QT原理

简单的使用代码

```
//CStudent.h
#include <QObject>
class CStudent : public QObject
{
    Q_OBJECT
public:
    CStudent(QObject *parent);
    ~CStudent();
    void Test();

public slots:
    void SetAgeSlot(int nAge);
    void SetSexSlot(bool bSex, int nVal);

signals:
void  SetAgeSig(int nAge);
void SetSexSig(bool bSex, int nVal);

private:
    int m_nAge;
};

//CStudent.cpp
#include <QDebug>
void CStudent::Test()
{
    emit SetAgeSig(10);
}

void CStudent::SetAgeSlot(int nAge)
{
    qDebug() << "SetAgeSlot : " << nAge << endl;
}

void CStudent::SetSexSlot(bool bSex, int nVal)
{

}

CStudent::CStudent(QObject *parent)
    : QObject(parent)
{
}

CStudent::~CStudent()
{
}
```

```cpp
//main.cpp
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    CStudent stu(&a);
    QObject::connect(&stu, SIGNAL(SetAgeSig(int)), &stu, SLOT(SetAgeSlot(int)));
    stu.Test();

    return a.exec();
}
```

当编译的时候，moc会在编译器前面扫描文件，然后产生`moc_CStudent.cpp`文件，VS中这个文件是在GEneratedfiles文件夹，QTCreator中这个文件是在编译目录下。

qt_meta_stringdata_CStudent

这个文件含有这个类的RTTI信息。这里面有两个重要的静态成员，一个是`qt_meta_stringdata_CStudent`,

```cpp
struct qt_meta_stringdata_CStudent_t {
    QByteArrayData data[9];
    char stringdata0[67];
};
#define QT_MOC_LITERAL(idx, ofs, len) \
    Q_STATIC_BYTE_ARRAY_DATA_HEADER_INITIALIZER_WITH_OFFSET(len, \
    qptrdiff(offsetof(qt_meta_stringdata_CStudent_t, stringdata0) + ofs \
        - idx * sizeof(QByteArrayData)) \
    )
static const qt_meta_stringdata_CStudent_t qt_meta_stringdata_CStudent = {
    {
QT_MOC_LITERAL(0, 0, 8), // "CStudent"
QT_MOC_LITERAL(1, 9, 9), // "SetAgeSig"
QT_MOC_LITERAL(2, 19, 0), // ""
QT_MOC_LITERAL(3, 20, 4), // "nAge"
QT_MOC_LITERAL(4, 25, 9), // "SetSexSig"
QT_MOC_LITERAL(5, 35, 4), // "bSex"
QT_MOC_LITERAL(6, 40, 4), // "nVal"
QT_MOC_LITERAL(7, 45, 10), // "SetAgeSlot"
QT_MOC_LITERAL(8, 56, 10) // "SetSexSlot"

    },
    "CStudent\0SetAgeSig\0\0nAge\0SetSexSig\0"
    "bSex\0nVal\0SetAgeSlot\0SetSexSlot"
};
```

这个静态成员是`qt_meta_stringdata_CStudent_t`类型的，有两个成员，第一个是个数组，数组中的元素是`QByteArrayData`结构体类型，下面摘取此类型的数据成员，此结构体大小为4个int，即16个字节。这里面只有两个值得注意的成员，在下方代码中已经注释了。

