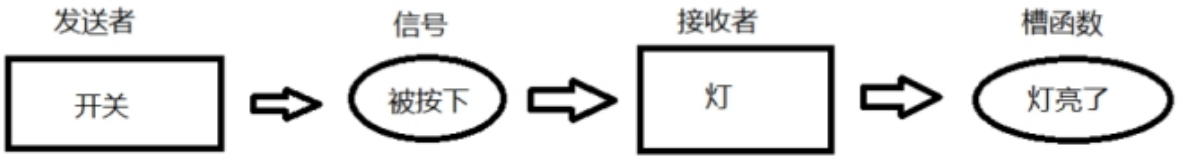
1 信号与槽的概念

信号槽是 Qt 框架引以为豪的机制之一。所谓信号槽，实际就是观察者模式。当某个事件发生之后，比如，按钮检测到自己被点击了一下，它就会发出一个信号（signal）。这种发出是没有目的的，类似广播。如果有对象对这个信号感兴趣，它就会使用连接（connect）函数，意思是，将想要处理的信号和自己的一个函数（称为槽（slot））绑定来处理这个信号。也就是说，当信号发出时，被连接的槽函数会自动被回调。这就类似观察者模式：当发生了感兴趣的事件，某一个操作就会被自动触发。



2 系统自带的信号与槽

connect()函数最常用的一般形式：

connect(sender, signal,　receiver, slot);

参数解释：

sender：发出信号的对象

signal：发送对象发出的信号

receiver：接收信号的对象

slot：接收对象在接收到信号之后所需要调用的函数（槽函数）

```
#include "widget.h"
#include <QPushButton>
Widget::Widget(QWidget *parent)
 : QWidget(parent)
{
 //创建一个按钮 点击按钮关闭窗口
 //this->close();
 QPushButton *p = new QPushButton("关闭",this);
 //设置连接 点击按钮产生信号 会调用窗口的close函数
  connect(p,&QPushButton::clicked,this, &QWidget::close);
 }

Widget::~Widget()
{

}
```

3 按钮常用的信号

那么系统自带的信号和槽通常如何查找呢，这个就需要利用帮助文档了，在帮助文档中比如我们上面的按钮的点击信号，在帮助文档中输入QPushButton，首先我们可以在Contents中寻找关键字 signals，信号的意思，但是我们发现并没有找到，这时候我们应该想到也许这个信号的被父类继承下来的，因此我们去他的父类QAbstractButton中就可以找到该关键字，点击signals索引到系统自带的信号有如下几个

Signals

```
void    clicked(bool checked = false)
void    pressed()
void    released()
void    toggled(bool checked)
```

- 3 signals inherited from QWidget
- 2 signals inherited from QObject

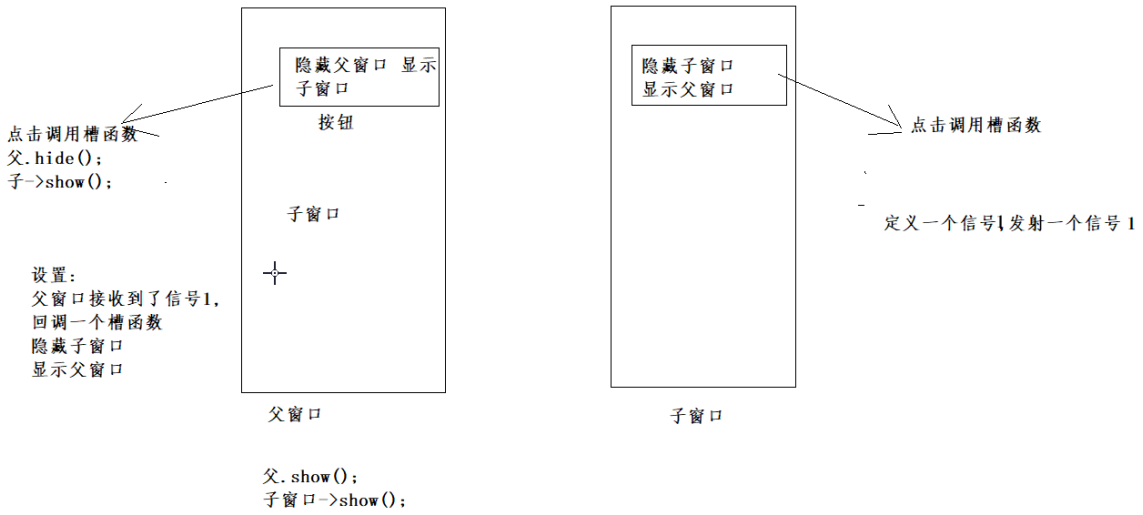
这里的clicked就是我们要找到，槽函数的寻找方式和信号一样，只不过他的关键字是slot。

