一位大佬对于 Qt 学习的最全总结（三万字干货）

胡安迪 嵌入式 ARM 今天

作者无私分享了一百多个自定义控件以及几本 Qt 的书籍（C++ Qt），嵌入式 ARM 特此引用这个大佬的干货，左下角可阅读原文进入 GitHub 原分享地址。

一、个人总结的经验语录

1. 学习编程是一个渐变的过程，1 年精通，3 年熟悉，8 年入门，10 年懵逼，15 年颈椎康复指南，30 年灰飞烟灭。
2. 老板或者客户：什么需求我不清楚，想要什么效果也不知道，但是你做出来什么是我不想要的，这个我很清楚！杀死一个程序员很简单，改几次需求就可以了！
3. 一个控件从 0 到基本可用，如果差 3 条街的话，从可用到好用，还差着 30 条街。控件如此，软件系统亦如此。
4. 一行代码，1 块钱，知道怎么写，100 块钱。好比医院的手术医生，手术所用药物 100 块钱，但是一台手术几千块，为什么，因为只有他知道怎么手术，同样是开刀，知道怎么开是最值钱的！
5. 很喜欢鬼谷子一段话：我们做了很多事情，不必在意答案，因为没有答案，如果答案是别人所需要的，我们就是在走别人的路，但是，我们是要给别人答案的人。没有什么是绝对的，答案也不会只有一个，所以论证答案，本身就是愚蠢的。
6. 关于读书无用论，学历只是很小的一方面，其实最重要的是三个能力：自主学习的能力，解决问题的能力，人际交往的能力！三方面都厉害，哪怕幼儿园毕业，都是顶层人物。读书，培养的就是这三方面的能力。只是有些人培养好了，有些人没有而已。
7. 软件写得再好，代码再牛逼，界面再漂亮，架构再厉害，不能解决用户的实际需求，不能卖钱或者赚不到钱，都是白搭！人，真的需要找对行业和运气！
8. 不懂技术的，往往把技术看的太简单，比如很多老板；懂技术的，很多时候把技术想的太复杂，比如程序员；

9. 什么叫享受人生？在周末的下午，在家里大厅里打开一部电影，开瓶红酒，拿出高脚杯倒上半杯慢饮，偶尔看看窗外怡人的风景，你会突然发现，还是特么去写几行代码更有意思，NND！
10. 我完全是兴趣主导，老板不给我钱，我也要写好代码！白天干，晚上干，周一周五干，周末继续干！编程已经深入我的基因，深入我的骨髓，深入我的灵魂！每当解决一个程序问题，比别人玩王者荣耀拿了第一名还开心！要想想，能干自己感兴趣的事情，像写诗一样写代码，玩游戏一样的开心心情，还能领工资！真他妈爽！没有比这个更爽的事情！
11. 不要以技术多牛逼自豪，赚了多少钱自豪，而要以名族自豪，文化自信，国家自强！
12. 我现在的模式是，通过卖代码，然后了解各行各业，然后探究需求，然后找准机会，下手，直接撸立马干！A 项目孵化 B 项目，B 项目孵化 C 项目。卖只是引子，顶多赚点零花钱，真实意图是定制。
13. 每当看到各位拼命努力学习的时候，我就慌得一逼，所以我只有努力的学习才能追赶各位大佬的脚步，然后不被时代所抛弃，只能用勤奋去换取知识技术的积累。我不聪明，但我会很努力--狂奔的蜗牛！
14. 别看我 QtWidget 水平比很多人高一点，其实全国也就几万名开外，你们是不知道天外有天，人外有人！应该到处走走，不要坐井观天！因为我遇到过几十个水平比我厉害的多了去了的，这还仅仅是冰山一角！真正的大佬一般不混群也不混论坛的，低调得很！可能这就是所谓的闷声发大财。
15. 如果我们把期限设定为人的一生，这就意味着年轻人应该多探索，到了后期就要更多的专注于收获。
16. 有的人等退休做喜欢的事，有的人做一辈子喜欢的事，不想退休。
17. 也许你现在做的事情看不到成果，但不要害怕，你并不是没有成长，而是在扎根。
18. 要么选择自己努力摸索出一套成长和赚钱的思维模式，要么选择学习别人已经成功和赚钱的思维模式并加以改造成自己的模式。
19. 码农最重要的是学会沟通，而不是写代码，不要抱怨没人回答你的问题，首先需要学会如何提问，其他人才能更好的理解你的问题。
20. 回答你的问题是情义，不理你是我的权力，没有人有义务要帮助你，你是在寻求帮助，态度好点，但也不要委屈自己，最后，如果你啥都不会，或者只会一点基础，不要期望我们告诉你怎么做！建议买本基础书籍多看看多练练。
21. 如何有效地提问:你做了什么，怎么做的；你查过什么资料，怎么说的；你是怎么设计的；贴出你的代码；编译错误是什么；代码截图，错误截图；结果有什么错，你期望的结果是什么；你是怎么想的；
22. 代码如尿崩，谁与我争疯！用生命写代码，用灵魂做界面！
23. 一个优秀的程序员写出诗一样的代码，一个平庸的程序员写出屎一样的代码。
24. 虽然楼主没有讲很多底层原理，但仅凭那寥寥数字，就让我心中澎湃不已，对楼主瞬间佩服的五体投地，有如醍醐灌顶之效，暮然回首发现自己已经没有什么技术可言，真是关公面前耍大刀班门弄斧。
25. 当年秦始皇无意中得到了两颗长生不老药，他自己服了一颗，还剩一颗让我服，我都没服，就服你！

26. 此情此景我想吟诗一首：误入 IT 悔无声，单枪匹马夜挑灯，一入 Qt 深似海，从此脂粉不沾身。
27. 此情此景我想再吟诗一首：Qt 群里行人稀，常有车手较高低，如今车道依旧在，不见当年老司机。
28. 在我们平时编程过程中，应该尽量压制人的某些本性，尽量做到不找理由、扛责任、降低攻击性、不断学习他人优点，并降低自己不动脑子思考的时间、不说谎、换位思考、尽量不自私、绝不窝里横。
29. 生活中人际提高一些，主要是说话前想一想、花花轿子人抬人、有来有往才是关系、努力胸怀大点，要敢于分钱、带小弟吃肉，给老大喂肉、能带他人一起成长。
30. 工作上，长期有规划，短期有计划；想明白就去干，执行力；复杂问题简单化，简单问题复杂化，大逻辑好；做事必有回应；工程和代码能力要好；做事至少做到 60 分；吹牛逼的本领要好。
31. 我接触过上海这边至少几十个老板，大大小小的，也有低学历的，但是都有个共性，就是非常喜欢学习，而且做人非常到位，无论自己身价多高，都是非常恭敬谦卑。如果想要成功，就必须培养好这几个共性。
32. 如何成为一名优秀的独立开发者，给出以下几点建议：做一些你喜欢做的东西；快速构建好产品，并及早放到市场上验证（精益创业）；不要在技术上浪费太多时间，定好发布日期便执行；不懂技术也可以打造一款好产品；如果你打算开始单干，记得打造好你的个人品牌；亲自为客户提供邮件支持，至少在一开始时要这么做；顾客至上；仔细思考并设定合理的目标，先实现产品再谈其他。
33. 大学生应该放下自己的身段学历等，主动积极做事情，端茶倒水搞卫生也无妨，先活下去再说，不分高低贵贱，忍辱负重不断积累资本，方能前途无量。假如你的运气更好，不但找到适合你的职业方向，同时又碰到能给你鼓励和提供发展机会发展空间的老板，那就可以将这份职业当成事业来做了！
34. 多换位思考，站在对方的角度思考问题，学会一套圆滑的处事高招。懂得化悲痛为力量，化干戈为玉帛，化腐朽为神奇！
35. 人一旦从一个巨大的阴霾中走出来，你会发现世界原来这么美好，前途一片大好光明。
36. 360 行，行行转程序员，现在大量的人员看到程序员职位赚钱吃香，都转行过来，有报班的有自学的，殊不知，一个人适不适合写程序，在你还是一颗精子的时候就决定了，如果自己没有编程的手感和意识，这条路太难了！
37. 我不是针对谁，我是说在座的各位，注意，是在座的各位，除了我都是大佬！
38. 一个人最开心的时候不是在你拿到钱的一刹那，而是在你知道你可以拿这么多钱的时候！
39. 上海没房的话，压根就没有存在感，所以伙伴们还是早点回老家创业吧，有句话说的好，上海挣钱上海花，一分别想寄回家。
40. 选择公司的时候一定要注意，要看一看公司的氛围，一家正常的公司，一般没有那么多乱七八糟形式化的东西，不会一大早喊口号，打鸡血，也不