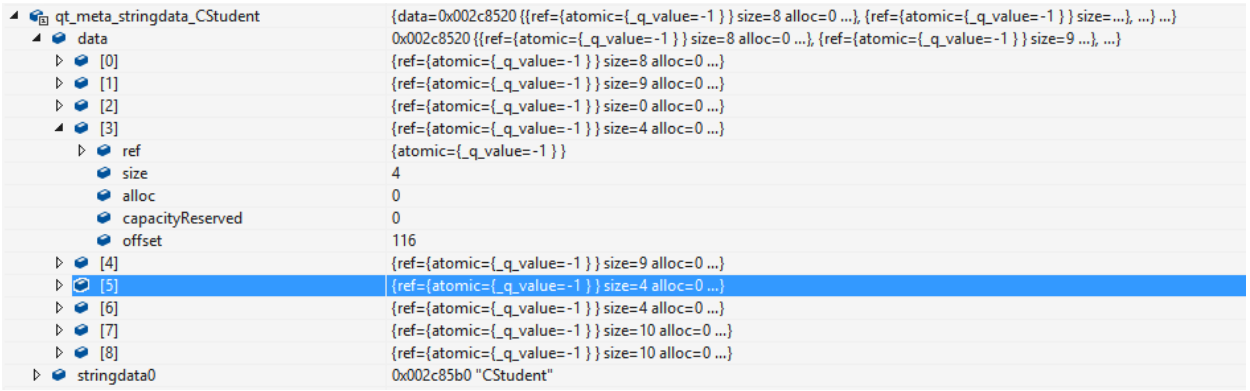
```
struct Q_CORE_EXPORT QArrayData
{
    QtPrivate::RefCount ref;
    int size; //字符串的长度
    uint alloc : 31;
    uint capacityReserved : 1;
    qptrdiff offset; // in bytes from beginning of header
                     // 字符串到本结构体对象首地址的偏移
};
```

`qt_meta_stringdata_CStudent_t`的第二个成员是存放了多个字符串的字符数组。这些数组包含了类名、信号函数和槽函数的名字和参数的名字。
现在回过头来看静态成员`qt_meta_stringdata_CStudent`，

```
static const qt_meta_stringdata_CStudent_t qt_meta_stringdata_CStudent = {
    {
QT_MOC_LITERAL(0, 0, 8), // "CStudent"
QT_MOC_LITERAL(1, 9, 9), // "SetAgeSig"
QT_MOC_LITERAL(2, 19, 0), // ""
QT_MOC_LITERAL(3, 20, 4), // "nAge"
QT_MOC_LITERAL(4, 25, 9), // "SetSexSig"
QT_MOC_LITERAL(5, 35, 4), // "bSex"
QT_MOC_LITERAL(6, 40, 4), // "nVal"
QT_MOC_LITERAL(7, 45, 10), // "SetAgeSlot"
QT_MOC_LITERAL(8, 56, 10) // "SetSexSlot"

    },
    "CStudent\0SetAgeSig\0\0nAge\0SetSexSig\0"
    "bSex\0nVal\0SetAgeSlot\0SetSexSlot"
};
```

这个成员中，`QT_MOC_LITERAL`宏是用来填充`QByteArrayData`结构体的，宏的三个参数，第一个是字符串的id，其实是此元素位于数组的下标索引，第二个是对应字符串在字符数组中的偏移，最后一个参数是字符串的长度。比如字符串"nAge"，它的id是3，位于数组中的索引值为3，位于字符数组中的偏移是19，其长度是4。
下面来看看这个结构体在运行时的内容：



这里仍然拿"nAge"来举例，它的下标索引为3，这里展开了索引为3的元素，其中`size`是字符串的长度4，`offset`是这个元素首地址与字符串"nAge"的偏移距离。这里可以看到，字符串"nAge"与字符数组的首地址的偏移为20，索引为3的元素其本身加上后面还有6个元素，每个元素是16个字节，一共是16*6=96个字节，最后此元素首地址到字符串"nAge"的距离为96+20=116，内存中的值也确实是116。

