[移植方案 3](#_Toc5852)

[1. 移植基础 3](#_Toc32415)

[2. 开发环境和配置 3](#_Toc19875)

[2.1. 操作系统 3](#_Toc10143)

[2.2. 开发平台 3](#_Toc30406)

[2.3. IDE 选择 4](#_Toc7021)

[2.3.1. Codeblocks 4](#_Toc15059)

[2.3.1.1. 安装 4](#_Toc10631)

[2.3.2. QtCreator 4](#_Toc22502)

[2.3.2.1. 安装 Qt 4](#_Toc31171)

[3. 移植步骤 5](#_Toc30699)

[3.1. 方案一 5](#_Toc21029)

~~[3.2. 方案二](#_Toc1424)~~ [6](#_Toc1424)

[4. 项目配置 7](#_Toc9333)

[5. MFC2Qt 11](#_Toc24074)

[5.1. 设计思路 11](#_Toc26997)

[5.1.1. UI界面 11](#_Toc28134)

[5.1.2. UI Class 11](#_Toc16109)

[5.1.3. 非GUI类 11](#_Toc31096)

[5.2. 界面设计移植 11](#_Toc9956)

[5.2.1. 界面移植步骤 11](#_Toc5544)

[5.3. MFC相关代码移植步骤 15](#_Toc31159)

[6. CQClass项目 18](#_Toc15396)

[6.1. Cqafx.h 18](#_Toc27132)

[6.2. Cqafxdd\_.h 18](#_Toc32333)

[6.3. Cqafxmsg\_.h 18](#_Toc12356)

[6.4. Cqstdafx.h 19](#_Toc13393)

[6.5. Cqwinuser.h 19](#_Toc26178)

[6.6. Cqafxwin.h 19](#_Toc12882)

[6.6.1. DECLARE\_DYNAMIC和IMPLEMENT\_DYNAMIC 19](#_Toc5095)

[6.6.2. DECLARE\_MESSAGE\_MAP BEGIN\_MESSAGE\_MAP END\_MESSAGE\_MAP 20](#_Toc7052)

[6.6.3. global function declations. 20](#_Toc18015)

[6.6.4. CRC 类 21](#_Toc27602)

[6.6.5. CDataExchange 21](#_Toc2084)

[6.6.6. CWnd 22](#_Toc20961)

[6.6.7. CDialog 25](#_Toc2687)

[6.6.8. CEdit 25](#_Toc2391)

[6.6.9. CCheckBox 26](#_Toc5162)

[6.6.10. CSliderCtrl 26](#_Toc6478)

[6.6.11. CComboBox 27](#_Toc9719)

[6.6.12. CAcModuleResourceOverride 27](#_Toc17648)

[6.7. cqrcc 28](#_Toc3486)

[7. 软件安装包 29](#_Toc9592)

[7.1. 打包命令 29](#_Toc8234)

[7.2. 天河cad软件目录 29](#_Toc24963)

[7.2.1. DEBIAN 30](#_Toc24730)

[7.2.2. 软件安装目录 31](#_Toc21405)

[7.3. 打包 33](#_Toc15533)

[8. 附录 33](#_Toc16997)

[8.1. MFC -> Qt类对照表 33](#_Toc6390)

[8.2. Problem and Solution 34](#_Toc7882)

# 移植方案

涂家聪

**哲学思想**:由简到难,改动最小

# 移植基础

BRX SDK提供了仿真windows API ，支持在 Linux 和 Mac 上使用的大量 Windows API 函数，包括

1 .native (Ansi + Unicode) MSVC Runtime and Windows API 仿真

2.非图形用户界面的 MFC 类 (CArray, CList, CString, etc.)

3.基于 COM 的数据类型和数据类，以及相关的工具函数 (Variant, SafeArray, BSTR, CComBSTR etc.)