4 自定义的槽函数

```
#ifndef WIDGET_H
#define WIDGET_H

#include <QWidget>
#include <QPushButton>
class Widget : public QWidget
{
 Q_OBJECT

public:
 Widget(QWidget *parent = 0);
 ~Widget();
 QPushButton *button;
public slots:
 void print();//槽函数

};

#endif // WIDGET_H

```

```cpp
#include "widget.h"
#include <QDebug>
Widget::Widget(QWidget *parent)
 : QWidget(parent)
{
 button = new QPushButton;
 button->setParent(this);
 button->setText("打印千锋物联网牛逼");
 //注册信号与槽的连接
  //槽函数可以是普通的成员函数 还可以是槽函数
  //如果信号没有参数 槽函数也不能有参数 如果信号有参数,槽函数可以有也可以没有
  connect(button,&QPushButton::pressed,this,&Widget::print);

}

 void Widget::print()
 {

 qDebug()<<"千锋物联网嵌入式学科业内最牛逼";
 }
Widget::~Widget()
{

}
```

5 自定义信号




```cpp

```

widget.h

```
#ifndef WIDGET_H
#define WIDGET_H

#include <QWidget>
#include "sonwidget.h"
#include <QPushButton>
class Widget : public QWidget
{
 Q_OBJECT

public:
 Widget(QWidget *parent = 0);
 ~Widget();
 Sonwidget *sonwindow;
 QPushButton *button1;
public slots:
 void button_cb();
 void signal_cb(int a);
};

#endif // WIDGET_H

```

widget.cpp

```
#include "widget.h"
#include "QDebug"
Widget::Widget(QWidget *parent)
 : QWidget(parent)
{
 this->setWindowTitle("父窗口");
 this->sonwindow = new Sonwidget;
 sonwindow->show();
 button1 = new QPushButton;
 button1->setText("隐藏父窗口 显示子窗口");
 button1->setParent(this);
 connect(button1,&QPushButton::clicked,this,&Widget::button_cb);

connect(sonwindow,&Sonwidget::show_hide_signal,this,&Widget::signal_cb);
}
void Widget::signal_cb(int a)
```

```
{
 qDebug()<<a;
 this->show();
 this->sonwindow->hide();
}
void Widget::button_cb()
{
 this->hide();
 this->sonwindow->show();

}
Widget::~Widget()
{

}

```

sonwidget.h

```
#ifndef SONWIDGET_H
#define SONWIDGET_H

#include <QWidget>
#include <QPushButton>
class Sonwidget : public QWidget
{
 Q_OBJECT
public:
 explicit Sonwidget(QWidget *parent = nullptr);
 QPushButton *button2;

signals: //定义信号
 //信号没有返回值 可以有参数 信号函数不需要定义 只需要声明
 void show_hide_signal(int a);
public slots:
 void emit_mysignal();
};

#endif // SONWIDGET_H

```

sonwidget.cpp

```cpp
#include "sonwidget.h"

Sonwidget::Sonwidget(QWidget *parent) : QWidget(parent)
{
 button2 = new QPushButton("隐藏子窗口显示父窗口",this);
 this->setWindowTitle("子窗口");
 connect(button2,&QPushButton::clicked,this,&Sonwidget::emit_mysignal);
}

void Sonwidget::emit_mysignal()
{
  //点击按钮的槽函数 发射信号
  emit show_hide_signal(10);

}

```

6 信号与槽需要注意的事项

自定义信号槽需要注意的事项

- 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）。

7 信号槽的拓展

一个信号可以和多个槽相连

如果是这种情况，这些槽会一个接一个的被调用，但是它们的调用顺序是不确定的。

多个信号可以连接到一个槽

只要任意一个信号发出，这个槽就会被调用。

一个信号可以连接到另外的一个信号

当第一个信号发出时，第二个信号被发出。除此之外，这种信号-信号的形式和信号-槽的形式没有什么区别。

槽可以被取消链接

这种情况并不经常出现，因为当一个对象delete之后，Qt自动取消所有连接到这个对象上面的槽。

信号槽可以断开

利用disconnect关键字是可以断开信号槽的

使用Lambda 表达式

在使用 Qt 5 的时候，能够支持 Qt 5 的编译器都是支持 Lambda 表达式的。在连接信号和槽的时候，槽函数可以使用Lambda表达式的方式进行处理。后面我们会详细介绍什么是Lambda表达式

8 信号与槽在QT4中的写法

QT5 写法

```
QObject::connect(&button, &QPushButton::clicked,&app, &QApplication::quit);
```

