

Qt 开发笔记

23090032047 计算机类 1 班 于景一

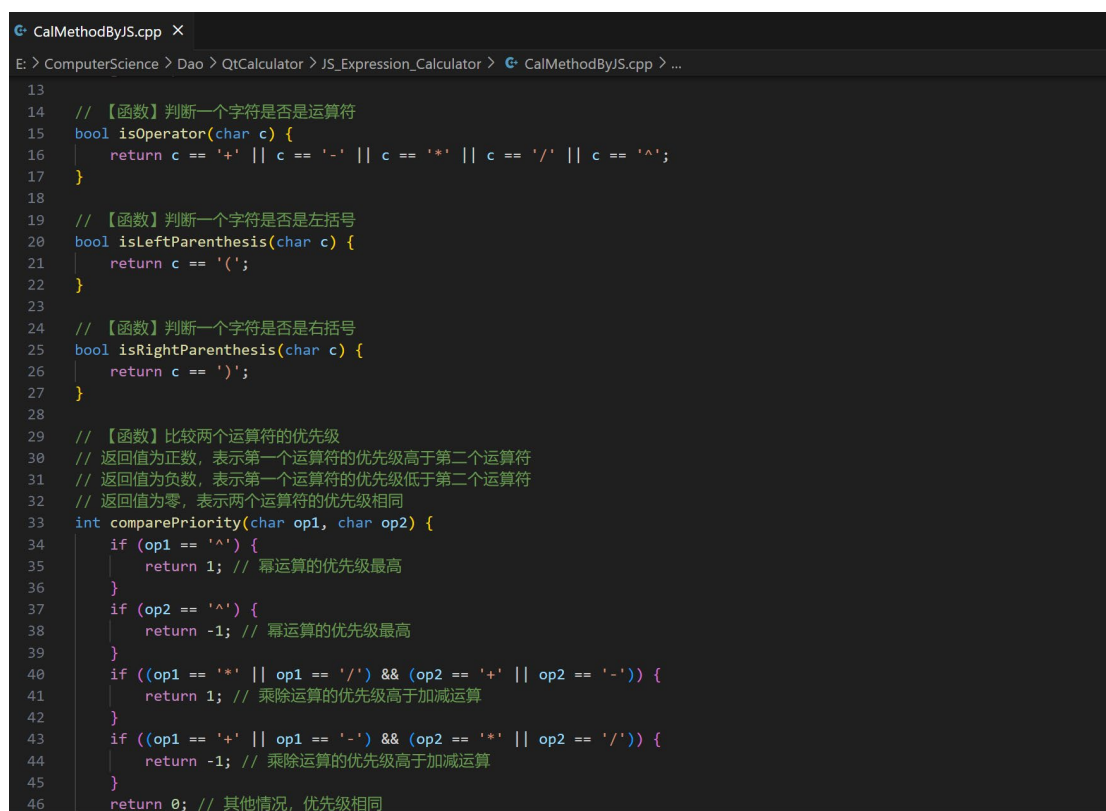
1. 开发思路

考虑其功能的实现方法；考虑如何正确设置开发环境；考虑是否要实现跨平台的特性，如何实现，如何封装。

1.1. 实现方法

① 计算模块：使用基于逆波兰表达式的双栈法进行表达式计算。

见：源码/CalMethodByJS.cpp



```
13
14 // 【函数】判断一个字符是否是运算符
15 bool isOperator(char c) {
16     return c == '+' || c == '-' || c == '*' || c == '/' || c == '^';
17 }
18
19 // 【函数】判断一个字符是否是左括号
20 bool isLeftParenthesis(char c) {
21     return c == '(';
22 }
23
24 // 【函数】判断一个字符是否是右括号
25 bool isRightParenthesis(char c) {
26     return c == ')';
27 }
28
29 // 【函数】比较两个运算符的优先级
30 // 返回值为正数，表示第一个运算符的优先级高于第二个运算符
31 // 返回值为负数，表示第一个运算符的优先级低于第二个运算符
32 // 返回值为零，表示两个运算符的优先级相同
33 int comparePriority(char op1, char op2) {
34     if (op1 == '^') {
35         return 1; // 幂运算的优先级最高
36     }
37     if (op2 == '^') {
38         return -1; // 幂运算的优先级最高
39     }
40     if ((op1 == '*' || op1 == '/') && (op2 == '+' || op2 == '-')) {
41         return 1; // 乘除运算的优先级高于加减运算
42     }
43     if ((op1 == '+' || op1 == '-') && (op2 == '*' || op2 == '/')) {
44         return -1; // 乘除运算的优先级高于加减运算
45     }
46     return 0; // 其他情况，优先级相同
```

