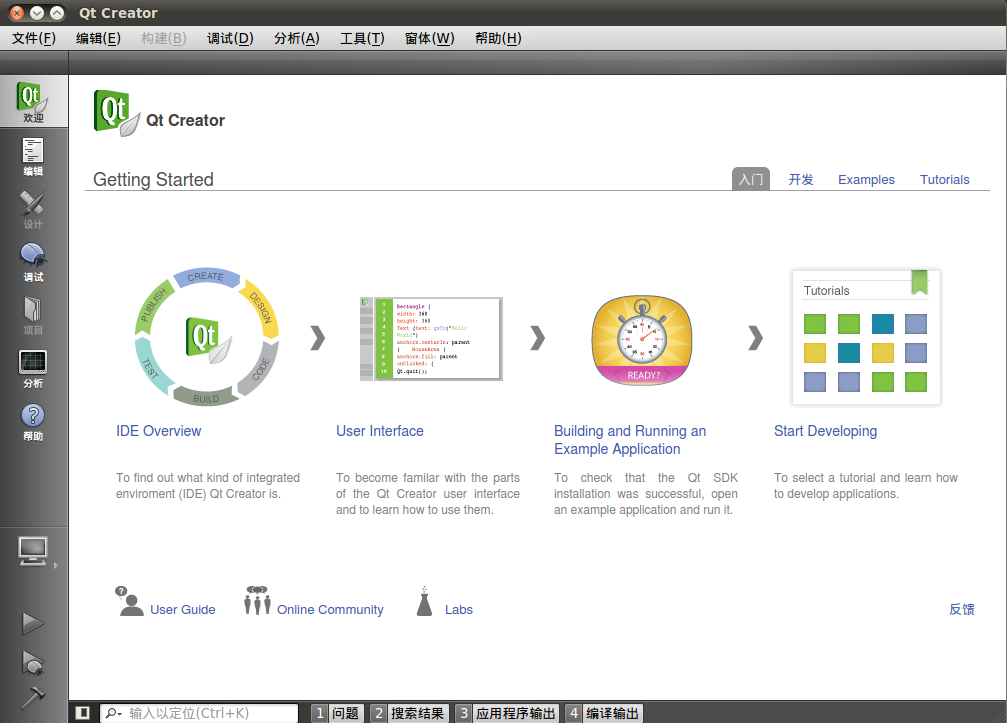
1. 熟悉QtCreator的简单操作。
2. 了解Qt程序编写框架。
3. 了解信号和槽机制，熟练掌握信号与槽在应用程序中的使用。

【实验内容】

1. 查看API手册，学习简单的Qt类的使用，如QLineEdit、QPushButton等。
2. 用QtCreator创建工程，用Qt编写计算器程序。
3. 对计算器程序进行移植。