会早请示，晚汇报，因为正经公司追求的是利润，做企业要赚钱的，大家没有那么多时间去浪费。

41. Qt 编程一时爽，一直 Qt 一直爽。左边跟我一起画个龙，在你右边画一道彩虹！
42. 人一定要有才华，这是所有自信的根基和源泉。
43. 输了你，赢了整个世界又如何！连心爱的人都留不住，代码写的再溜又能怎样?
44. 当你对编程有着浓厚兴趣的时候，你就会因为这种兴趣的吸引而全身心的投入进去，也正是这种投入与忍耐，恒心和坚持，让你变得更加牛逼。
45. 世界上只有两种区块链，一种在技术天才们的头脑里，还没有落地，另一种在中国人的微信群里，已经在做交易。
46. 技术人员做技术时间久了，很容易陷入一个技术思维陷阱，以为技术就是整个世界整个生活，老子技术天下第一，其他人写的都是垃圾的感觉，很多人觉得自己 5 年 10 年的时候技术很牛逼，自我感觉良好，非常蔑视瞧不起那些做销售做市场做商业的技术人员，其实大错特错，这些人全身心投入到技术上，技术深度未必比你差。建议技术人员头 5-10 年，可以全身心投入技术研发，深扎根，打牢基础，过了这个阶段以后，可以适当的往市场、需求、商业、管理等更高层次方面拓展，格局慢慢变大而不仅限于技术方面。
47. 有一种生意最赚钱，那就是教别人如何赚钱，让别人赚钱。有一种成功最成功，那就是让别人成功从而让自己获得成功！
48. 在编程这条技术路线过程中，除了要耐得住寂寞，扛得住诱惑，技术深扎根以外，更需要自我总结、自我推动、自我激发、自我生产、自我疗伤、自我成长。学会左手温暖右手！
49. 活在当下都不容易，多少人为了养家糊口远离他乡和父母小孩在外打工奋斗，住着简易的房子，甚至有些还不足 10 平米，一边还着房贷车贷，一边交着房租，而且很多人还是交着两个房租，一个是家里边为了小孩在县城读书租房的，一个是自己在外打工租房的，但是各位心中依然憧憬着美好的未来，致敬每一个在外打工漂泊的人。
50. 程序员不要随意吹牛逼，这个月薪五万那个年薪百万，还有个刚毕业就月薪三万，当你跟着一起瞎几把吹牛逼的时候，有一天你会突然发现只有你在吹牛逼，别人说的都是真的！
51. 如果不是为了荣归故里，谁他么愿意 ” 抛妻弃子 “ “漂泊流浪” ” 远走他乡 “。
52. 我感觉我的技术遇到了瓶颈，该会的都已经会了，不会的再怎么学也不会，估计这辈子也就这样了，顶多就是学会另外一门语言框架做做过的事情，没有或者很难有质的突破！
53. 当你离开大学走上社会以后，你会发现，大学的就业指导课的老师很可能自己都没去社会就过业，大学的计算机老师很可能没做过商业项目的开发，入门甚至还没入门的水平！
54. 中间部分持续增加中，欢迎关注，敬请期待！
55. 最后祝大家头发浓密，睡眠良好，情绪稳定，财富自由！

二、程序员懵逼瞬间

1. 不加代码前运行的好好的，手贱加了几行代码，就不行了，然后把加的这几行代码注释掉重新彻底编译，他娘的也还是不行，懵逼中，关键是他娘的真的回不去了。
2. 昨天还运行的好好的，今天一开机运行就不行了，关键是什么都没干。一般这种情况，回去睡一觉第二天就莫名其妙的好了。
3. 编译后明明只有 1 个错误，找到错误的地方改了改，信心满满按下 F5，尼玛，几百个错误出来了。

三、开发经验总结

1. 当编译发现大量错误的时候，从第一个看起，一个一个的解决，不要急着去看下一个错误，往往后面的错误都是由于前面的错误引起的，第一个解决后很可能都解决了。
2. 定时器是个好东西，学会好使用它，有时候用 QTimer::singleShot 可以解决意想不到的问题。
3. 打开 creator，在构建套件的环境中增加 MAKEFLAGS=-j8，可以不用每次设置多线程编译。珍爱时间和生命。新版的 QtCreator 已经默认就是 j8。
4. 如果你想顺利用 QtCreator 部署安卓程序，首先你要在 AndroidStudio 里面配置成功，把坑全部趟平。
5. 很多时候找到 Qt 对应封装的方法后，记得多看看该函数的重载，多个参数的，你会发现不一样的世界，有时候会恍然大悟，原来 Qt 已经帮我们封装好了。
6. 可以在 pro 文件中写上标记版本号+ico 图标（Qt5 才支持）

```
VERSION  = 2020.10.25RC_ICONS = main0.ico
```

1. 管理员运行程序，限定在 MSVC 编译器。

```
QMAKE_LFLAGS  +=  /MANIFESTUAC:"level='requireAdministrator'
uiAccess='false'"  #  以  管  理  员  运  行  QMAKE_LFLAGS  +=
/SUBSYSTEM:WINDOWS,"5.01" #VS2013 在 XP 运行
```

1. 运行文件附带调试输出窗口 CONFIG += console pro
2. 绘制平铺背景 QPainter::drawTiledPixmap, 绘制圆角矩形 QPainter::drawRoundedRect(),而不是 QPainter::drawRoundRect();
3. 移除旧的样式

```
//移除原有样式 style()->unpolish(ui->btn);//重新设置新的该控件的样式。
style()->polish(ui->btn); 获取类的属性 const QMetaObject
*metaobject = object->metaObject();int count = metaobject-
>propertyCount();for (int i = 0; i < count; ++i)
{      QMetaProperty metaproperty = metaobject->property(i);
const char *name = metaproperty.name();     QVariant value =
object->property(name);    qDebug() << name << value;}
```

1. Qt 内置图标封装在 QStyle 中，大概七十多个图标，可以直接拿来用。

```
SP_TitleBarMenuButton,SP_TitleBarMinButton,SP_TitleBarMaxButt
on,SP_TitleBarCloseButton,SP_MessageBoxInformation,SP_Message
BoxWarning,SP_MessageBoxCritical,SP_MessageBoxQuestion,...
```

1. 根据操作系统位数判断加载

```
win32 {         contains(DEFINES, WIN64) { DESTDIR = $$
{PWD}/../../bin64               }    else    {    DESTDIR    =
$${PWD}/../../bin32 }}
```

1. Qt5 增强了很多安全性验证，如果出现 setGeometry: Unable to set geometry，请将该控件的可见移到加入布局之后。
2. 可以将控件 A 添加到布局，然后控件 B 设置该布局，这种灵活性大大提高了控件的组合度，比如可以在文本框左侧右侧增加一个搜索按钮，按钮设置图标即可。

```
QPushButton *btn = new QPushButton;btn->resize(30, ui->lineEdit->height());QHBoxLayout *layout = new QHBoxLayout(ui->lineEdit);layout->setMargin(0);layout->addStretch();layout->addWidget(btn);
```

1. 对 QLCDNumber 控件设置样式，需要将 QLCDNumber 的 segmentstyle 设置为 flat。
2. 巧妙的使用 findChildren 可以查找该控件下的所有子控件。findChild 为查找单个。

```
// 查找指定类名 objectName 的控件 QList<QWidget *> widgets = parentWidget.findChildren<QWidget *>("widgetname");// 查找所有 QPushButtonQList<QPushButton *> allPButtons = parentWidget.findChildren<QPushButton *>();//查找一级子控件,不然会一直遍历所有子控件 QList<QPushButton *> childButtons = parentWidget.findChildren<QPushButton *>(QString(), Qt::FindDirectChildrenOnly);
```

1. 巧妙的使用 inherits 判断是否属于某种类。

```
QTimer *timer = new QTimer; // QTimer inherits QObjecttimer->inherits("QTimer"); // returns truetimer->inherits("QObject"); // returns truetimer->inherits("QAbstractButton"); // returns false
```