图 1.1-1 表达式计算程序的一种实现

② Signals and Slots:

对此二者的理解如下：

Signal 是当某个对象的状态发生变化时发出的一种通知。Qt 的控件有许多预定义的信号，但我们也可以通过继承控件来添加自己的信号。

Slot 是响应某个信号而调用的一种函数。Qt 的控件有许多预定义的插槽，但我们也可以通过继承控件来添加自己的插槽。

信号和插槽是松耦合的：发出信号的对象不知道也不关心有哪些插槽接收了信号。Qt 的信号和插槽机制保证了如果我们将一个信号和一个插槽连接起来，那么在适当的时候，插槽会被信号的参数调用。

如何链接此两者：

连接信号和插槽：我们可以使用 QObject::connect 函数来连接信号和插槽。我们可以使用基于函数指针的语法，也可以使用基于字符串的语法。基于函数指针的语法更加简洁和类型安全，但是基于字符串的语法更加灵活和动态。

参看：源码/JSCalculator.cpp: Line 170..199

```
// 连接被点击的按钮
// 注意：QObject::connect 应该被传递三个参数，当context被删除时，表达式不必被执行，防止内存泄露。
connect(ui->num0, &QPushButton::clicked, this, [=]{expr.insert('0'); RfrInput(); UpdStack();});
connect(ui->num1, &QPushButton::clicked, this, [=]{expr.insert('1'); RfrInput(); UpdStack();});
connect(ui->num2, &QPushButton::clicked, this, [=]{expr.insert('2'); RfrInput(); UpdStack();});
connect(ui->num3, &QPushButton::clicked, this, [=]{expr.insert('3'); RfrInput(); UpdStack();});
connect(ui->num4, &QPushButton::clicked, this, [=]{expr.insert('4'); RfrInput(); UpdStack();});
connect(ui->num5, &QPushButton::clicked, this, [=]{expr.insert('5'); RfrInput(); UpdStack();});
connect(ui->num6, &QPushButton::clicked, this, [=]{expr.insert('6'); RfrInput(); UpdStack();});
connect(ui->num7, &QPushButton::clicked, this, [=]{expr.insert('7'); RfrInput(); UpdStack();});
connect(ui->num8, &QPushButton::clicked, this, [=]{expr.insert('8'); RfrInput(); UpdStack();});
connect(ui->num9, &QPushButton::clicked, this, [=]{expr.insert('9'); RfrInput(); UpdStack();});
connect(ui->opDot, &QPushButton::clicked, this, [=]{expr.insert('.'); RfrInput(); UpdStack();});
connect(ui->opAdd, &QPushButton::clicked, this, [=]{expr.insert('+'); RfrInput(); UpdStack();});
connect(ui->opSub, &QPushButton::clicked, this, [=]{expr.insert('-'); RfrInput(); UpdStack();});
connect(ui->opMulti, &QPushButton::clicked, this, [=]{expr.insert('*'); RfrInput(); UpdStack();});
connect(ui->opDiv, &QPushButton::clicked, this, [=]{expr.insert('/'); RfrInput(); UpdStack();});
connect(ui->opSqr, &QPushButton::clicked, this, [=]{expr.insert('^'); RfrInput(); UpdStack(); expr.insert('2'); RfrInput();});
connect(ui->opPow, &QPushButton::clicked, this, [=]{expr.insert('^'); RfrInput(); UpdStack();});
connect(ui->opMod, &QPushButton::clicked, this, [=]{expr.insert('%'); RfrInput(); UpdStack();});
connect(ui->opBrckL, &QPushButton::clicked, this, [=]{expr.insert('('); RfrInput(); UpdStack();});
connect(ui->opBrckR, &QPushButton::clicked, this, [=]{expr.insert(')'); RfrInput(); UpdStack();});
connect(ui->opEqual, &QPushButton::clicked, this, [=]{expr.insert('='); UpdStack(); RfrInput(); RfrResult(); UpdHistory();});
connect(ui->opClear, &QPushButton::clicked, this, [=]{expr.clear(); RfrInput(); UpdStack(true);});
connect(ui->opCkSpace, &QPushButton::clicked, this, [=]{expr.backSpace(); RfrInput(); UpdStack(true);});

// 注：非标准化的签名会导致不必要的内存分配，影响性能。在这里进行修改。
connect(this, SIGNAL(addHistoryTerm(QString,QString,QString)), tabPage, SLOT(Controller_AddHistoryTerm(QString,QString,QString)));
connect(this, SIGNAL(addHistoryTerm(const QString &, const QString &, const QString &)), tabPage, SLOT(Controller_AddHistoryTerm(const QString &, const QString &, const QString &)));
connect(this, SIGNAL(updateStackPage(QString,QString)), tabPage, SLOT(Controller_UpdateStackPage(QString,QString)));
connect(this, SIGNAL(updateStackPage(const QString &, const QString &)), tabPage, SLOT(Controller_UpdateStackPage(const QString &, const QString &)));
```