【实验步骤】

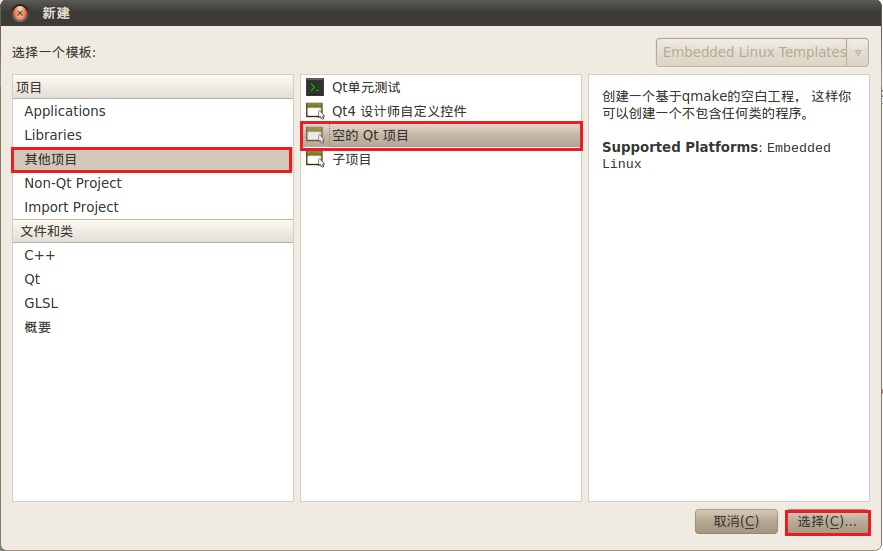
创建工程

1. 打开QtCreator，如图1所示。

2）新建工程，点击“文件（F）”->”新建文件或工程“。如图所示



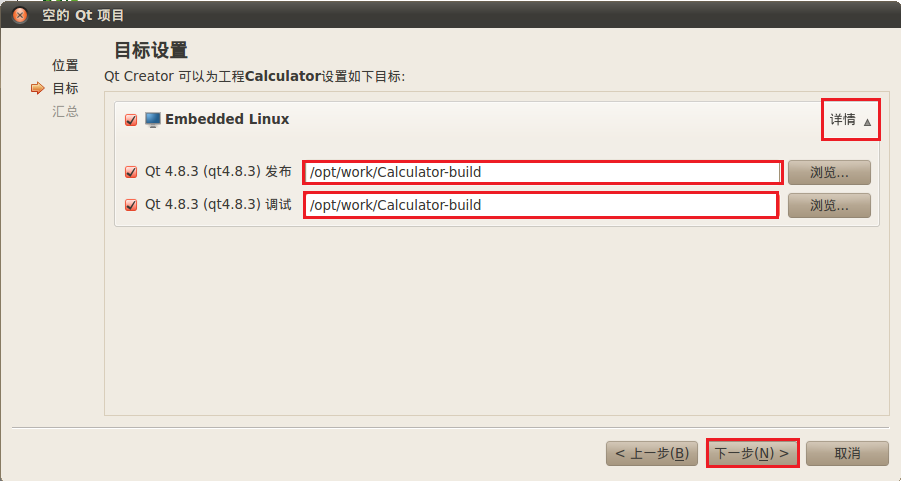
3）在弹出的窗口选择“其它项目”->”空的Qt项目“ 然后“下一步”



4）定义工程名，填写“Calculator”,路径填“/opt/work/”然后设为默认路径并点“下一步”



5）在这一不下，会显示目标，在所看到的Embedded linux左边点击“详情”。把发布和调试目录设为 /opt/work/Calculator-build. 如下图。



下一步点完成就可以了。

计算器程序的实现

计算器程序主要分以下两部分工作：一是实现计算器的图形界面；二是实现按键事件和该事件对应的功能绑定，即信号和对应处理槽函数的绑定。

1. 计算器图形界面的实现

通过分析计算器的功能我们可知，需要16个按键和一个显示框，同时考虑到整体的排布，还需要水平布局器和垂直布局器。通过组织这些类我们可以实现一个简单的带有数字0~9，可以进行简单四则运算且具有清屏功能的计算器。对于这些类的具体操作会在后面的代码中详细说明。