qt_meta_data_CStudent

第二个静态成员是`qt_meta_data_CStudent`，

```
static const uint qt_meta_data_CStudent[] = {

 // content:
       7,       // revision
       0,       // classname
       0,    0, // classinfo
       4,   14, // methods
       0,    0, // properties
       0,    0, // enums/sets
       0,    0, // constructors
       0,       // flags
       2,       // signalCount

 // signals: name, argc, parameters, tag, flags
       1,    1,   34,    2, 0x06 /* Public */,
       4,    2,   37,    2, 0x06 /* Public */,

 // slots: name, argc, parameters, tag, flags
       7,    1,   42,    2, 0x0a /* Public */,
       8,    2,   45,    2, 0x0a /* Public */,

 // signals: parameters
    QMetaType::Void, QMetaType::Int,    3,
    QMetaType::Void, QMetaType::Bool, QMetaType::Int,    5,    6,

 // slots: parameters
    QMetaType::Void, QMetaType::Int,    3,
    QMetaType::Void, QMetaType::Bool, QMetaType::Int,    5,    6,

       0        // eod
};
```

这个成员看似是个`uint`类型的数组，其实是由两部分内容构成，一部分是跟随注释`content`后面的内容，第二部分则是除去第一部分的内容后剩下的内容，描述了信号槽函数的相关信息。
注释`content`后面跟随着的内容是个`QMetaObjectPrivate`结构体类型

```
struct QMetaObjectPrivate
{
    // revision 7 is Qt 5.0 everything lower is not supported
    enum { OutputRevision = 7 }; // Used by moc, qmetaobjectbuilder and qdbus

    int revision; //QT版本号
    int className; //类名
    int classInfoCount, classInfoData;
    int methodCount, methodData; //信号槽函数的个数，注释sinals内容在qt_meta_data_CStudent
中的偏移
    int propertyCount, propertyData;
    int enumeratorCount, enumeratorData;
    int constructorCount, constructorData;
    int flags;
    int signalCount; //信号的个数
};
```

第二部分内容注释很清楚，因为我的类里面有4个信号槽，2个信号，2个槽函数，所以第二部分会有两个信号，两个槽，这里其实还可以分为两个小部分。上面的小部分是描述了函数的信息，下面小部分是描述了对应函数的参数信息。
这里就拿第一个信号来举例解析。

```
 // signals: name, argc, parameters, tag, flags
       1,    1,   34,    2, 0x06 /* Public */,
       4,    2,   37,    2, 0x06 /* Public */,

 // slots: name, argc, parameters, tag, flags
       7,    1,   42,    2, 0x0a /* Public */,
       8,    2,   45,    2, 0x0a /* Public */,

 // signals: parameters
    QMetaType::Void, QMetaType::Int,    3,
    QMetaType::Void, QMetaType::Bool, QMetaType::Int,    5,    6,

 // slots: parameters
    QMetaType::Void, QMetaType::Int,    3,
    QMetaType::Void, QMetaType::Bool, QMetaType::Int,    5,    6,
```

这个信号的`name`是1，这个1是结构体`qt_meta_stringdata_CStudent`中的中下标为1的数组索引，

```
static const qt_meta_stringdata_CStudent_t qt_meta_stringdata_CStudent = {
    {
QT_MOC_LITERAL(0, 0, 8), // "CStudent"
QT_MOC_LITERAL(1, 9, 9), // "SetAgeSig"
    },
    "CStudent\0SetAgeSig\0\0nAge\0SetSexSig\0"
```

```
    "bSex\0nVal\0SetAgeSlot\0SetSexSlot"
};
```

所以这个信号的名字为`SetAgeSig`， `argc`是参数的个数，这个信号只有一个参数，

```
class CStudent : public QObject
{
signals:
void  SetAgeSig(int nAge);
};
```

`parameters`是参数详细描述位于`qt_meta_data_CStudent`中的偏移，这里的偏移值为34，即注释`signals: parameters`中的第一项，

```
 // signals: parameters
    QMetaType::Void, QMetaType::Int,    3,
    QMetaType::Void, QMetaType::Bool, QMetaType::Int,    5,    6,
```