图 1.1-2 在计算器程序中的应用

1.2. 设置开发环境

① 下载并安装 Qt

在 https://download.qt.io/official_releases/online_installers/ 获取到正确 Online Installer，在无商用 License 的时候会默认下载开源的版本。

对于 Windows 选择 qt-unified-windows-x64-online.exe

对于 Linux 选择 qt-unified-linux-x64-online.run

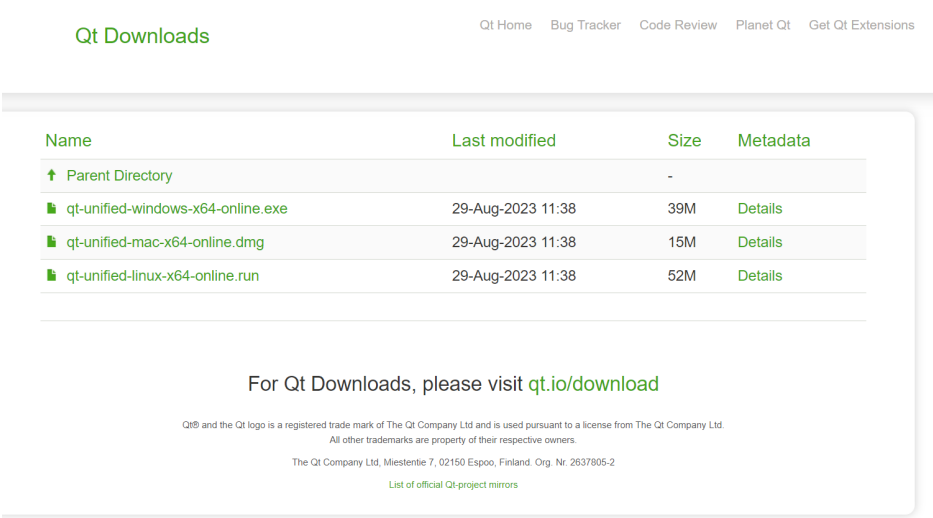


图 1.2-1 下载 Qt

② 使用 WSL2 建立 Windows 下的 Ubuntu 子系统

考虑到 Windows11 带有子系统的功能，可以在 Windows 运行之上实现 Linux 子系统的运行，做到无缝衔接，便于开发，且内存占用率低。在这里提供简单的实现方法：

```
PS C:\Users\JStar> wsl -l -v
  NAME      STATE      VERSION
* Ubuntu    Running    2
PS C:\Users\JStar> |
```