1. 使用弱属性机制，可以存储临时的值用于传递判断。可以通过 widget->dynamicPropertyNames()列出所有弱属性名称，然后通过 widget->property("name")取出对应的弱属性的值。
2. 在开发时，无论是出于维护的便捷性，还是节省内存资源的考虑，都应该有一个 qss 文件来存放所有的样式表，而不应该将 setStyleSheet 写的到处都是。如果是初学阶段或者测试阶段可以直接 UI 上右键设置样式表，正式项目还是建议统一到一个 qss 样式表文件比较好，统一管理。
3. 如果出现 Z-order assignment: is not a valid widget.错误提示，用记事本打开对应的 ui 文件，找到为空的地方，删除即可。
4. 善于利用 QComboBox 的 addItem 的第二个参数设置用户数据，可以实现很多效果，使用 itemData 取出来。
5. 如果用了 webengine 模块，发布程序的时候带上 QtWebEngineProcess.exe+translations 文件夹+resources 文件夹。

6. 默认 Qt 是一个窗体一个句柄，如果要让每个控件都拥有独立的句柄，设置下 a.setAttribute(Qt::AA_NativeWindows);
7. Qt+Android 防止程序被关闭。

```
#if defined(Q_OS_ANDROID)QAndroidService a(argc, argv);return a.exec()#elseQApplication a(argc, argv);return a.exec();#endif
```

1. 可以对整体的指示器设置样式，例如 ::down-arrow,::menu-indicator{}::up-arrow:disabled,::up-arrow:off{}。
2. 可以执行位置设置背景图片。

```
QMainWindow > .QWidget { background-color: gainsboro; background-image: url(:/images/pagefold.png); background-position: top right; background-repeat: no-repeat}
```

1. 嵌入式 linux 运行 Qt 程序 Qt4 写法：./HelloQt -qws & Qt5 写法：./HelloQt --platform xcb
2. Qtcreator 软件的配置文件存放在：C:\Users\Administrator\AppData\Roaming\QtProject，有时候如果发现出问题了，将这个文件夹删除后打开 creator 自动重新生成即可。
3. QMediaPlayer 是个壳，依赖本地解码器，视频这块默认基本上就播放个 MP4，如果要支持其他格式需要下载 k-lite 或者 LAV Filters 安装即可（WIN 上，其他系统上自行搜索）。如果需要做功能强劲的播放器，初学者建议用 vlc、mpv，终极大法用 ffmpeg。
4. 判断编译器类型、编译器版本、操作系统。

//GCC 编译器#ifdef __GNUC__#if __GNUC__ >= 3 // GCC3.0 以上
//MSVC 编译器#ifdef _MSC_VER#if _MSC_VER >=1000 // VC++4.0 以上#if _MSC_VER >=1100 // VC++5.0 以上#if _MSC_VER >=1200 // VC++6.0 以上#if _MSC_VER >=1300 // VC2003 以上#if _MSC_VER >=1400 // VC2005 以上#if _MSC_VER >=1500 // VC2008 以上#if _MSC_VER >=1600 // VC2010 以上#if _MSC_VER >=1700 // VC2012 以上#if _MSC_VER >=1800 // VC2013 以上#if _MSC_VER >=1900 // VC2015 以上
//Borland C++#ifdef __BORLANDC__
//Cygwin#ifdef __CYGWIN__#ifdef __CYGWIN32__

```
//mingw#ifdef __MINGW32__
//windows#ifdef    _WIN32                //32bit#ifdef    _WIN64
//64bit#ifdef _WINDOWS      //图形界面程序#ifdef _CONSOLE      //
控制台程序//Windows（95/98/Me/NT/2000/XP/Vista）和Windows CE都定
义了#if (WINVER >= 0x030a)      // Windows 3.1以上#if (WINVER >=
0x0400)           //  Windows  95/NT4.0  以 上 #if  (WINVER  >=
0x0410)      // Windows 98以上#if (WINVER >= 0x0500)      //
Windows Me/2000以上#if (WINVER >= 0x0501)      // Windows XP以
上 #if  (WINVER  >=  0x0600)           //  Windows  Vista  以
上 //_WIN32_WINNT  内 核 版 本 #if  (_WIN32_WINNT  >=  0x0500)  //
Windows 2000以上#if (_WIN32_WINNT >= 0x0501) // Windows XP以上
#if (_WIN32_WINNT >= 0x0600) // Windows Vista以上
```

1. 在pro中判断Qt版本及构建套件位数

```
#打印版本信息message(qt version: $$QT_VERSION)#判断当前qt版本号
QT_VERSION  =  $$[QT_VERSION]QT_VERSION  =  $$split(QT_VERSION,
".")QT_VER_MAJ  =  $$member(QT_VERSION,  0)QT_VER_MIN  =  $
$member(QT_VERSION, 1)#下面是表示 Qt5.5greaterThan(QT_VER_MAJ,
4) {greaterThan(QT_VER_MIN, 4) {#自己根据需要做一些处理}}
#QT_ARCH 是 Qt5 新增的，在 Qt4 上没效果#打印当前 Qt 构建套件的信息
message($$QT_ARCH)#表示arm平台构建套件 contains(QT_ARCH, arm)
```

{}#表示 32 位的构建套件 contains(QT_ARCH, i386) {}#表示 64 位的构建套件 contains(QT_ARCH, x86_64) {}

1. Qt 最小化后恢复界面假死冻结，加上代码

```
void showEvent(QShowEvent *e)
{    setAttribute(Qt::WA_Mapped);    QWidget::showEvent(e);}
```

1. 获取标题栏高度：style()->pixelMetric(QStyle::PM_TitleBarHeight); PM_TitleBarHeight 点进去你会发现新大陆。
2. 设置高分屏属性以便支持 2K4K 等高分辨率，尤其是手机 app。必须写在 main 函数的 QApplication a(argc, argv);的前面。

```
#if (QT_VERSION > QT_VERSION_CHECK(5,6,0))
QGuiApplication::setAttribute(Qt::AA_EnableHighDpiScaling);#endif
```

1. 如果运行程序出现 Fault tolerant heap shim applied to current process. This is usually due to previous crashes. 错误。办法：打开注册表，找到 HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Layers\，选中 Layers 键值，从右侧列表中删除自己的那个程序路径即可。
2. Qt 内置了 QFormLayout 表单布局用于自动生成标签+输入框的组合的表单界面。
3. qml 播放视频在 linux 需要安装 sudo apt-get install libpulse-dev。
4. 可以直接继承 QSqlQueryModel 实现自定义的 QueryModel，比如某一列字体颜色，占位符，其他样式等，重写 QVariant CustomSqlModel::data(const QModelIndex &index, int role) const。
5. Qt5 以后提供了类 QScroller 直接将控件滚动。

```
// 禁用横向滚动条 ui->listWidget->setHorizontalScrollBarPolicy(Qt::ScrollBarAlwaysOff);// 禁用纵向滚动条 ui->listWidget->setVerticalScrollBarPolicy(Qt::ScrollBarAlwaysOff);// 设置横向按照像素值为单位滚动 ui->listWidget->setHorizontalScrollMode(QListWidget::ScrollPerPixel);// 设置纵向按照像素值为单位滚动 ui->listWidget->setVerticalScrollMode(QListWidget::ScrollPerPixel);// 设置滚动对象以及滚动方式为鼠标左键拉动滚动 QScroller::grabGesture(ui->listWidget, QScroller::LeftMouseButtonGesture);// 还有个 QScrollerProperties 可以设置滚动的一些参数
```

1. 如果使用 sqlite 数据库不想产生数据库文件，可以创建内存数据库。

```
QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");db.setDatabaseName(":memory:");
```

1. 清空数据表并重置自增 ID，sql = truncate table table_name。
2. Qtchart 模块从 Qt5.7 开始自带，最低编译要求 Qt5.4。在安装的时候记得勾选，默认不勾选。使用该模块需要引入命名空间。

```
#include <QChartView>QT_CHARTS_USE_NAMESPACEclass CustomChart : public QChartView
```

1. QPushButton 左对齐文字，需要设置样式表 QPushButton{text-align:left;}
2. QLabel 有三种设置文本的方法，掌握好 Qt 的属性系统，举一反三，可以做出很多效果。

```
ui->label->setStyleSheet("qproperty-text:hello;");ui->label->setProperty("text", "hello");ui->label->setText("hello");
```

1. 巧妙的用 QEventLoop 开启事件循环，可以使得很多同步获取返回结果而不阻塞界面。QEventLoop 内部新建了线程执行。

```
QEventLoop loop;connect(reply, SIGNAL(finished()), &loop, SLOT(quit()));loop.exec();
```

