**Qt生命周期**

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

Vs2017HelloWorld w;

QLabel label("Hello ,world");

label.show();

w.show();

return a.exec();

}

QApplication 这个对象用于管理 Qt 程序的生命周期，

a.exec()，开启事件循环 ,在a创建和循环，这个中间都可以理解为是一个栈，中间的事件先进后出。

**connect的详细介绍在信号槽**

一个信号槽的调用大约相当于四个模板函数调用

// !!! Qt 5

connect(sender, signal,

receiver, slot);

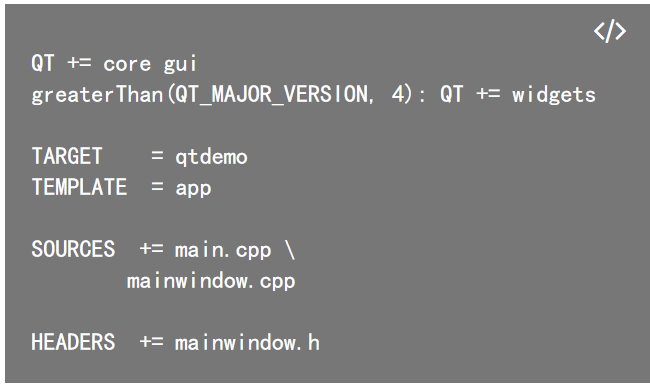
connect()一般会使用前面四个参数，第一个是发出信号的对象，第二个是发送对象发出的信号，第三个是接收信号的对象，第四个是接收对象在接收到信号之后所需要调用的函数。也就是说，当 sender 发出了 signal 信号之后，会自动调用 receiver 的 slot 函数。

**Q\_OBJECT（继承这个类同时书写作用）**

Q\_OBJECT。不管是不是使用信号槽，都应该添加这个宏。这个宏的展开将为我们的类提供信号槽机制、国际化机制以及 Qt 提供的不基于 C++ RTTI 的反射能力。因此，如果你觉得你的类不需要使用信号槽，就不添加这个宏，就是错误的

Qt 5 模块分为 Essentials Modules 和 Add-on Modules

**.pro文件的结构**



简单解释一下 pro 文件。首先，我们定义了 QT，用于告诉编译器，需要使用哪些模块。这些模块都在前面章节中有过介绍。我们通常需要添加 core 和 gui。

第二行，TARGET 是生成的程序的名字。

TEMPLATE 是生成 makefile 所使用的模板，比如 app 就是编译成一个可执行程序，而 lib 则是编译成一个链接库（默认是动态链接库）。

SOURCES 和 HEADERS 顾名思义，就是项目所需要的源代码文件和头文件。现在，我们只需使用默认的 pro 文件即可。以后随着项目的不断增大，pro 文件通常会非常复杂。

**QAction**

定义：

QAction类，抽象出公共的动作，我们把QAction对象添加到菜单，就显示成一个菜单项，添加到工具栏，就显示成一个工具按钮。

包含了图标、菜单文字、快捷键、状态栏文字、浮动帮助等信息

编译规则

**通过继承QObject类，我们可以很方便地获得这些特性。 获得moc编译机制**

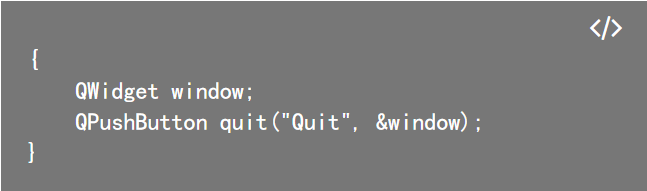
Qt 先使用一个叫做 moc（Meta Object Compiler，元对象编译器）的工具，先对 Qt 源代码进行一次预处理（***注意，这个预处理与标准 C++ 的预处理有所不同。Qt 的 moc 预处理发生在标准 C++ 预处理器工作之前，并且 Qt 的 moc 预处理不是递归的。***），生成标准 C++ 源代码，

Qt 使用 moc，为标准 C++ 增加了一些特性：

信号槽机制 强大的事件机制以及事件过滤器； 基于上下文的字符串翻译机制（国际化），也就是 tr() 复杂的定时器实现， 层次化的可查询的对象树，提供一种自然的方式管理对象关系。 智能指针（QPointer） 能够跨越库边界的动态转换机制。

对象树概念

在一定程度上解决了内存问题，



他能够保证，在父组件消失的时候子组件全部没有，设置顺序 是父组件最先创建，子组件在创建，指定父组件（记得这个顺序，如果子组件先创建，子组件会被delete两次，报错）

**菜单栏、工具栏和状态栏**

对于没有这些函数的QDialog或者QWidget怎么做呢？要记得，QToolBar以及QStatusBar都是QWidget的子类，因此我们就可以将其结合布局管理器添加到另外的QWidget上面。QLayout布局提供了setMenuBar()函数，可以方便的添加菜单栏。具体细节还是详见文档。