图 1.2-2 正确配置后的 WSL 列表

简单给出一个设置方法：

```
wsl --update
wsl --set-default-version 2
wsl --export Ubuntu-22.04 ****\ubuntu22.04.tar
wsl --unregister Ubuntu-22.04
wsl --import Ubuntu-22.04 ****\UbuntuWSL ****\ubuntu22.04.tar --version 2
ubuntu2204.exe config --default-user USER_YOU_SET_BEFORE
sudo passwd root
```

对于一些基本的带 GUI 窗口，WSL2 默认可以实现（基于 Wayland）。

配置系统基本的程序见微软的文档：

<https://learn.microsoft.com/zh-cn/windows/wsl/tutorials/gui-apps>

下面给出正确运行 WSL2 的证明。

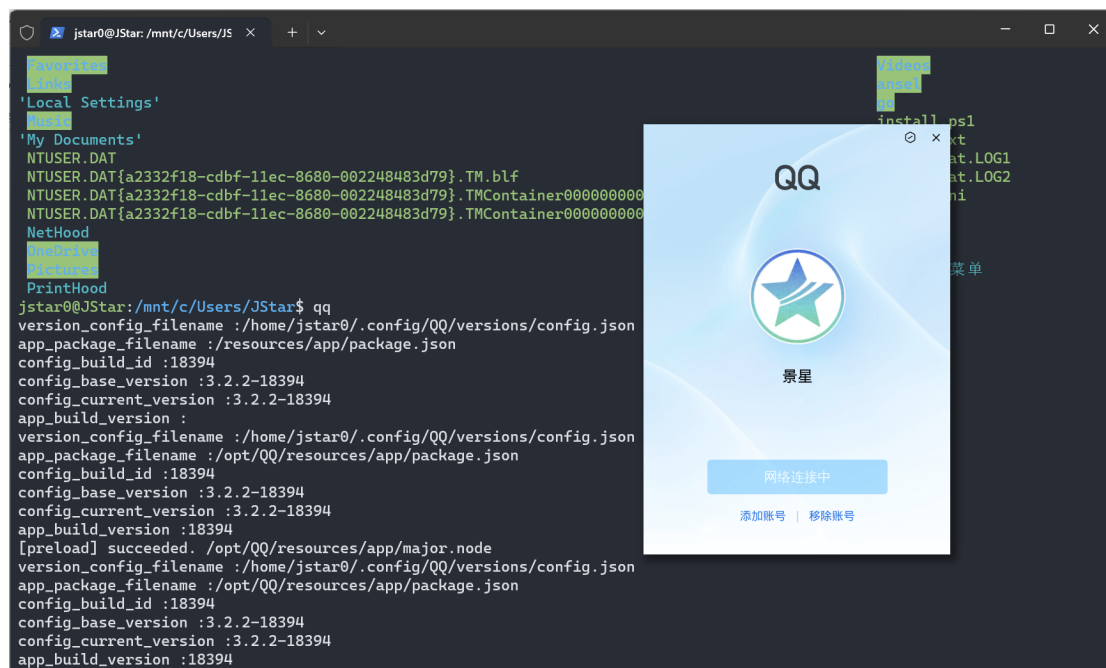


图 1.2-3 在 WSL2 的 Ubuntu 22.04 LTS 运行 Linux QQ

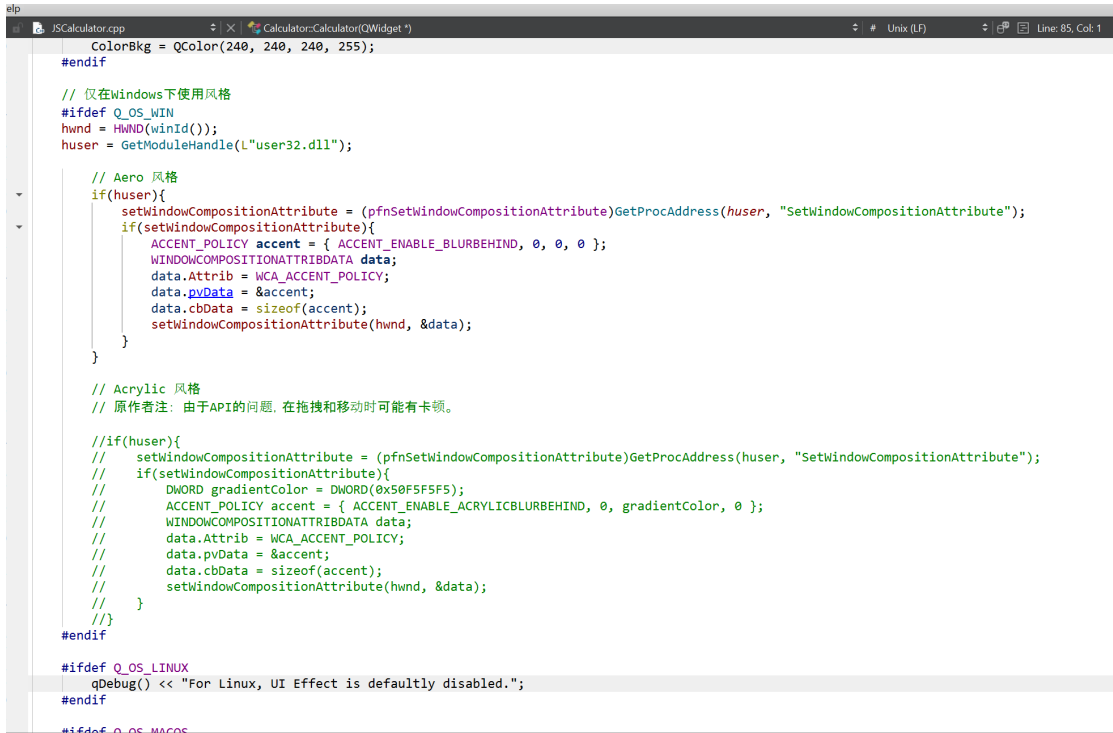
1.3. 跨平台的实现

Qt 具有显而易见的跨平台性，然而在程序也应注意这一点。

如果我要仅在 Windows 下调用 WinAPI (引用 windows.h), 而在其他平台用其他的方法来替换, 则参考:

```
#ifdef CONDITION_1
    #include <Windows.h>
#else
    qDebug() << "For other platforms, UI Effect is defaultly disabled.";
#endif
```

在本项目中, 请看: 源代码/JSCalculator.cpp 及其他。



```
ColorBkg = QColor(240, 240, 240, 255);
#endif

// 仅在Windows下使用风格
#ifdef Q_OS_WIN
hwnd = HWND(winId());
huser = GetModuleHandle(L"user32.dll");

// Aero 风格
if(huser){