1. 多种预定义变量 #if (defined webkit) || (defined webengine)，去掉生成空的 debug 和 release 目录 CONFIG -= debug_and_release。

2. 新版的 Qtcreator 增强了语法检查，会弹出很多警告提示等，可以在插件列表中关闭 clang 打头的几个即可，Help》About Plugins。也可以设置代码检查级别，Tools》Options 》C++ 》Code Model。
3. QSqlTableModel 的 rowCount 方法，默认最大返回 256，如果超过 256，可以将表格拉到底部，会自动加载剩余的，每次最大加载 256 条数据，如果需要打印或者导出数据，记得最好采用 sql 语句去查询，而不是使用 QSqlTableModel 的 rowCount 方法。不然永远最大只会导出 256 条数据。如果数据量很小，也可以采用如下方法：

```
// 主动加载所有数据，不然获取到的行数 <=256while(model->canFetchMore()) {    model->fetchMore();}
```

1. 如果需要指定无边框窗体，但是又需要保留操作系统的边框特性，可以自由拉伸边框，可以使用 setWindowFlags(Qt::CustomizeWindowHint);
2. 在某些 http post 数据的时候，如果采用的是&字符串连接的数据发送，中文解析乱码的话，需要将中文进行 URL 转码。

```
QString content = "测试中文";QString note = content.toUtf8().toPercentEncoding();
```

1. Qt 默认不支持大资源文件，比如添加了字体文件，需要 pro 文件开启。CONFIG += resources_big
2. Qt 中继承 QWidget 之后，样式表不起作用，解决办法有三个。强烈推荐方法一。

- 方法一：设置属性 this->setAttribute(Qt::WA_StyledBackground, true);
- 方法二：改成继承 QFrame，因为 QFrame 自带 paintEvent 函数已做了实现，在使用样式表时会进行解析和绘制。
- 方法三：重新实现 QWidget 的 paintEvent 函数时，使用 QStylePainter 绘制。

```
void Widget::paintEvent(QPaintEvent *){        QStyleOption option;    option.initFrom(this);    QPainter painter(this);style()->drawPrimitive(QStyle::PE_Widget, &option, &painter, this);}
```

1. 有时候在界面上加了弹簧，需要动态改变弹簧对应的拉伸策略，对应方法为 changeSize，很多人会选择使用 set 开头去找，找不到的。
2. 在使用 QFile 的过程中，不建议频繁的打开文件写入然后再关闭文件，比如间隔 5ms 输出日志，IO 性能瓶颈很大，这种情况建议先打开文件不要关闭，等待合适的时机比如析构函数中或者日期变了需要重新变换日志文件的时候关闭文件。不然短时间内大量的打开关闭文件会很卡，文件越大越卡。
3. 在很多网络应用程序，需要自定义心跳包来保持连接，不然断电或者非法关闭程序，对方识别不到，需要进行超时检测，但是有些程序没有提供心跳协议，此时需要启用系统层的保活程序，此方法适用于 TCP 连接。

```
int fd = tcpSocket->socketDescriptor();int keepAlive = 1; //开启 keepalive 属性,缺省值:0(关闭)int keepIdle = 5;        //如果在 5 秒内没有任何数据交互,则进行探测,缺省值:7200(s)int keepInterval = 2;    //探测时发探测包的时间间隔为 2 秒,缺省值:75(s)int keepCount = 2; //探测重试的次数,全部超时则认定连接失效,缺省值:9(次)setsockopt(fd, SOL_SOCKET,       SO_KEEPALIVE,       (void       *)&keepAlive, sizeof(keepAlive));setsockopt(fd,    SOL_TCP,    TCP_KEEPIDLE, (void *)&keepIdle, sizeof(keepIdle));setsockopt(fd, SOL_TCP, TCP_KEEPINTVL,              (void              *)&keepInterval, sizeof(keepInterval));setsockopt(fd,   SOL_TCP,   TCP_KEEPCNT, (void *)&keepCount, sizeof(keepCount));
```

1. 如果程序打包好以后弹出提示 This application failed to start because it could not find or load the Qt platform plugin 一般都是因为 platforms 插件目录未打包或者打包错了的原因导致的。
2. 非常不建议 tr 中包含中文，尽管现在的新版 Qt 支持中文到其他语言的翻译，但是很不规范，也不知道 TMD 是谁教的，tr 的本意是包含英文，然后翻译到其他语言比如中文，现在大量的初学者滥用 tr，如果没有翻译的需求，禁用 tr，tr 需要开销的，Qt 默认会认为他需要翻译，会额外进行特殊处理。
3. 很多人 Qt 和 Qt Creator 傻傻分不清楚，经常问 Qt 什么版本结果发一个 Qt Creator 的版本过来，Qt Creator 是使用 Qt 编写的集成开发环境 IDE，和宇宙第一的 Visual Studio 一样，他可以是 msvc 编译器的（WIN 对应的 Qt 集成安装环境中自带的 Qt Cerator 是 msvc 的），也可以是 mingw 编译的，还可以是 gcc 的。如果是自定义控件插件，需要集成到 Qt Creator 中，必须保证该插件的动态库文件（dll 或者 so 等文件）对应的编译器和 Qt 版本以及位数和 Qt Creator 的版本完全一致才行，否则基本不大可能集成

进去。特别注意的是 Qt 集成环境安装包中的 Qt 版本和 Qt Creator 版本未必完全一致，必须擦亮眼睛看清楚，有些是完全一致的。

4. 超过两处相同处理的代码，建议单独写成函数。代码尽量规范精简，比如 if(a == 123) 要写成 if (123 == a)，值在前面，再比如 if (ok == true) 要写成 if (ok)，if (ok == false) 要写成 if (!ok)等。
5. 很多人问 Qt 嵌入式平台用哪个好，这里统一回答（当前时间节点 2018 年）：imx6+335x 比较稳定，性能高就用 RK3288 RK3399，便宜的话就用全志 H3，玩一玩可以用树莓派香橙派。
6. 对于大段的注释代码，建议用 #if 0 #endif 将代码块包含起来，而不是将该段代码选中然后全部 // ，下次要打开这段代码的话，又需要重新选中一次取消，如果采用的是 #if 0 则只要把 0 改成 1 即可，效率大大提升。
7. Qt 打包发布，有很多办法，Qt5 以后提供了打包工具 windeployqt（linux 上为 linuxdeployqt，mac 上为 macdeployqt）可以很方便的将应用程序打包，使用下来发现也不是万能的，有时候会多打包一些没有依赖的文件，有时候又会忘记打包一些插件尤其是用了 qml 的情况下，而且不能识别第三方库，比如程序依赖 ffmpeg，则对应的库需要自行拷贝，终极大法就是将你的可执行文件复制到 Qt 安装目录下的 bin 目录，然后整个一起打包，挨个删除不大可能依赖的组件，直到删到正常运行为止。
8. Qt 中的动画，底层用的是 QElapsedTimer 定时器来完成处理，比如产生一些指定规则算法的数据，然后对属性进行处理。
9. 在绘制无背景颜色只有边框颜色的圆形时候，可以用绘制 360 度的圆弧替代，效果完全一致。

-
-
-
-

```
QRect rect(-radius, -radius, radius * 2, radius * 2);//以下两种方法二选一，其实绘制 360 度的圆弧=绘制无背景的圆形 painter->drawArc(rect, 0, 360 * 16);painter->drawEllipse(rect);
```

1. 不要把 d 指针看的很玄乎，其实就是在类的实现文件定义了一个私有类，用来存放局部变量，个人建议在做一些小项目时，没有太大必要引入这种机制，会降低代码可读性，增加复杂性，新手接受项目后会看的很懵逼。
2. 很多人在绘制的时候，设置画笔以为就只可以设置个单调的颜色，其实 QPen 还可以设置 brush，这样灵活性就提高不知道多少倍，比如设置 QPen 的 brush 以后，可以使用各种渐变，比如绘制渐变颜色的进度条和文字等，而不再是单调的一种颜色。
3. 很多控件都带有 viewport，比如 QTextEdit/QTableWidget/QScrollArea，有时候对这些控件直接处理的时候发现不起作用，需要对其 viewport() 设置才行，比如设置滚动条区域背景透明，需要使用 scrollArea->viewport()->setStyleSheet("background-color:transparent;"); 而不是 scrollArea->setStyleSheet("QScrollArea{background-color:transparent;}");
4. 有时候设置了鼠标跟踪 setMouseTracking 为真，如果该窗体上面还有其他控件，当鼠标移到其他控件上面的时候，父类的鼠标移动事件