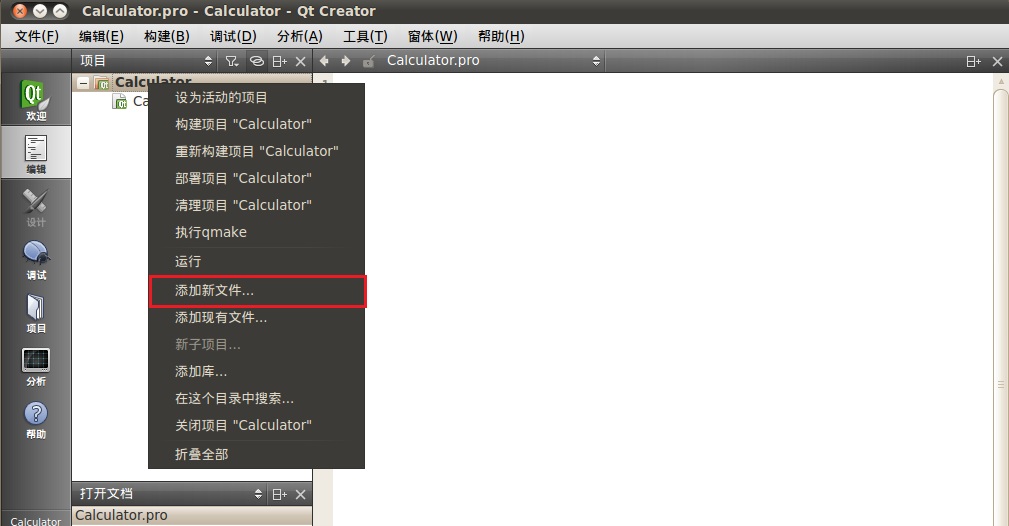
1. 信号和对应槽函数的绑定

分析计算器的按键我们可以把按键事件分为以下三类，一是简单的数字按键，

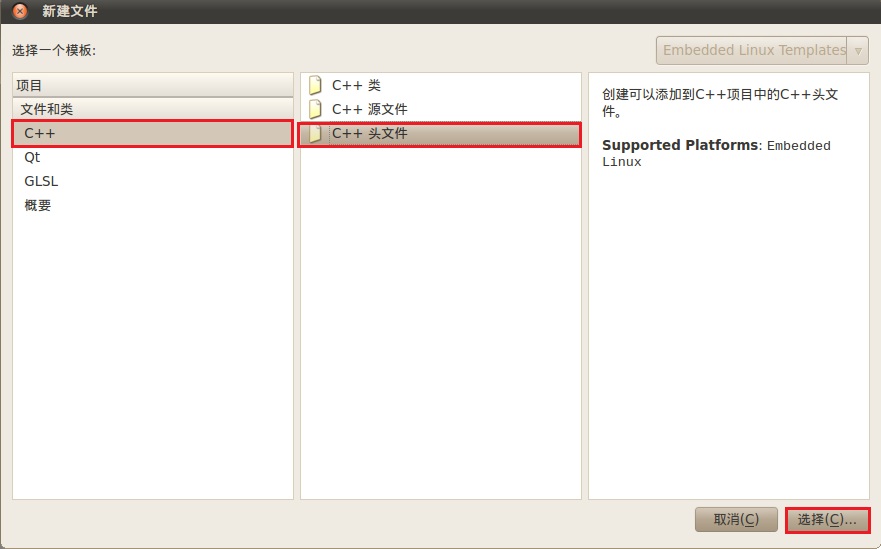
主要进行数字的录入，这类按键包括按键0~9；二是运算操作键，用于输入数学运算符号，进行数学运算和结果的显示，这类按键包括“+”，“-”，“\*”，“/”，“=”；三是清屏操作键，用于显示框显示信息的清除。

1. 进入刚才创建的空工程，双击左侧的Calculator.pro，在主编辑框中目前显示Calculator.pro的内容为空。
2. 添加文件calculator.h

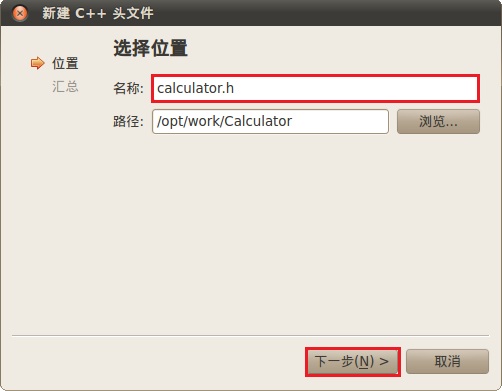
在工程Calculator上面点击右键，然后点击“添加新文件”