编译期间检查类型是否匹配。Qt 5 中，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。

QT4 写法

```
QObject::connect(&button, SIGNAL(clicked()), &app, SLOT(quit()));
```

SIGNAL和SLOT不会检查里面的字符串

这里使用了SIGNAL和SLOT这两个宏，将两个函数名转换成了字符串。注意到connect()函数的 signal 和 slot 都是接受字符串，一旦出现连接不成功的情况，Qt4是没有编译错误的（因为一切都是字符串，编译期是不检查字符串是否匹配），而是在运行时给出错误。这无疑会增加程序的不稳定性。

9 lambda表达式

C++11中的Lambda表达式用于定义并创建匿名的函数对象，以简化编程工作。首先看一下Lambda表达式的基本构成：

```
[capture](parameters) mutable ->return-type
{
statement
}
```

[函数对象参数](操作符重载函数参数)mutable ->返回值{函数体}

① 函数对象参数；

[]，标识一个Lambda的开始，这部分必须存在，不能省略。函数对象参数是传递给编译器自动生成的函数对象类的构造函数的。函数对象参数只能使用那些到定义Lambda为止时Lambda所在作用范围内可见的局部变量（包括Lambda所在类的this）。函数对象参数有以下形式：

n 空。没有使用任何函数对象参数。

n =。函数体内可以使用Lambda所在作用范围内所有可见的局部变量（包括Lambda所在类的this），并且是值传递方式（相当于编译器自动为我们按值传递了所有局部变量）。

n &。函数体内可以使用Lambda所在作用范围内所有可见的局部变量（包括Lambda所在类的this），并且是引用传递方式（相当于编译器自动为我们按引用传递了所有局部变量）。

n this。函数体内可以使用Lambda所在类中的成员变量。

n a。将a按值进行传递。按值进行传递时，函数体内不能修改传递进来的a的拷贝，因为默认情况下函数是const的。**要修改传递进来的a的拷贝**，可以添加mutable修饰符。

n &a。将a按引用进行传递。

n a，&b。将a按值进行传递，b按引用进行传递。

n =，&a，&b。除a和b按引用进行传递外，其他参数都按值进行传递。

n &，a，b。除a和b按值进行传递外，其他参数都按引用进行传递。

② 操作符重载函数参数；

标识重载的()操作符的参数，没有参数时，这部分可以省略。参数可以通过按值（如：(a,b)）和按引用（如：(&a,&b)）两种方式进行传递。

③ 可修改标示符；

mutable声明，这部分可以省略。按值传递函数对象参数时，加上mutable修饰符后，可以修改按值传递进来的拷贝（注意是能修改拷贝，而不是值本身）。

```
QPushButton * myBtn = new QPushButton (this);
QPushButton * myBtn2 = new QPushButton (this);
myBtn2->move(100,100);
int m = 10;

connect(myBtn,&QPushButton::clicked,this,[m] ()mutable { m = 100 + 10; qDebug() << m; });

connect(myBtn2,&QPushButton::clicked,this,[=] () { qDebug() << m; });

qDebug() << m;
```

④ 函数返回值；

->返回值类型，标识函数返回值的类型，当返回值为void，或者函数体中只有一处return的地方（此时编译器可以自动推断出返回值类型）时，这部分可以省略。

⑤ 是函数体；

{}，标识函数的实现，这部分不能省略，但函数体可以为空。

```
Widget::Widget(QWidget *parent)
  : QWidget(parent)
{
  button = new QPushButton("点我",this);
  int a=10;
  int b = 20;
  //槽函数可以是一个Lambda表达式
  //Lambda表达式中[]中写的是= ,代表将上面得函数中的局部变量以值传递的方式传入到Lambda表达式
  //Lambda表达式中[]中写的是& ,代表将上面得函数中的局部变量以引用传递的方式传入到Lambda表达式
  //Lambda表达式中[]中写的是a ,代表将上面得函数中的局部变量a以值传递的方式传入到Lambda表达式
  // Lambda表达式中[]中写的是a,b ,代表将上面得函数中的局部变量a和b以值传递的方式传入到Lambda表达式
  //Lambda表达式中[]中写的是&a ,代表将上面得函数中的局部变量a以引用传递的方式传入到Lambda表达式
  //mutable修饰了,作用是可以在Lambda修改传入变量的值
  //->int 代表Lambda表达式返回值是一个int类型
  connect(button,&QPushButton::clicked,this,[&]() mutable->int{
  a=100;
  qDebug()<<a;
  qDebug()<<b;
  qDebug()<<"点我啊";

  });

}
```