    setWindowCompositionAttribute = (pfnSetWindowCompositionAttribute)GetProcAddress(huser, "SetWindowCompositionAttribute");
    if(setWindowCompositionAttribute){
        ACCENT_POLICY accent = { ACCENT_ENABLE_BLURBEHIND, 0, 0, 0 };
        WINDOWCOMPOSITIONATTRIBDATA data;
        data.Attrib = WCA_ACCENT_POLICY;
        data.pvData = &accent;
        data.cbData = sizeof(accent);
        setWindowCompositionAttribute(hwnd, &data);
    }
}

// Acrylic 风格
// 原作者注: 由于API的问题, 在拖拽和移动时可能有卡顿。

//if(huser){
//    setWindowCompositionAttribute = (pfnSetWindowCompositionAttribute)GetProcAddress(huser, "SetWindowCompositionAttribute");
//    if(setWindowCompositionAttribute){
//        DWORD gradientColor = DWORD(0x50F5F5F5);
//        ACCENT_POLICY accent = { ACCENT_ENABLE_ACRYLICBLURBEHIND, 0, gradientColor, 0 };
//        WINDOWCOMPOSITIONATTRIBDATA data;
//        data.Attrib = WCA_ACCENT_POLICY;
//        data.pvData = &accent;
//        data.cbData = sizeof(accent);
//        setWindowCompositionAttribute(hwnd, &data);
//    }
//}
#endif

#ifdef Q_OS_LINUX
qDebug() << "For Linux, UI Effect is defaultly disabled.";
#endif

#ifdef Q_OS_MACOS
```