选择添加C++ 头文件

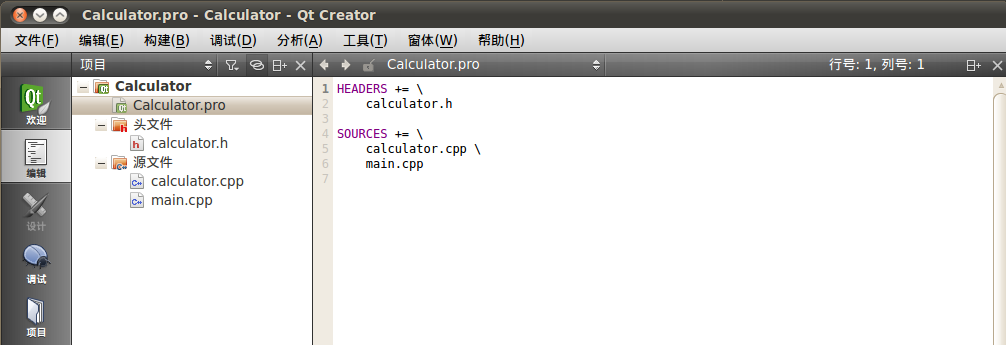


进入下一步后输入文件名calculator.h（如图所示），然后完成文件的添加。



1. 添加文件calculator.cpp和main.cpp

与添加文件calculator.h的过程类似，只是在选择文件类型时选择为C++ 源文件。完成后可以查看Calculator.pro文件的内容。工程结构如下。



计算器程序源代码的分析说明

1. 对calculator.h源代码的简要说明

#ifndef CALCULATOR\_H

#define CALCULATOR\_H //对calculator.h头文件的声明

#include <QWidget> //包含主窗体类

#include <QPushButton> //包含按键类

#include <QVBoxLayout> //包含垂直布局器类

#include <QHBoxLayout> //包含水平布局器类

#include <QLineEdit> //包含显示框类

class Calculator : public QWidget //计算器继承自主窗体类

{

Q\_OBJECT //必须加上这句，如果要调用信号，槽函数的操作的话

public:

Calculator(); //计算器类的构造函数

~Calculator(); //计算器类的析构函数

public slots: //定义各个按键按下后对应操作处理的槽函数

void zeroButtonPress();

void oneButtonPress();

void twoButtonPress();

void threeButtonPress();

void fourButtonPress();

void fiveButtonPress();

void sixButtonPress();

void sevenButtonPress();

void eightButtonPress();

void nineButtonPress();

void addButtonPress();

void subButtonPress();

void mulButtonPress();

void divButtonPress();

void clearButtonPress();

void equButtonPress();

private:

QLineEdit \*operateEdit;//声明显示框

QPushButton \*zeroButton;//声明数字按键¹

QPushButton \*oneButton;

QPushButton \*twoButton;

QPushButton \*threeButton;

QPushButton \*fourButton;

QPushButton \*fiveButton;

QPushButton \*sixButton;

QPushButton \*sevenButton;

QPushButton \*eightButton;

QPushButton \*nineButton;

QPushButton \*clearButton;//声明运算符按键

QPushButton \*addButton;

QPushButton \*subButton;

QPushButton \*divButton;

QPushButton \*mulButton;

QPushButton \*equButton;

QHBoxLayout \*firstLayout;//声明水平布局器，该布局器主要对16个按键进行布局

QHBoxLayout \*secondLayout;

QHBoxLayout \*thirdLayout;

QHBoxLayout \*fourthLayout;

QVBoxLayout \*mainLayout;//声明垂直布局器，该布局器主要对主窗体上面的空间进行排布

QString input1;//计算器第一个运算操作数

QString input2;//计算器第二个运算操作数

char operate;//运算符

};

#endif // CALCULATOR\_H

1. 对calculator.cpp源代码的简要说明

首先是构造函数的实现：

Calculator::Calculator()

{

operateEdit = new QLineEdit(this);//初始化显示框

operateEdit->setReadOnly(true); //设置显示框为只读

operateEdit->setText(tr("0"));//初始化显示框显示数据为0

zeroButton = new QPushButton;//初始化按键

zeroButton->setText(tr("0"));//设置按键上显示的标签，以下对按键相关的操作类似

oneButton = new QPushButton;

oneButton->setText(tr("1"));

twoButton = new QPushButton;

twoButton->setText(tr("2"));

threeButton = new QPushButton;

threeButton->setText(tr("3"));

fourButton = new QPushButton;

fourButton->setText(tr("4"));

fiveButton = new QPushButton;

fiveButton->setText(tr("5"));

sixButton = new QPushButton;

sixButton->setText(tr("6"));

sevenButton = new QPushButton;

sevenButton->setText(tr("7"));

eightButton = new QPushButton;

eightButton->setText(tr("8"));

nineButton = new QPushButton;

nineButton->setText(tr("9"));

clearButton = new QPushButton;

clearButton->setText(tr("Clear"));

addButton = new QPushButton;

addButton->setText(tr("+"));

subButton = new QPushButton;

subButton->setText(tr("-"));

mulButton = new QPushButton;

mulButton->setText(tr("\*"));

divButton = new QPushButton;

divButton->setText(tr("/"));

equButton = new QPushButton;

equButton->setText(tr("="));

