一．信号和槽机制

信号和槽机制分为信号和槽函数，用于处理事件。当某个事件发生时，比如说某个按钮被点击后，它就会发出一个信号（signal），如果有对象对这个信号感兴趣，那么它就会使用连接（connect）函数，将该信号与自己的一个槽函数（slot）连接起来，槽函数被调用。槽函数负责点击按钮后的处理实现。

二，系统自带的信号和槽函数

Connect函数连接信号和槽函数，信号由发送者发送，槽函数由信号的接收者调用。

Connect函数的各个参数如下：

信号的发送者

发送的信号 地址

信号的接收者

信号的接收器

处理的槽函数 地址

需求： 点击按钮， 关闭窗口

QPushButton \* quitBtn = new QPushButton("关闭窗口",this);

connect(quitBtn,&QPushButton::clicked,this,&MyWidget::close);

系统自带的信号和槽函数如何查找呢？

通过帮助文档查找。如上面代码中的信号地址是如何找到的呢？

首先，我们的发送者是按钮对象，那么查找QPushButton的帮助文档，在文档中寻找signals，可是并没有找到点击按钮的信号，那么我们就要想到继续往QPushButton的父类中查找，父类为QAbstractButton，终于找到了 clicked()信号。因此将信号的地址填到函数的参数中。

槽函数和信号的地址用本对象或者父对象都可以。因此，用QPushButton或者QAbstractButton都可以。

槽函数的查找方式和信号是一样的，在此就不多赘述了。

三．自定义的信号和槽函数

（一）不带参数的自定义信号和槽函数

当系统自带的信号和槽函数不符合要求时，我们需要自己来定义信号和槽函数。

首先，信号和槽函数一定是分别属于某个对象的，而这个对象一般是自己定义的类对象。自定义的信号只需要声明在头文件中，不需要实现。返回值是void，且需要在关键字signals:下声明。如下所示：

class **Teacher** : public QObject

{

Q\_OBJECT

public:

explicit **Teacher**(QObject \*parent = nullptr);

signals:

//自定义信号,写道signals下

//返回类型时void

//信号只需要声明，不需实现

//信号可以有参数，也可以重载

void **hungry**(QString foodName);

void **hungry**();

};

自定义的槽函数既需要声明，也需要实现。声明在头文件中，且在关键字public slots:下，如下所示：  
class **Student** : public QObject

{

Q\_OBJECT

public:

explicit **Student**(QObject \*parent = nullptr);

signals:

public slots:

//自定义槽函数

//高版本直接写在public或者全局

//返回值void

//槽函数需要声明和实现

//槽函数可以有参数，也可以重载

void **treat**(QString foodName);

void **treat**();

};

槽函数的实现在对应的cpp文件中，

void Student::**treat**()

{

qDebug() << "请老师吃饭！";

}

定义和实现完信号和槽函数后，就可以在窗口cpp文件中添加connect函数将信号和槽函数连接起来，

teacher = new Teacher(this);

student = new Student(this);

connect(teacher,&Teacher::hungury,student,&Student::treat);

光Connect连接信号和槽函数还是不够，自定义的信号还需要通过发送者发送信号，如下所示：

Emit teacher->hungry();

（二）有参的自定义信号和槽函数

有参的和无参的差别不大，若有参和无参的信号和槽函数是同时存在的，那么connect的信号和槽函数地址就会出现无法辨别的错误，需要我们自定义函数指针指向两种函数地址，然后再做connect操作。如下所示：

自定义带参信号和槽函数

void hungury(QString name); 自定义信号

void treat(QString name ); 自定义槽

自定义函数指针

void (Teacher:: \* teacherSingal)(QString) = &Teacher::hungury;

void (Student:: \* studentSlot)(QString) = &Student::treat;

connect(teacher,teacherSingal,student,studentSlot);

（三）注意事项

* **发送者和接收者都需要是QObject的子类（当然，槽函数是全局函数、Lambda 表达式等无需接收者的时候除外）；**
* **信号和槽函数返回值是 void**
* **信号只需要声明，不需要实现**
* **槽函数需要声明也需要实现**
* **槽函数是普通的成员函数，作为成员函数，会受到 public、private、protected 的影响；**
* **使用 emit 在恰当的位置发送信号；**
* **使用connect()函数连接信号和槽。**
* **任何成员函数、static 函数、全局函数和 Lambda 表达式都可以作为槽函数**
* **信号槽要求信号和槽的参数一致，所谓一致，是参数类型一致。**
* **如果信号和槽的参数不一致，允许的情况是，槽函数的参数可以比信号的少，即便如此，槽函数存在的那些参数的顺序也必须和信号的前面几个一致起来。这是因为，你可以在槽函数中选择忽略信号传来的数据（也就是槽函数的参数比信号的少）。**

四．扩展

（一）一个信号可以和多个槽函数连接

可以这样理解，我们通过传递一个信号，可以调用多个槽函数，实现多个功能，这些槽函数是一个个被调用的，但是调用顺序是不确定的。

（二）多个信号可以和一个槽函数连接

理解：多个信号都想要实现一种功能，如关闭操作我想通过窗口的X图标实现，也可以通过状态栏实现。

（三）一个信号可以连接到另一个信号

理解：信号不光可以和槽函数直接连接，也可以和信号连接，实现更多种形式的信号传播形式。如通过按钮信号连接Teacher::hungty，再通过Teacher::hungry连接Student::treat。

（四）信号和槽函数的连接可以断开

通过disconnect函数，语法和connect函数一样。

（五）槽函数可以使用Lambda表达式