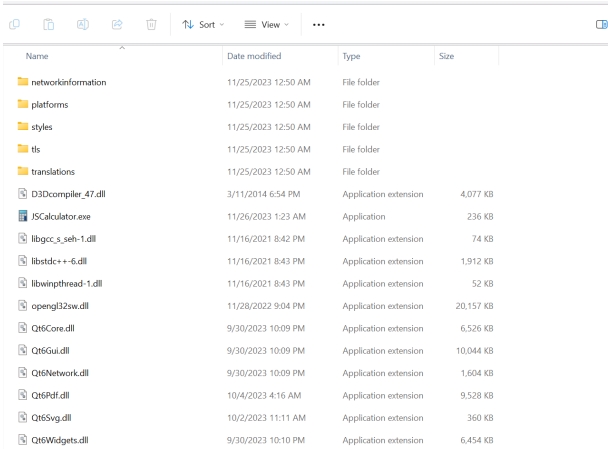
图 1.3 Qt 对应平台的设置

1.4. 如何封包

在 Windows 考虑: windeployqt

在 Linux 考虑 linuxdeployqt 或 Go_AppImage 等

在此不加赘述。



Name	Date modified	Type	Size
networkinformation	11/25/2023 12:50 AM	File folder	
platforms	11/25/2023 12:50 AM	File folder	
styles	11/25/2023 12:50 AM	File folder	
ts	11/25/2023 12:50 AM	File folder	
translations	11/25/2023 12:50 AM	File folder	
D3DCompiler_47.dll	3/11/2014 6:54 PM	Application extension	4,077 KB
JSCalculator.exe	11/26/2023 1:23 AM	Application	236 KB
libgcc_s_seh-1.dll	11/16/2021 8:42 PM	Application extension	74 KB
libstdc++-6.dll	11/16/2021 8:43 PM	Application extension	1,912 KB
libwinpthread-1.dll	11/16/2021 8:43 PM	Application extension	52 KB
opengl32sw.dll	11/28/2022 9:04 PM	Application extension	20,157 KB
Qt6Core.dll	9/30/2023 10:09 PM	Application extension	6,526 KB
Qt6Gui.dll	9/30/2023 10:09 PM	Application extension	10,044 KB
Qt6Network.dll	9/30/2023 10:09 PM	Application extension	1,604 KB
Qt6Pdf.dll	10/4/2023 4:16 AM	Application extension	9,528 KB
Qt6Svg.dll	10/2/2023 11:11 AM	Application extension	360 KB
Qt6Widgets.dll	9/30/2023 10:10 PM	Application extension	6,454 KB