firstLayout = new QHBoxLayout;//初始化水平布局器firstLayout

firstLayout->addWidget(zeroButton); //把按键zeroButton添加到firstLayout

firstLayout->addWidget(oneButton); //把按键oneButton添加到firstLayout

firstLayout->addWidget(twoButton); //把按键twoButton添加到firstLayout

firstLayout->addWidget(addButton); //把按键threeButton添加到firstLayout，以下对水平布局器的操作类似

secondLayout = new QHBoxLayout;

secondLayout->addWidget(threeButton);

secondLayout->addWidget(fourButton);

secondLayout->addWidget(fiveButton);

secondLayout->addWidget(subButton);

thirdLayout = new QHBoxLayout;

thirdLayout->addWidget(sixButton);

thirdLayout->addWidget(sevenButton);

thirdLayout->addWidget(eightButton);

thirdLayout->addWidget(mulButton);

fourthLayout = new QHBoxLayout;

fourthLayout->addWidget(nineButton);

fourthLayout->addWidget(clearButton);

fourthLayout->addWidget(equButton);

fourthLayout->addWidget(divButton);

mainLayout = new QVBoxLayout(this);//初始化垂直布局器mainLayout

mainLayout->addWidget(operateEdit); //把显示数据框operateEdit加到mainLayout

mainLayout->addLayout(firstLayout); //把水平布局器firstLayout添加到mainLayout

mainLayout->addLayout(secondLayout); //把水平布局器secondLayout添加到mainLayout

mainLayout->addLayout(thirdLayout); //把水平布局器thirdLayout添加到mainLayout

mainLayout->addLayout(fourthLayout); //把水平布局器fourthLayout添加到mainLayout

connect(zeroButton,SIGNAL(clicked()),this,SLOT(zeroButtonPress()));

//把按键zeroButton的按下事件同zeroButtonPress()绑定到一起，以下操作类似

connect(oneButton,SIGNAL(clicked()),this,SLOT(oneButtonPress()));

connect(twoButton,SIGNAL(clicked()),this,SLOT(twoButtonPress()));

connect(threeButton,SIGNAL(clicked()),this,SLOT(threeButtonPress()));

connect(fourButton,SIGNAL(clicked()),this,SLOT(fourButtonPress()));

connect(fiveButton,SIGNAL(clicked()),this,SLOT(fiveButtonPress()));

connect(sixButton,SIGNAL(clicked()),this,SLOT(sixButtonPress()));

connect(sevenButton,SIGNAL(clicked()),this,SLOT(sevenButtonPress()));

connect(eightButton,SIGNAL(clicked()),this,SLOT(eightButtonPress()));

connect(nineButton,SIGNAL(clicked()),this,SLOT(nineButtonPress()));

connect(addButton,SIGNAL(clicked()),this,SLOT(addButtonPress()));

connect(subButton,SIGNAL(clicked()),this,SLOT(subButtonPress()));

connect(mulButton,SIGNAL(clicked()),this,SLOT(mulButtonPress()));

connect(divButton,SIGNAL(clicked()),this,SLOT(divButtonPress()));

connect(equButton,SIGNAL(clicked()),this,SLOT(equButtonPress()));

connect(clearButton,SIGNAL(clicked()),this,SLOT(clearButtonPress()));

this->setWindowTitle(tr("Calculator"));//设置窗体标题为Calculator

input2= "0";//初始化运算操作数2为0

input1 = "0";//初始化运算操作数1为0

operate = '0';//初始化运算符为’0’

}

然后是析构函数的实现：

Calculator::~Calculator()//析构函数主要完成对构造函数中所声明的QLineEdit、QPushButton、QHBoxLayout、QVBoxLayout类的对象的回收工作（可以不定义析构函数，程序运行结束时会自动调用系统默认的析构函数）

{

if (operateEdit != NULL) //

{

delete operateEdit;

operateEdit = NULL;

}

if (zeroButton != NULL)

{

delete zeroButton;

zeroButton = NULL;

}

......