MouseMove 识别不到了，此时需要用到 HoverMove 事件，需要先设置 setAttribute(Qt::WA_Hover, true);

5. Qt 封装的 QDateTime 日期时间类非常强大，可以字符串和日期时间相互转换，也可以毫秒数和日期时间相互转换，还可以 1970 经过的秒数和日期时间相互转换等。

```
QDateTime dateTime;QString dateTime_str = dateTime.currentDateTime().toString("yyyy-MM-dd hh:mm:ss");//从字符串转换为毫秒（需完整的年月日时分秒）datetime.fromString("2011-09-10 12:07:50:541", "yyyy-MM-dd hh:mm:ss:zzz").toMSecsSinceEpoch();//从字符串转换为秒（需完整的年月日时分秒）datetime.fromString("2011-09-10 12:07:50:541", "yyyy-MM-dd hh:mm:ss:zzz").toTime_t();//从毫秒转换到年月日时分秒 datetime.fromMSecsSinceEpoch(1315193829218).toString("yyyy-MM-dd hh:mm:ss:zzz");//从秒转换到年月日时分秒（若有 zzz，则为 000 ） datetime.fromTime_t(1315193829).toString("yyyy-MM-dd hh:mm:ss[:zzz]");
```

1. 在我们使用 QList、QStringList、QByteArray 等链表或者数组的过程中，如果只需要取值，而不是赋值，强烈建议使用 at() 取值而不是 [] 操作符，在官方书籍《C++ GUI Qt 4 编程（第二版）》的书中有特别的强调说明，此教材的原作者据说是 Qt 开发的核心人员编写的，所以还是比较权威，至于使用 at() 与使用 [] 操作符速度效率的比较，网上也有网友做过此类对比。原文在书的 212 页，这样描述的：Qt 对所有的容器和许多其他类都使用隐含共享，隐含共享是 Qt 对不希望修改的数据决不进行复制的保证，为了使隐含共享的作用发挥得最好，可以采用两个新的编程习惯。第一种习惯是对于一个（非常量的）向量或者列表进行只读存取时，使用 at() 函数而不用 [] 操作符，因为 Qt 的容器类不能辨别 [] 操作符是否将出现在一个赋值的左边还是右边，他假设最坏的情况出现并且强制执行深层赋值，而 at() 函数则不被允许出现在一个赋值的左边。
2. 如果是 dialog 窗体，需要在 exec 以后还能让其他代码继续执行，请在 dialog 窗体 exec 前增加一行代码，否则会阻塞窗体消息。

```
QDialog
dialog;dialog.setWindowModality(Qt::WindowModal);dialog.exec(
);
```

1. 安全的删除 Qt 的对象类，强烈建议使用 deleteLater 而不是 delete，因为 deleteLater 会选择在合适的时机进行释放，而 delete 会立即释放，很可能会出错崩溃。如果要批量删除对象集合，可以用 qDeleteAll，比如 qDeleteAll(btns);
2. 在 QTableView 控件中，如果需要自定义的列按钮、复选框、下拉框等其他模式显示，可以采用自定义委托 QItemDelegate 来实现，如果需要禁用某列，则在自定义委托的重载 createEditor 函数返回 0 即可。自定义委托对应的控件在进入编辑状态的时候出现，如果想一直出现，则需要重载 paint 函数用 drawPrimitive 或者 drawControl 来绘制。
3. 将 QApplication::style() 对应的 drawPrimitive、drawControl、drawItemText、drawItemPixmap 等几个方法用熟悉了，再结合 QStyleOption 属性，可以玩转各种自定义委托，还可以直接使用 paint 函数中的 painter 进行各种绘制，各种牛逼的表格、树状列表、下拉框等，绝对 炸天。QApplication::style()->drawControl 的第 4 个参数如果不设置，则绘制出来的控件不会应用样式表。
4. 心中有坐标，万物皆 painter，强烈建议在学习自定义控件绘制的时候，将 qpainter.h 头文件中的函数全部看一遍、试一遍、理解一遍，这里边包含了所有 Qt 内置的绘制的接口，对应的参数都试一遍，你会发现很多新大陆，会大大激发你的绘制的兴趣，犹如神笔马良一般，策马崩腾遨游代码绘制的世界。
5. 在使用 setItemWidget 或者 setCellWidget 的过程中，有时候会发现设置的控件没有居中显示而是默认的左对齐，而且不会自动拉伸填充，对于追求完美的程序员来说，这个可不大好看，有个终极通用办法就是，将这个控件放到一个 widget 的布局中，然后将 widget 添加到 item 中，这样就完美解决了，而且这样可以组合多个控件产生复杂的控件。

```
//实例化进度条控件 QProgressBar *progress = new QProgressBar;//增
加 widget+ 布 局 巧 妙 实 现 居 中 QWidget *widget = new
QWidget;QHBoxLayout *layout = new QHBoxLayout;layout-
>setSpacing(0);layout->setMargin(0);layout-
>addWidget(progress);widget->setLayout(layout);ui-
>tableWidget->setCellWidget(0, 0, widget);
```

1. 很多时候需要在已知背景色的情况下，能够清晰的绘制文字，这个时候需要计算对应的文字颜色。

```
//根据背景色自动计算合适的前景色 double gray = (0.299 * color.red() + 0.587 * color.green() + 0.114 * color.blue()) / 255;QColor textColor = gray > 0.5 ? Qt::black : Qt::white;
```

1. 对 QTableView 或者 QTableWidget 禁用列拖动。

```
#if (QT_VERSION <= QT_VERSION_CHECK(5,0,0))    ui->tableView->horizontalHeader()->setResizeMode(0, QHeaderView::Fixed);#else                          ui->tableView->horizontalHeader()->setSectionResizeMode(0, QHeaderView::Fixed);#endif
```

1. 从 Qt4 转到 Qt5，有些类的方法已经废弃或者过时了，如果想要在 Qt5 中启用 Qt4 的方法，比如 QHeadVew 的 setMovable，可以在你的 pro 或者 pri 文件中加上一行即可：DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0
2. Qt 中的 QColor 对颜色封装的很完美，支持各种转换，比如 rgb、hsb、cmy、hsl，对应的是 toRgb、toHsv、toCmyk、toHsl，还支持透明度设置，颜色值还能转成 16 进制格式显示。

```
QColor color(255, 0, 0, 100);qDebug() << color.name() << color.name(QColor::HexArgb);//输出 #ff0000 #64ff0000
```

1. QVariant 类型异常的强大，可以说是万能的类型，在进行配置文件的存储的时候，经常会用到 QVariant 的转换，QVariant 默认自带了 toString、toFloat 等各种转换，但是还是不够，比如有时候需要从 QVariant 转到 QColor，而却没有提供 toColor 的函数，这个时候就要用到万能办法。

```
if (variant.typeName() == "QColor") {       QColor color = variant.value<QColor>();                    QFont     font     =
```

```
variant.value<QFont>();                QString nodeValue =
color.name(QColor::HexArgb);}
```

1. Qt 中的 QString 和 const char * 之间转换，最好用 toStdString().c_str() 而不是 toLocal8Bit().constData()，比如在 setProperty 中如果用后者，字符串中文就会不正确，英文正常。
2. Qt 的信号槽机制非常牛逼，也是 Qt 的独特的核心功能之一，有时候我们在很多窗体中传递信号来实现更新或者处理，如果窗体层级比较多，比如窗体 A 的父类是窗体 B，窗体 B 的父类是窗体 C，窗体 C 有个子窗体 D，如果窗体 A 一个信号要传递给窗体 D，问题来了，必须先经过窗体 B 中转到窗体 C 再到窗体 D 才行，这样的话各种信号关联信号的 connect 会非常多而且管理起来比较乱，可以考虑增加一个全局的单例类 AppEvent，公共的信号放这里，然后窗体 A 对应信号绑定到 AppEvent，窗体 D 绑定 AppEvent 的信号到对应的槽函数即可，干净清爽整洁。
3. QTextEdit 右键菜单默认英文的，如果想要中文显示，加载 widgets.qm 文件即可，一个 Qt 程序中可以安装多个翻译文件，不冲突。
4. Qt 中有个全局的焦点切换信号 focusChanged，可以用它做自定义的输入法。Qt4 中默认会安装输入法上下文，比如在 main 函数打印 a.inputContext 会显示值，这个默认安装的输入法上下文，会拦截两个牛逼的信号 QEvent::RequestSoftwareInputPanel 和 QEvent::CloseSoftwareInputPanel，以至于就算你安装了全局的事件过滤器依然识别不到这两个信号，你只需要在 main 函数执行 a.setInputContext(0)即可，意思是安装输入法上下文为空。
5. 在 Qt5.10 以后，表格控件 QTableWidget 或者 QTableView 的默认最小列宽改成了 15，以前的版本是 0，所以在新版的 qt 中，如果设置表格的列宽过小，不会应用，取的是最小的列宽。所以如果要设置更小的列宽需要重新设置 ui->tableView->horizontalHeader()->setMinimumSectionSize(0);
6. Qt 源码中内置了一些未公开的不能直接使用的黑科技，都藏在对应模块的 private 中，比如 gui-private widgets-private 等，比如 zip 文件解压类 QZipReader、压缩类 QZipWriter 就在 gui-private 模块中，需要在 pro 中引入 QT += gui-private 才能使用。