图 1.4 在 Windows 下封包成功结果

2. 实验日志

2.1. 编写程序源码

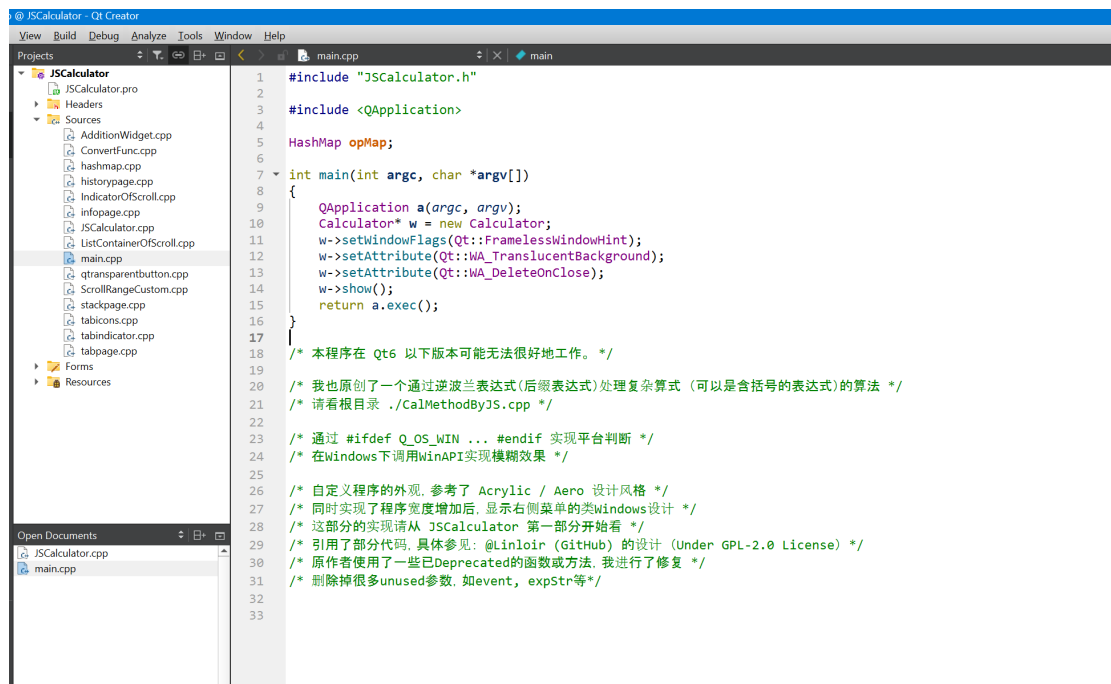


图 2.1 源码成果

说明:

1. 本程序在 Qt6 以下版本可能无法很好地工作。
2. 我也原创了一个通过逆波兰表达式 (后缀表达式) 处理复杂算式 (可以是含括号的表达式) 的算法。请看源码./CalMethodByJS.cpp
3. 通过 #ifdef Q_OS_WIN ... #endif 实现平台判断, 在 Windows 下调用 WinAPI 实现模糊效果
4. 自定义程序的外观, 参考了 Acrylic / Aero 设计风格, 同时实现了程序宽度增加后, 显示右侧菜单的类 Windows 设计。这部分的实现请从 JSCalculator 第一部分开始看。

这里引用了部分代码, 具体参见: @Linloir (GitHub) 的设计 (Under GPL-2.0 License)

原作者使用了一些已 Deprecated 的函数或方法, 我进行了修复, 删除掉很多 unused 参数, 如 event, expStr 等。

2.2. 移植到 Ubuntu 下构建

有了 1.3 的基础, 这个很简单解决。

您可以发现, 在 Linux 下, WindowCompositionAttribute.h 默认被禁用。

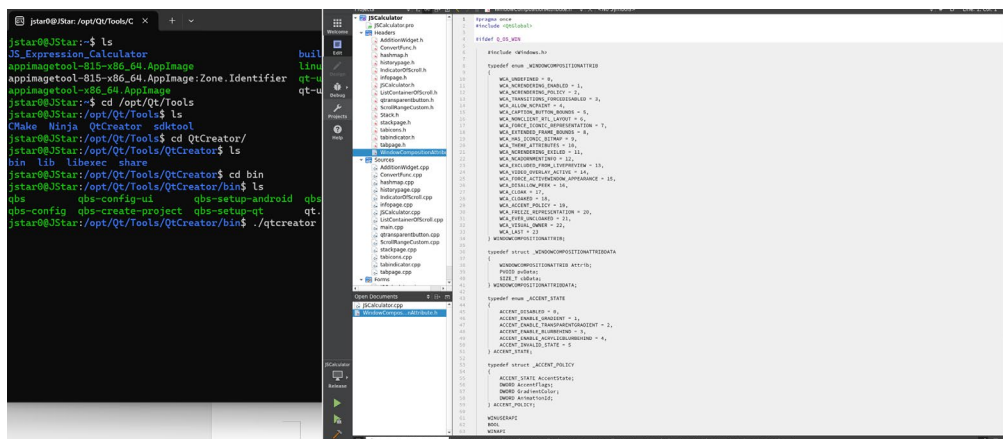


图 2.2-1 在 Linux 下 WindowCompositionAttribute 默认禁用
构建出一个可运行的二进制程序 JSCalculator:

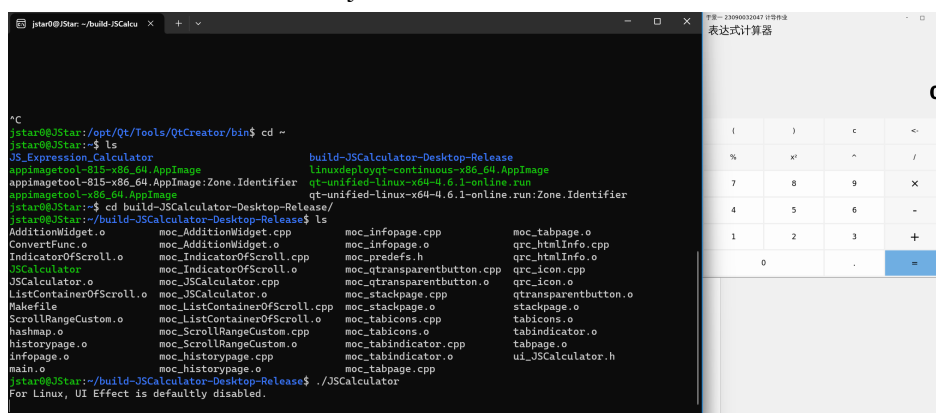


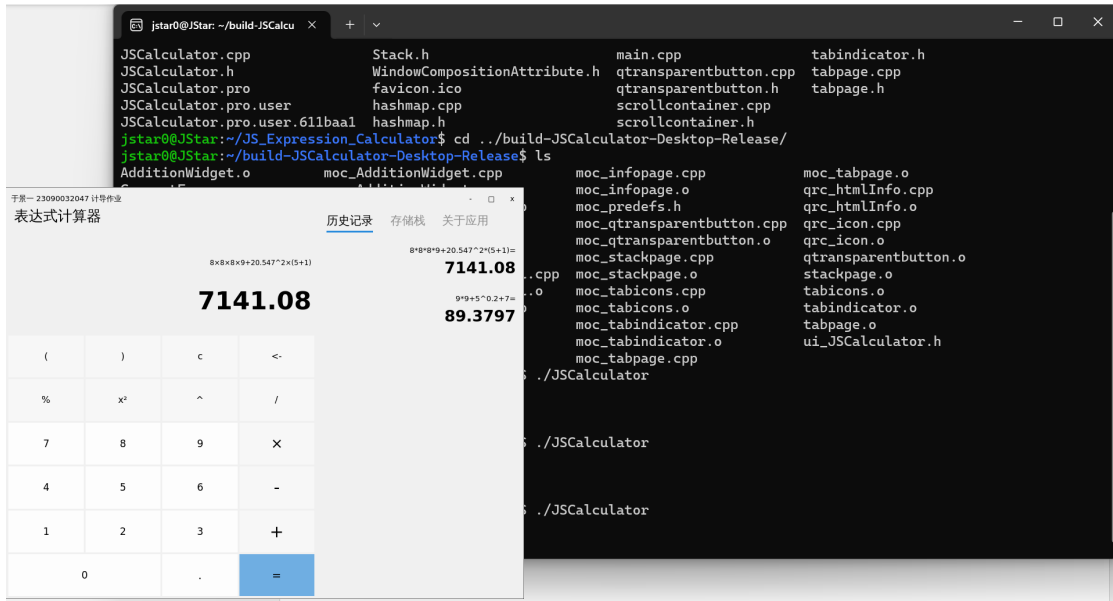
图 2.2-2 在 Linux 下构建
然后进行跨平台处理即可。

3. 正确运行的截图（脱离 QtCreator）

3.1. 在 Windows 下



3.2. 在 Linux (Ubuntu 22.04 LTS) 下



23090032047 于景一

2023/11/26