}

根据前面对按键事件的分析，有数字输入键，运算操作符输入键和清屏键三种，故对每种事件的槽响应函数都只说明一种，其他依此类推。

数字输入键响应槽函数，以按键“1”为例：

void Calculator::oneButtonPress()

{

if(input2=="0")//如果当前显示框为0

{

input2="1";//变0为1

}

else//如果当前显示框不为0

{

input2= operateEdit->text();

input2.append(tr("1"));//在显示的数据后面追加1

}

operateEdit->setText(input2);//更新显示框中的显示信息

}

运算操作符输入键响应槽函数，以按键“+”为例：

void Calculator::addButtonPress()

{

float first,second;

input2= operateEdit->text();//把当前显示的数据保存到运算操作数2中

if(operate == '0')//如果是第一次按下运算符键

{

input1 = input2;//把运算操作数2中的数据保存到运算操作数1中

input2 = "0";//清除运算操作数2中的数据

operate = '+';//把运算符键置‘+’

}

Else//如果是第二次按下运算符键

{

second=input2.toFloat();//把运算操作数2中的数据转化为浮点类型

first=input1.toFloat();//把运算操作数1中的数据转化为浮点类型

switch(operate)//根据当前的运算符判断做何操作

{

case '+':first = first+second;break;

case '-':first = first-second;break;

case '\*':first = first\*second;break;

case '/':first = first/second;break;

}

input1 = QString::number(first,'f',10);//把运算的结果转化成为可以在显示框显示的类型

input2 = "0";//清除运算操作数2中的数据

operate = '+';//把运算符键置‘+’

}

operateEdit->setText(input1);//更新显示框中的显示内容

}

清屏操作响应函数：

void Calculator::clearButtonPress()

{

operate = '0'; //把运算符键置‘0’

input2 = "0";//把运算操作数2清零

input1 = "0";//把运算操作数1清零

operateEdit->setText(input2); //更新显示框中的显示内容

}

1. 对main.cpp源代码的简要说明

#include <QApplication>//包含应用程序类

#include "calculator.h"//包含计算器类

int main(int argc, char \*argv[])//main函数的标准写法

{

QApplication app(argc, argv); //创建一个QApplication对象，管理应用程序的资源

Calculator mainwindow; //产生一个计算器对象

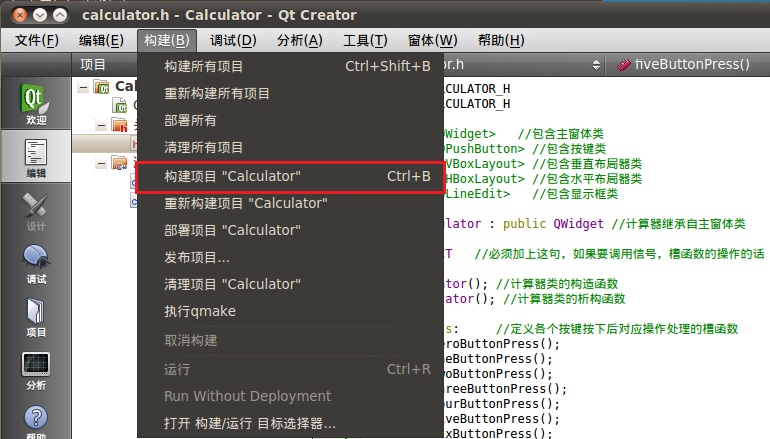
mainwindow.showMaximized();//显示计算器窗体(默认以最大化的形式显示)

return app.exec();//让程序进入消息循环，等待可能的菜单、工具条、鼠标等的输入，进行响应。

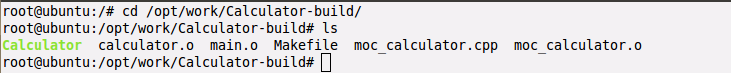
}

计算器程序在 X86 上的编译，目标机运行

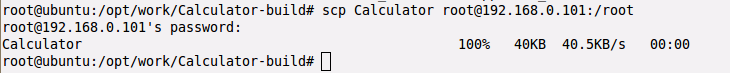
1. 完成源程序的编辑后，可以按一下的图片，从菜单点击“构建”->”构建项目Calculator”，这时QtCreator会自动编译源程序并生成可执行程序（这里默认的编译环境是目标机的）。



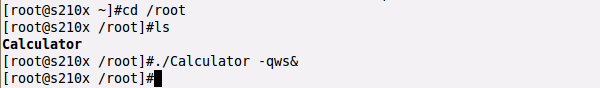
可以在/opt/work/Calculator-build目录下找到可执行程序。



2）我们从这边就可以用scp命令来发到我们的目标机/root 目录。



3）现在就可以在我们的目标机上运行。



由于本程序没有集成键盘动作，只能通过鼠标点击按钮来进行操作