-
-
-
-
-
-
-
-
-
-
-
-
-

```
#include "QtGui/private/qzipreader_p.h"#include "QtGui/private/qzipwriter_p.h"
QZipReader reader(dirPath);QString path("");//解压文件夹到当前目录 reader.extractAll(path);// 文件夹名称 QZipReader::FileInfo fileInfo = reader.entryInfoAt(0);// 解压文件 QFile file(filePath);file.open(QIODevice::WriteOnly);file.write(reader.fileData(QString::fromLocal8Bit("%1").arg(filePath)));file.close();reader.close();
QZipWriter *writer = new QZipWriter(dirPath);// 添加文件夹 writer->addDirectory(unCompress);// 添加文件 QFile file(filePath);file.open(QIODevice::ReadOnly);writer->addFile(data, file.readAll());file.close();writer->close();
```

1. 理论上串口和网络收发数据都是默认异步的，操作系统自动调度，完全不会卡住界面，网上那些说收发数据卡住界面主线程的都是扯几把蛋，真正的耗时是在运算以及运算后的处理，而不是收发数据，在一些小数据量运算处理的项目中，一般不建议动用线程去处理，线程需要调度开销的，不要什么东西都往线程里边扔，线程不是万能的。只有当真正需要将一些很耗时的操作比如编码解码等，才需要移到线程处理。
2. 在构造函数中获取控件的宽高很可能是不正确的，需要在控件首次显示以后再获取才是正确的，控件是在首次显示以后才会设置好正确的宽高值，记住是在首次显示以后，而不是构造函数或者程序启动好以后，如果程序启动好以后有些容器控件比如 QTabWidget 中的没有显示的页面的控件，你去获取宽高很可能也是不正确的，万无一失的办法就是首次显示以后去获取。
3. 数据库处理一般建议在主线程，如果非要在其他线程，务必记得打开数据库也要在那个线程，即在那个线程使用数据库就在那个线程打开，不能打开数据库在主线程，执行 sql 在子线程，很可能出问题。
4. 新版的 QTcpServer 类在 64 位版本的 Qt 下很可能不会进入 incomingConnection 函数，那是因为 Qt5 对应的 incomingConnection 函数参数变了，由之前的 int 改成了 qintptr，改成 qintptr 有个好处，在 32 位上自动是 quint32 而在 64 位上自动是 quint64，如果在 Qt5 中继续写的参数是 int 则在 32 位上没有问题在 64 位上才有问题，所以为了兼容 Qt4 和 Qt5，必须按照不一样的参数写。

```
#if (QT_VERSION > QT_VERSION_CHECK(5,0,0))          void
incomingConnection(qintptr handle);#else            void
incomingConnection(int handle);#endif
```

1. Qt 支持所有的界面控件比如 QPushButton、QLineEdit 自动关联 on_控件名_信号(参数) 信号槽，比如按钮的单击信号 on_pushButton_clicked()，然后直接实现槽函数即可。
2. QWebEngineView 控件由于使用了 opengl，在某些电脑上可能由于 opengl 的驱动过低会导致花屏或者各种奇奇怪怪的问题，比如 showfullscreen 的情况下鼠标右键失效，需要在 main 函数启用软件 opengl 渲染。

```
#if (QT_VERSION > QT_VERSION_CHECK(5,4,0))    //下面两种方法都可以 ,Qt 默认采用的是 AA_UseDesktopOpenGL
QCoreApplication::setAttribute(Qt::AA_UseOpenGLES);
//QCoreApplication::setAttribute(Qt::AA_UseSoftwareOpenGL);#endif    QApplication a(argc, argv);
```

另外一个方法解决 全屏+QWebEngineView 控件一起会产生右键菜单无法弹出的 bug,需要上移一个像素

```
QRect rect = qApp->desktop()->geometry();rect.setY(-1);rect.setHeight(rect.height());this->setGeometry(rect);
```

1. QStyle 内置了很多方法用处很大，比如精确获取滑动条鼠标按下处的值。

```
QStyle::sliderValueFromPosition(minimum(), maximum(), event->x(), width());
```

1. 用 QFile 读写文件的时候，推荐用 QTextStream 文件流的方式来读写文件，速度快很多，基本上会有 30%的提升，文件越大性能区别越大。

```
// 从文件加载英文属性与中文属性对照表 QFile file(":/propertyname.txt");if (file.open(QFile::ReadOnly)) { //QTextStream 方法读取速度至少快 30%#if 0 while(!file.atEnd()) { QString line = file.readLine(); appendName(line); }#else QTextStream in(&file); while (!in.atEnd()) { QString line = in.readLine(); appendName(line); }#endif file.close();}
```

1. 用 QFile.readAll()读取 QSS 文件默认是 ANSI 格式，不支持 UTF8，如果在 QtCreator 中打开 qss 文件来编辑保存，这样很可能导致 qss 加载以后没有效果。

```
void frmMain::initStyle(){    //加载样式表    QString qss;    //
QFile    file(":/qss/psblack.css");                    //QFile
file(":/qss/flatwhite.css");                             QFile
file(":/qss/lightblue.css");                               if
(file.open(QFile::ReadOnly)) {#if 1        //用QTextStream读取
样式文件不用区分文件编码 带bom也行          QStringList list;
QTextStream  in(&file);                //in.setCodec("utf-8");
while (!in.atEnd()) {             QString line;             in
>> line;            list << line;        }
        qss = list.join("\n");#else        //用readAll读取默认
支持的是ANSI格式,如果不小心用creator打开编辑过了很可能打不开
qss = QLatin1String(file.readAll());#endif            QString
paletteColor  =  qss.mid(20,  7);                       qApp-
>setPalette(QPalette(QColor(paletteColor)));            qApp-
>setStyleSheet(qss);        file.close();    }}
```

1. QString内置了很多转换函数，比如可以调用toDouble转为double数据，但是当你转完并打印的时候你会发现精确少了，只剩下三位了，其实原始数据还是完整的精确度的，只是打印的时候优化成了三位，如果要保证完整的精确度，可以调用 qSetRealNumberPrecision 函数设置精确度位数即可。

```
QString s1, s2;s1 = "666.5567124";s2.setNum(888.5632123, 'f',
7);qDebug() << qSetRealNumberPrecision(10) << s1.toDouble()
<< s2.toDouble();
```

1. 用QScriptValueIterator解析数据的时候，会发现总是会多一个节点内容并且内容为空，如果需要跳过则增加一行代码。

```
while (it.hasNext()) {    it.next();        if (it.flags() &
QScriptValue::SkipInEnumeration)                  continue;
qDebug() << it.name();}
```

