本篇是QT系列的第一篇记录，复盘学习和使用QT过程中值得记录的知识。

QT是跨平台的C++ GUI框架，那么就选择linux平台好了。以前都是用virtual box装个ubuntu，现在Windows11提供了WSL2，基本上可以满足开发需求。现在有很多的windows terminal程序可以远程连接WSL，甚至直接用vscode也可以。至于linux的发行版，选择最新的ubuntu2004就好。

为了完成的QT版的hello world，首先要装好QT的开发环境。

安装C++编译环境，原因是微软商店提供的ubuntu啥都没有

|  |
| --- |
| sudo apt update && sudo apt upgrade -y  sudo apt install build-essential -y |

安装OpenGL库

|  |
| --- |
| sudo apt install libgl1-mesa-dev -y |

安装其他依赖库

|  |
| --- |
| sudo apt install libxkbcommon-x11-dev libx11-xcb-dev libxcb-xkb-dev libxcb-icccm4-dev libxcb-keysyms1-dev libxcb-render-util0-dev libxcb-util-dev libxcb-xinerama0-dev -y |

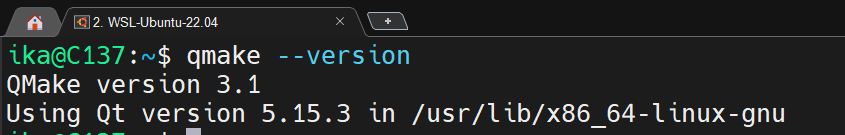
安装QT

WSL的包管理器是提供了QT的预编译版本可以直接安装

|  |
| --- |
| sudo apt install qt5-default qtcreator -y |

验证安装

|  |
| --- |
| qmake --version |

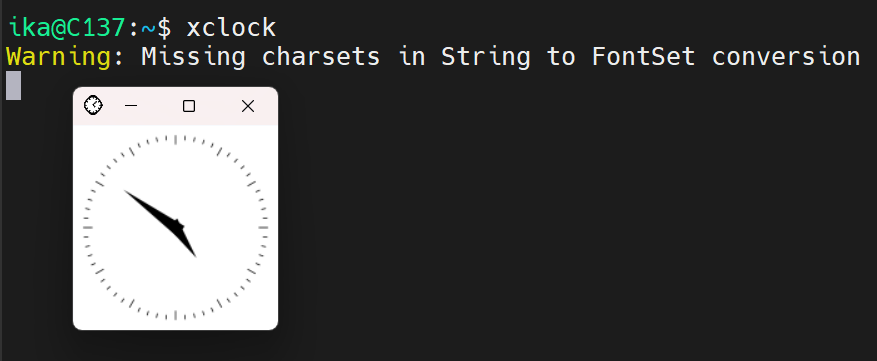


关于qmake，这是QT官方提供编译源文件的工具。不过，我习惯了使用CMAKE，而且QT也支持，那么就继续用CMAKE了。这里遇到了第一个问题，在高版本的ubuntu中移除了qt5-default。可以通过其他方式安装QT5。

安装核心开发包

|  |
| --- |
| sudo apt install qtbase5-dev qt5-qmake qtchooser -y |

好了，现在linux的开发环境就基本搞定了。不对，还差一个可视化程序，用vcxsrv就可以了，下载好后执行默认安装就可以。安装好后，输入xclock出现一个时钟界面，表示安装成功。



现在开始编写QT程序。选择CMAKE构建。

|  |
| --- |
| cmake\_minimum\_required(VERSION 3.12)  set(CMAKE\_CXX\_STANDARD 17)  set(CMAKE\_AUTOMOC ON)  set(CMAKE\_AUTOUIC ON)  set(CMAKE\_AUTORCC ON)  project(qtguide)  #cmake通过这个变量来寻找路径  set(Qt5\_DIR /usr/lib/x86\_64-linux-gnu/cmake/Qt5)  find\_package(Qt5 COMPONENTS Core Widgets REQUIRED)  file(GLOB sources \*.cpp)  add\_executable(qtguide ${sources})  target\_link\_libraries(qtguide PRIVATE Qt5::Core Qt5::Widgets) |

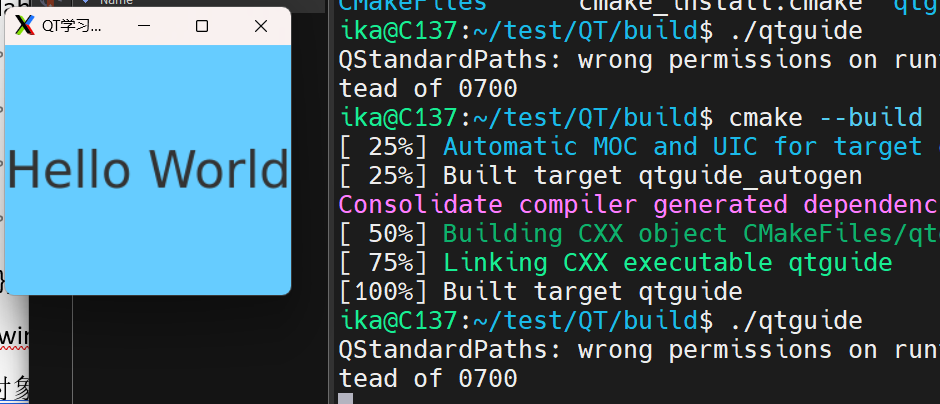
CMAKE通过Qt5\_DIR来寻找\*\*\*.cmake路径，这个cmake文件会在find\_package