从这里可以看出，信号的返回值为`void`，参数类型为`int`，后面的3是参数的名字，也是结构体`qt_meta_stringdata_CStudent`中的数组索引

```
static const qt_meta_stringdata_CStudent_t qt_meta_stringdata_CStudent = {
    {
QT_MOC_LITERAL(0, 0, 8), // "CStudent"
QT_MOC_LITERAL(1, 9, 9), // "SetAgeSig"
QT_MOC_LITERAL(2, 19, 0), // ""
QT_MOC_LITERAL(3, 20, 4), // "nAge"
    },
    "CStudent\0SetAgeSig\0\0nAge\0SetSexSig\0"
    "bSex\0nVal\0SetAgeSlot\0SetSexSlot"
};
```

这个参数名字为`nAge`。
其余的内容不晓得是在做什么，暂时放弃。

其它内容

信号实现

Qt中的信号其实只是个函数声明，但moc其实对每个信号都实现了函数体，函数内部通过查表来调用对应的槽函数。

```
void CStudent::SetAgeSig(int _t1)
{
    void *_a[] = { nullptr, const_cast<void*>(reinterpret_cast<const void*>(&_t1)) };
```

```
    QMetaObject::activate(this, &staticMetaObject, 0, _a);
}
```

元对象，或者RTTI

对象的信息保存在静态成员中`staticMetaObject`，通过虚函数`metaObject`来获取。

```
const QMetaObject CStudent::staticMetaObject = {
    { &QObject::staticMetaObject, qt_meta_stringdata_CStudent.data,
      qt_meta_data_CStudent,  qt_static_metacall, nullptr, nullptr}
};


const QMetaObject *CStudent::metaObject() const
{
    return QObject::d_ptr->metaObject ? QObject::d_ptr->dynamicMetaObject() :
&staticMetaObject;
}
```

`QMetaObject`的定义如下：

```
struct Q_CORE_EXPORT QMetaObject
{
    struct { // private data
        const QMetaObject *superdata;  //指向父类的staticMetaObject
        const QByteArrayData *stringdata; //指向信号槽的字符串数组信息
        const uint *data; //指向qt_meta_data_CStudent
        typedef void (*StaticMetacallFunction)(QObject *, QMetaObject::Call, int, void
**);
        StaticMetacallFunction static_metacall;//调用信号槽的函数
        const QMetaObject * const *relatedMetaObjects;
        void *extradata; //reserved for future use
    } d;
};
```

这里面一个重要的成员是`static_metacall`，最终的信号槽函数会在这里被调用:

```
void CStudent::qt_static_metacall(QObject *_o, QMetaObject::Call _c, int _id, void **_a)
{
    if (_c == QMetaObject::InvokeMetaMethod) {
        CStudent *_t = static_cast<CStudent *>(_o);
        Q_UNUSED(_t)
        //调用信号槽函数
        switch (_id) {
        case 0: _t->SetAgeSig((*reinterpret_cast< int(*)>(_a[1]))); break;
        case 1: _t->SetSexSig((*reinterpret_cast< bool(*)>(_a[1])),(*reinterpret_cast<
int(*)>(_a[2]))); break;
        case 2: _t->SetAgeSlot((*reinterpret_cast< int(*)>(_a[1]))); break;
        case 3: _t->SetSexSlot((*reinterpret_cast< bool(*)>(_a[1])),(*reinterpret_cast<
int(*)>(_a[2]))); break;
```

```
        default: ;
        }
    } else if (_c == QMetaObject::IndexOfMethod) {
        int *result = reinterpret_cast<int *>(_a[0]);
        {
            typedef void (CStudent::*_t)(int );
            if (*reinterpret_cast<_t *>(_a[1]) == static_cast<_t>(&CStudent::SetAgeSig))
{
                *result = 0;
                return;
            }
        }
        {
            typedef void (CStudent::*_t)(bool , int );
            if (*reinterpret_cast<_t *>(_a[1]) == static_cast<_t>(&CStudent::SetSexSig))
{
                *result = 1;
                return;
            }
        }
    }
}
```