1. setPixmap 是最糟糕的贴图方式，一般只用来简单的不是很频繁的贴图，频繁的建议 painter 绘制，默认双缓冲，在高级点用 opengl 绘制，利用 GPU。
2. 如果需要在尺寸改变的时候不重绘窗体，则设置属性即可 this->setAttribute(Qt::WA_StaticContents, true); 这样可以避免可以避免对已经显示区域的重新绘制。
3. 默认程序中获取焦点以后会有虚边框，如果看着觉得碍眼不舒服可以去掉，设置样式即可：setStyleSheet("*{outline:0px;}");
4. Qt 表格控件一些常用的设置封装，QTableWidget 继承自 QTableView，所以下面这个函数支持传入 QTableWidget。

```
void QUIHelper::initTableView(QTableView *tableView, int rowHeight, bool headVisible, bool edit){    //奇数偶数行颜色交替 tableView->setAlternatingRowColors(false);    //垂直表头是否可见 tableView->verticalHeader()->setVisible(headVisible);    //选中一行表头是否加粗 tableView->horizontalHeader()->setHighlightSections(false);    //最后一行拉伸填充 tableView->horizontalHeader()->setStretchLastSection(true);    //行标题最小宽度尺寸 tableView->horizontalHeader()->setMinimumSectionSize(0);    //行标题最大高度 tableView->horizontalHeader()->setMaximumHeight(rowHeight);    //默认行高 tableView->verticalHeader()->setDefaultSectionSize(rowHeight);    //选中时一行整体选中 tableView->setSelectionBehavior(QAbstractItemView::SelectRows);    //只允许选择单个 tableView->setSelectionMode(QAbstractItemView::SingleSelection);    //表头不可单击#if (QT_VERSION > QT_VERSION_CHECK(5,0,0)) tableView->horizontalHeader()->setSectionsClickable(false);#else tableView->horizontalHeader()->setClickable(false);#endif    //鼠标按下即进入编辑模式 if (edit) { tableView->setEditTriggers(QAbstractItemView::CurrentChanged | QAbstractItemView::DoubleClicked); } else { tableView->setEditTriggers(QAbstractItemView::NoEditTriggers); }}
```

1. 在一些大的项目中，可能嵌套了很多子项目，有时候会遇到子项目依赖其他子项目的时候，比如一部分子项目用来生成动态库，一部分子项目依赖这个动态库进行编译，此时就需要子项目按照顺序编译。

```
TEMPLATE = subdirs#设置ordered参数以后会依次编译 demo designer examplesCONFIG += orderedSUBDIRS += demoSUBDIRS += designerSUBDIRS += examples
```

1. MSVC 编译器的选择说明

- 如果是 32 位的 Qt 则编译器选择 x86 开头的
- 如果是 64 位的 Qt 则编译器选择 amd64 开头的

- 具体是看安装的 Qt 构建套件版本以及目标运行平台的系统位数和架构
- 一般现在的电脑默认以 64 位的居多，选择 amd64 即可
- 如果用户需要兼容 32 位的系统则建议选择 32 位的 Qt，这样即可在 32 位也可以在 64 位系统运行
- 诸葛大佬补充：x86/x64 都是编译环境和运行环境相同，没有或。带下划线的就是交叉编译，前面是编译环境，后面是运行环境。
  名称说明

x86	32/64 位系统上编译在 32/64 位系统上运行
x86_amd64	32/64 位系统上编译在 64 位系统上运行
x86_arm	32/64 位系统上编译在 arm 系统上运行
amd64	64 位系统上编译在 64 位系统上运行
amd64_x86	64 位系统上编译在 32/64 位系统上运行
amd64_arm	64 位系统上编译在 arm 系统上运行

1. 很多时候用 QDialog 的时候会发现阻塞了消息，而有的时候我们希望是后台的一些消息继续运行不要终止，此时需要做个设置。

```
QDialog dialog;dialog.setWindowModality(Qt::WindowModal);
```

1. 很多初学者甚至几年工作经验的人，对多线程有很深的误解和滥用，尤其是在串口和网络通信这块，什么都往多线程里面丢，一旦遇到界面卡，就把数据收发啥的都搞到多线程里面去，殊不知绝大部分时候那根本没啥用，因为没找到出问题的根源。
- 如果你没有使用 wait***函数的话，大部分的界面卡都出在数据处理和展示中，比如传过来的是一张图片的数据，你需要将这些数据转成图片，这个肯定是耗时的；
- 还有就是就收到的数据曲线绘制出来，如果过于频繁或者间隔过短，肯定会给 UI 造成很大的压力的，最好的办法是解决如何不要频繁绘制 UI 比如合并数据一起绘制等；
- 如果是因为绘制 UI 造成的卡，那多线程也是没啥用的，因为 UI 只能在主线程；
- 串口和网络的数据收发默认都是异步的，由操作系统调度的，如果数据处理复杂而且数据量大，你要做的是将数据处理放到多线程中；
- 如果没有严格的数据同步需求，根本不需要调用 wait***之类的函数来立即发送和接收数据，实际需求中大部分的应用场景其实异步收发数据就足够了；
- 有严格数据同步需求的场景还是放到多线程会好一些，不然你 wait***就卡在那边了；
- 多线程是需要占用系统资源的，理论上来说，如果线程数量超过了 CPU 的核心数量，其实多线程调度可能花费的时间更多，各位在使用过程中要权衡利弊；

1. 在嵌入式 linux 上，如果设置了无边框窗体，而该窗体中又有文本框之类的，发现没法产生焦点进行输入，此时需要主动激活窗体才行。

```
//这种方式设置的无边框窗体在嵌入式设备上无法产生焦点
setWindowFlags(Qt::WindowStaysOnTopHint | Qt::FramelessWindowHint | Qt::X11BypassWindowManagerHint);
//需要在 show 以后主动激活窗体 w->show();w->activateWindow();
```

1. QString 的 replace 函数会改变原字符串，切记，他在返回替换后的新字符串的同时也会改变原字符串，我的乖乖！
2. QGraphicsEffect 类的相关效果很炫，可以实现很多效果比如透明、渐变、阴影等，但是该类很耗 CPU，如果不是特别需要一般不建议用，就算用也是要用在该部件后期不会发生频繁绘制的场景，不然会让你哭晕在厕所。
3. 在不同的平台上文件路径的斜杠也是不一样的，比如 linux 系统一般都是 / 斜杠，而在 windows 上都是 \ 两个反斜杠，Qt 本身程序内部无论在 win 还是 linux 都支持 / 斜杠的路径，但是一些第三方库的话可能需要转换成对应系统的路径，这就需要用到斜杠转换，Qt 当然内置类方法。

```
QString path = "C:/temp/test.txt";path = QDir::toNativeSeparators(path);//输出 C:\\temp\\test.txt
QString path = "C:\\temp\\test.txt";path = QDir::toNativeSeparators(path);//输出 C:/temp/test.txt
```

1. 巧用 QMetaObject::invokeMethod 方法可以实现很多效果，包括同步和异步执行，比如有个应用场景是在回调中，需要异步调用一个 public 函数，如果直接调用的话会发现不成功，此时需要使用 QMetaObject::invokeMethod(obj, "fun", Qt::QueuedConnection); 这种方式来就可以。invokeMethod 函数有很多重载参数，可以传入返回值和执行方法的参数等。
2. Qt5 中的信号是 public 的，可以在需要的地方直接 emit 即可，而在 Qt4 中信号是 protected 的，不能直接使用，需要定义一个 public 函数来 emit。
3. Qt5.15 版本开始官方不再提供安装包，只提供源码，可以自行编译或者在线安装，估计每次编译各种版本太麻烦，更多的是为了统计收集用户使用信息比如通过在线安装，后期可能会逐步加大商业化力度。

4. 有时候我们需要判断当前 Qt 版本有没有某个模块可以使用 qtHaveModule（Qt5 新引入的判断）来判断，如果要判断自己的项目中有没有 QT += 的方式添加的模块，可以用 contains 来判断。

```
qtHaveModule(webenginewidgets) {message("当前 Qt 库有找到 webenginewidgets 模块")}
!qtHaveModule(webkit) {message("当前 Qt 库没有找到 webkit 模块")}
contains(QT, network) {message("当前项目已经引入 network 模块")}
!contains(QT, widgets) {message("当前项目没有引入 widgets 模块")}c++11 新引入了原始字符串格式，用户避免在字符串中加入转义字符\，可以用于表示 json 字符串等场景。QString s1 = R"(test\001.jpg)";s1.replace("\\", "#");qDebug()<< s1;// 结果 test#001.jpg
```

1. 安卓上打印信息建议使用 qInfo() 而不是 qDebug()，qInfo()才有效果。
2. Qt 的默认定时器精度不够高（比如应用场景是 1 分钟保存一条记录或者文件，当你用默认的定时器的时候你会发现有些时候是 60 秒而有些是 59 秒随机的，如果客户有要求这就需要设置精度了。当然我们所做的绝大部分项目也不需要精度非常高的定时器，毕竟精度越高，占用的系统资源可能越大），如果需要设置更高的精度可以设置 setTimerType(Qt::PreciseTimer)。Qt 有两种定时器处理，一种是 QTimer 类，还有一种是 QObject 类就内置的 timeevent 事件，如果是 QObject 类的定时器要设置的话调用 startTimer(interval, Qt::PreciseTimer);

