qt_connect

在 Qt 中，`connect` 函数用于连接信号和槽，使得当信号被发出时，槽函数会被调用。`connect` 函数有几个重载版本，但其基本语法和参数是类似的。以下是常用的 `connect` 函数的参数解释：

基本语法

```
QObject::connect(sender, signal, receiver, method, type);
```

参数解释

1. **sender**（`const QObject*`）：
   - 发出信号的对象的指针。
2. **signal**（`const char*`）：
   - 信号的标识符。通常使用 `SIGNAL` 宏来指定信号，例如：`SIGNAL(signalName(arguments))`。
3. **receiver**（`const QObject*`）：
   - 接收信号的对象的指针，即槽函数所在的对象。
4. **method**（`const char*`）：
   - 槽函数的标识符。通常使用 `SLOT` 宏来指定槽函数，例如：`SLOT(slotName(arguments))`。
5. **type**（`Qt::ConnectionType`）：
   - 连接类型，是一个可选参数，用于指定信号和槽的连接方式。连接类型的默认值是 `Qt::AutoConnection`。常见的连接类型包括：
     - `Qt::AutoConnection`：默认连接类型，根据信号和槽所在的线程自动选择 `Qt::DirectConnection` 或 `Qt::QueuedConnection`。
     - `Qt::DirectConnection`：直接连接，信号发出后立即调用槽函数。
     - `Qt::QueuedConnection`：队列连接，信号发出后将槽函数调用放入接收对象的事件队列中。
     - `Qt::BlockingQueuedConnection`：阻塞队列连接，发送信号的线程会阻塞，直到槽函数执行完毕（仅适用于多线程环境）。
     - `Qt::UniqueConnection`：唯一连接，确保同一个信号与同一个槽之间只有一个连接。

示例代码

```
class Sender : public QObject {
    Q_OBJECT
public:
    void emitSignal() {
        emit mySignal();
    }
signals:
    void mySignal();
};

class Receiver : public QObject {
    Q_OBJECT
public slots:
    void mySlot() {
        qDebug() << "Slot function executed.";
    }
};

int main(int argc, char *argv[]) {
    QCoreApplication a(argc, argv);

    Sender sender;
    Receiver receiver;

    // 连接信号和槽
    QObject::connect(&sender, SIGNAL(mySignal()), &receiver,
SLOT(mySlot()));

    // 发出信号
    sender.emitSignal();

    return a.exec();
}
```

使用新语法

在 Qt 5 及更新版本中，可以使用函数指针来连接信号和槽，这种方式更加类型安全和可读：

```
QObject::connect(sender, &Sender::mySignal, receiver, &Receiver::mySlot);
```

参数解释

1. **sender** (`const QObject*`):
   - 发出信号的对象的指针。
2. **signal** (`Pointer to member function`):
   - 信号的指针，例如 `&Sender::mySignal`。
3. **receiver** (`const QObject*`):
   - 接收信号的对象的指针。
4. **method** (`Pointer to member function`):
   - 槽函数的指针，例如 `&Receiver::mySlot`。
5. **type** (`Qt::ConnectionType`):
   - 连接类型，是一个可选参数，用于指定信号和槽的连接方式。

示例代码

```
class Sender : public QObject {
    Q_OBJECT
public:
    void emitSignal() {
        emit mySignal();
    }
signals:
    void mySignal();
};

class Receiver : public QObject {
    Q_OBJECT
public slots:
    void mySlot() {
        qDebug() << "Slot function executed.";
    }
};

int main(int argc, char *argv[]) {
    QCoreApplication a(argc, argv);

    Sender sender;
    Receiver receiver;

    // 使用函数指针连接信号和槽
```

```
26      QObject::connect(&sender, &Sender::mySignal, &receiver,
    &Receiver::mySlot);
27
28      // 发出信号
29      sender.emitSignal();
30
31      return a.exec();
32  }
```

这种新的连接方式不仅更加直观，而且可以在编译时进行类型检查，提高代码的安全性和稳定性。