用到。这个命令会用来寻找需要使用的QT核心库，用于后面链接使用，就是那个Qt5::\*\*的内容。

接下来是主程序。

|  |
| --- |
| #include <QApplication>  #include <QMainWindow>  #include <QDebug>  #include <QLabel>  //QObject类是所有QT对象的基类，QObject类提供了对象树的支持  //QT维护了一颗对象树  //对象树的作用：当父对象被释放时，子对象也会被释放  //对象树的好处：不需要手动释放对象，不需要担心内存泄漏  //对象树的注意事项：当对象树中的对象被释放时，会自动释放子对象，但是子对象的指针并不会被置为nullptr  //因此手动释放子对象会导致内存泄漏（C++11中，delete一个nullptr是没有任何effect的）  //QObject的派生关系  //QObejct->QWidget->QMainWindow, QLabel, QPushButton, QComboBox等  //QT中的main函数  *int* main(*int* *argc*, *char* \*\**argv*){      //单例模式实现类对象，每个Qt程序只能有一个QApplication对象      QApplication app(*argc*, *argv*);      {          QObject obj1;          QObject \*obj2 = new QObject();          obj2->setParent(&obj1); //obj2的父对象是obj1      }//这里obj1和obj2都会被释放      //只能有一个主窗口      QMainWindow window; //这里面就是一个事件循环      window.setWindowTitle("QT学习");      window.setGeometry(500, 500, 800, 600);      QLabel \*label = new QLabel;//显示创建一个对象，没有指定父对象,这个对象不会被释放      label->setText("Hello World");      //设置对齐方式,参数是个枚举值，可以组合使用      label->setAlignment(Qt::AlignCenter | Qt::AlignVCenter);      //层叠样式表，对应前端的CSS，qt中叫做qss      //C++14中的原始字符串，R"()"，里面的内容不会被转义      label->setStyleSheet(R"(          QLabel{          color: #333333;          font-size: 40px;          background-color: #66ccff      })");      window.setCentralWidget(label); //label作为窗口的中心控件，共用了同一个父对象widget      window.show();      return app.exec();  } |

main函数是整个程序的入口点，对C++程序来说不管怎么封装这是不变的。



整个程序运行后就是这样。

一个简单的QT程序有这两行语句就足够了。

QApplication app(*argc*, *argv*);

app.exec();

这是一个单例模式实现的类对象，每个程序只有一个QApplication对象，其他地方使用的话可以直接获取。QMainWindow类创建一个窗口，调用成员函数show可以展示这个窗口，不过是空白的。setWindowTitle成员函数可以给窗口加个标题，setGeometry成员函数可以调整程序启动时候的窗口的初始位置和大小，坐标系参考屏幕的左上角。

设置好窗口后，给窗口加个label。区别于QMainWindow，QLabel对象创建在堆上，调用默认的无参构造函数。成员函数setText设置名字，setAlignment设置对齐方式，参数是一些枚举值，对齐方式包括居中，左对齐这些，支持组合使用。setStyleSheet成员函数设置层叠样式表，对应前端的QSS，可以叫QSS。函数参数是个string类型，这里使用了C++14引入的原始字符串，R”()”中的内容不会被转义。最后，将QLabel设置为QMainWindow的中心部件。这样浪费了整个窗口，好在QT支持在窗口布局（layout），后面叙述。setCentralWidget传入了一个QLabel对象，产生了一个side effect，QLabel和QMainWindow共用了同一个父对象。

这里涉及到了QT的对象树。QT中的所有类都有一个基类QObject，QObject类提供了对象树的支持。在对象树中的对象，当父对象被销毁的时候，最对象也会被销毁。但是，有个缺陷，就是最对象被销毁后不会被置为nullptr。所以，会有内存泄漏的风险。（C++11中，delete一个nullptr不会有任何side effect）。因此，即便显示的new了一个QLabel对象，但是不会有任何内存泄漏。因为程序结束的时候，栈上的QMainWindow对象会被销毁，对象树上的其他对象也会被销毁。既然QObject是基类，它的派生关系是什么。QObejct->QWidget->QMainWindow, QLabel, QPushButton, QComboBox等。就是这样，大部分用到的类都是QWidget派生的。

好了，本次记录到这里。

附加一个遇到的小问题，VSCODE找不到include的QT源文件该怎么办。在.vscode中的c\_cpp\_properties.json中手动配置，如果没有这个文件就创建一个。

|  |
| --- |
| {  *"configurations"*: [        {  *"name"*: "Linux",  *"includePath"*: [            "${workspaceFolder}/\*\*",            "/usr/include/x86\_64-linux-gnu/qt5",            "/usr/include/x86\_64-linux-gnu/qt5/QtCore",            "/usr/include/x86\_64-linux-gnu/qt5/QtWidgets"          ],  *"defines"*: [],  *"compilerPath"*: "/usr/bin/gcc",  *"cStandard"*: "c11",  *"cppStandard"*: "c++17",  *"intelliSenseMode"*: "gcc-x64"        }      ],  *"version"*: 4    } |

配置好includePath后，VSCODE就会在指定的目录里搜索源文件，在使用的时候就可以看到代码提示了。