- Qt::PreciseTimer 精确的定时器，尽量保持毫秒精度。
- Qt::CoarseTimer 粗略的定时器，尽量保持精度在所需的时间间隔 5%范围内。
- Qt::VeryCoarseTimer 很粗略的定时器，只保留完整的第二精度。

- 精度再高，也依赖对应的操作系统中断，假设中断需要 5ms，则定时器精度不可能高于 5 毫秒。

1. QGraphicsEffect 相关类很耗 CPU，甚至在绘制的时候和某些地方有冲突干扰，基本上不建议使用，情非得已只建议少量使用和非频繁触发绘制的地方使用。
2. 用 QSettings 设置注册表，如果不是管理员身份运行会打印 QSettings: failed to set subkey "xxx" (拒绝访问。)，你需要手动鼠标右键管理员身份运行就可以。
3. QLineEdit 除了单纯的文本框以外，还可以做很多特殊的处理用途。

- 限制输入只能输入 IP 地址。
- 限制输入范围，强烈推荐使用 QRegExpValidator 正则表达式来处理。

```
//正在表达式限制输入 QString str = "\\b(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\b";ui->lineEdit->setValidator(new QRegExpValidator(QRegExp(str)));// 用 于 占 位 ui->lineEdit->setInputMask("000.000.000.000");
#if 0//下面代码设置浮点数范围限制失败 ui->lineEdit->setValidator(new
```

```
QDoubleValidator(20, 50, 1));#else//下面代码设置浮点数范围限制成功
QDoubleValidator *validator = new QDoubleValidator(20, 50, 1);validator->setNotation(QDoubleValidator::StandardNotation);ui->lineEdit->setValidator(validator);#endif//下面代码设置整数范围限制成功ui->lineEdit->setValidator(new QIntValidator(10, 120));
//其实上面的代码缺陷很多，只能限制只输入小数，无法设定数值范围，很操蛋//需要来个万能的牛逼的 QRegExpValidator
//限制浮点数输入范围为[-180,180]QRegExp regexp("^-?(180|1?[0-7]?\\d(\\.\\d+)?)$");//限制浮点数输入范围为[-90,90]并限定为小数位后4位 QRegExp regexp("^-?(90|[1-8]?\\d(\\.\\d{1,4})?)$");QRegExpValidator *validator = new QRegExpValidator(regexp, this);ui->lineEdit->setValidator(validator);
```

1. 在继承自 QAbstractItemView 的控件中，比如 QTableView、QTableWidget，如果文本超过队列 item 的宽度，则会自动省略号显示，想要快速显示完整的文本，可以在该列和下一列分割线中间双击即可，会自动自适应显示最大宽度，如果是 Qt5.14 或者更高版本，你会发现显示省略号的计算规则变了，如果是 rtsp、http 之类的开头的英文字符串，同样的列宽下，会提前就显示省略号，比如字符串 rtmp://58.200.131.2:1935/livetv/cctv1，会显示成 rtmp://...，而在旧版本的 Qt 中会显示成 rtmp://58.200.131...，很多时候我们并不想看到烦人的省略号，可以设置取消。

```
//取消自动换行 tableView->setWordWrap(false);//超出文本不显示省略号
tableView->setTextElideMode(Qt::ElideNone);
```

四、其他经验

1. Qt 界的中文乱码问题，版本众多导致的如何选择安装包问题，如何打包发布程序的问题，堪称 Qt 界的三座大山！
2. 在 Qt 的学习过程中，学会查看对应类的头文件是一个好习惯，如果在该类的头文件没有找到对应的函数，可以去他的父类中找找，实在不行还有爷爷类，肯定能找到的。通过头文件你会发现很多函数接口其实 Qt 已经帮我们封装好了，有空还可以阅读下他的实现代码。
3. Qt 安装目录下的 Examples 目录下的例子，看完学完，月薪 20K 起步；Qt 常用类的头文件的函数看完学完使用一遍并加以融会贯通，月薪 30K 起步。
4. Qt 在开发阶段不支持中文目录，切记，这是无数人可能犯的错误，在安装 Qt 集成开发环境以及编译器的时候，务必记得目录必须英文，否则很可能不正常，建议尽量用默认的安装位置。
5. 如果出现崩溃和段错误，80%都是因为要么越界，要么未初始化，死扣这两点，80%的问题解决了。

6. Qt 一共有几百个版本，关于如何选择 Qt 版本的问题，我一般保留四个版本，为了兼容 Qt4 用 4.8.7，最后的支持 XP 的版本 5.7.0，最新的长期支持版本比如 5.12，最高的新版本比如 5.14.2。强烈不建议使用 4.7 以前和 5.0 到 5.3 之间的版本，太多 bug 和坑，稳定性和兼容性相比于之后的版本相当差，能换就换，不能换睡服领导也要换。
7. Qt 和 msvc 编译器常见搭配是 Qt5.7+VS2013、Qt5.9+VS2015、Qt5.12+VS2017，按照这些搭配来，基本上常用的模块都会有，比如 webengine 模块，如果选用的 Qt5.12+msvc2015，则很可能官方没有编译这个模块，只是编译了 Qt5.12+msvc2017 的。
8. 终极秘籍：如果遇到问题搜索 Qt 方面找不到答案，试着将关键字用 JAVA C# android 打头，你会发现别有一番天地，其他人很可能做过！
9. 新版本 Qt 安装包安装的时候需要填写注册信息，如果不想填写，先禁用网卡，在运行安装包，可以直接跳过这一步进行安装。
10. 最后一条：珍爱生命，远离编程。祝大家头发浓密，睡眠良好，情绪稳定，财富自由！

五、推荐的 Qt 论坛+个人博客+网站+群

名称	网址
QtWidget 开源 demo 集合	https://gitee.com/feiyangqingyun/QWidgetDemo
QtQuick/Qml 开源 demo 集合	https://gitee.com/jaredtao/TaoQuick
qtcn	http://www.qtcn.org
豆子的空间	https://www.devbean.net
yafeilinux	http://www.qter.org
一去二三里	http://blog.csdn.net/liang19890820
乌托邦 2 号	http://blog.csdn.net/taiyang1987912
foruok	http://blog.csdn.net/foruok
jason	http://blog.csdn.net/wsj18808050
朝十晚八	http://www.cnblogs.com/swarmbees
BIG_C_GOD	http://blog.csdn.net/big_c_god
公孙二狗	https://qtdebug.com/qtbook
雨田哥	https://blog.csdn.net/ly305750665
郑天佐	https://blog.csdn.net/zhengtianzuo06

名称	网址
寒山-居士	https://blog.csdn.net/esonpo
feiyangqingyun	https://blog.csdn.net/feiyangqingyun
前行中小猪	http://blog.csdn.net/goforwardtostep
涛哥的知乎专栏	https://zhuanlan.zhihu.com/TaoQt
Qt 君	https://blog.csdn.net/nicai_xiaoqinxi
Qt 老外视频教程	http://space.bilibili.com/2592237/#!/index
Qt 维基补充文档	https://wiki.qt.io/Main
Qt 源码查看网站	https://code.woboq.org/qt5
Qt 官方下载地址	https://download.qt.io
Qt 官方下载新地址	https://download.qt.io/new_archive/qt/
Qt 国内镜像下载地址	https://mirrors.cloud.tencent.com/qt
Qt 安装包下载地址	http://qthub.com/download/ (超过 1000 多个，由 Qt 君整理)
精美图表控件 QWT	http://qwt.sourceforge.net/
精美图表控件 QCustomPlot	https://www.qcustomplot.com/
免费图标下载	http://www.easyicon.net/
图形字体下载	https://www.iconfont.cn/
漂亮界面网站	https://www.ui.cn/

六、其他

1. C++入门书籍推荐《C++ primer plus》，进阶书籍推荐《C++ primer》。
2. Qt 入门书籍推荐霍亚飞的《Qt Creator 快速入门》，Qt 进阶书籍推荐官方的《C++ GUI Qt4 编程》，qml 书籍推荐《Qt5 编程入门》。
3. 强烈推荐程序员自我修养和规划系列书《大话程序员》《程序员的成长课》《解忧程序员》，受益匪浅，受益终生！

-END-