在非 Windows 操作系统上不可用的功能包括

1.基于 MFC 的UI扩展 (AcUi, AdUi)

2.COM interfaces callable from C++

3.一些使用 MFC GUI 类（如 CWnd）的未记录和全局函数不可用。

原文链接[BricsCAD V22 - Developer Reference (bricsys.com)](https://developer.bricsys.com/bricscad/help/en_US/V22/DevRef/index.html?page=source/BRX_02.htm)

# 开发环境和配置

## 操作系统

UOS基于Deepin的国产操作系统

## 开发平台

kunhengCAD 基于bricscadv21

## IDE 选择

### Codeblocks

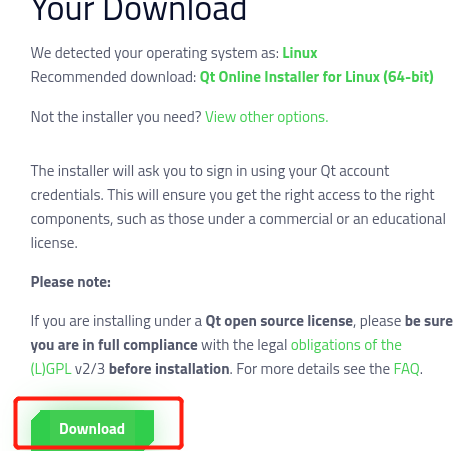
#### 安装

在Codeblocks 官网下载[Binary releases - Code::Blocks (codeblocks.org)](http://www.codeblocks.org/downloads/binaries/" \l "imagesoslinux48pnglogo-linux-32-and-64-bit)，使用 .deb 版本安装。

### QtCreator

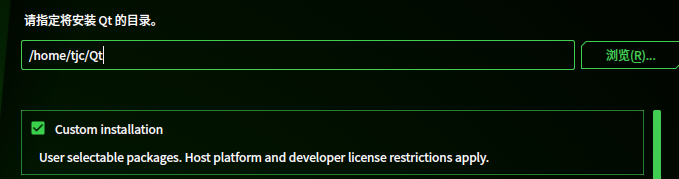
#### 安装 Qt

1. 下载Qt\_linux\_online.run在线安装程序(https://www.qt.io/download-qt-installer)：

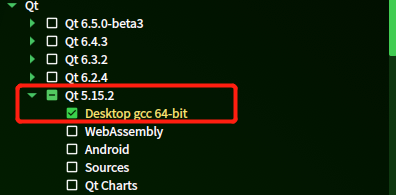


得到Qt\_linux\_online.run文件。

1. 注册登录qt 账号, 勾选自定义安装，然后下一步。



1. 勾选某个版本的Desktop gcc 64-bit组件，然后下一步直到安装。



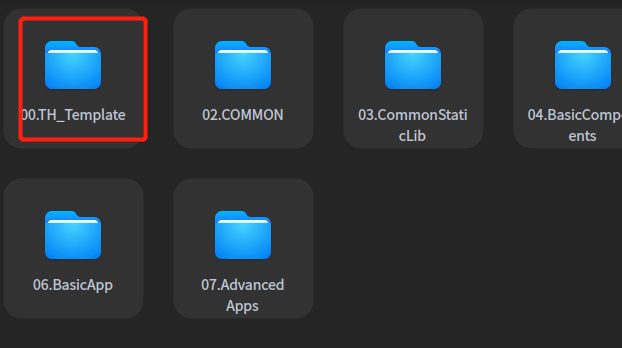
1. 后续可以打开Qt 安装目录下的qt\_maintain 增加删除qt组件。

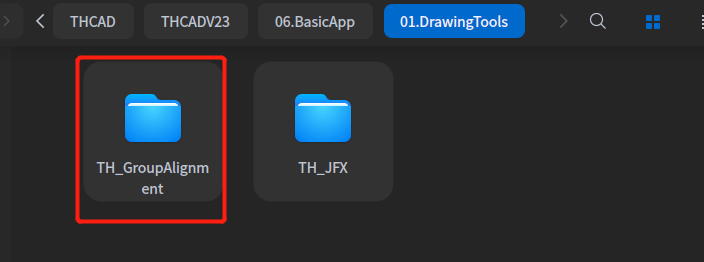
# 移植步骤

## 方案一

以TH\_GroupAlignment 的项目移植为例。

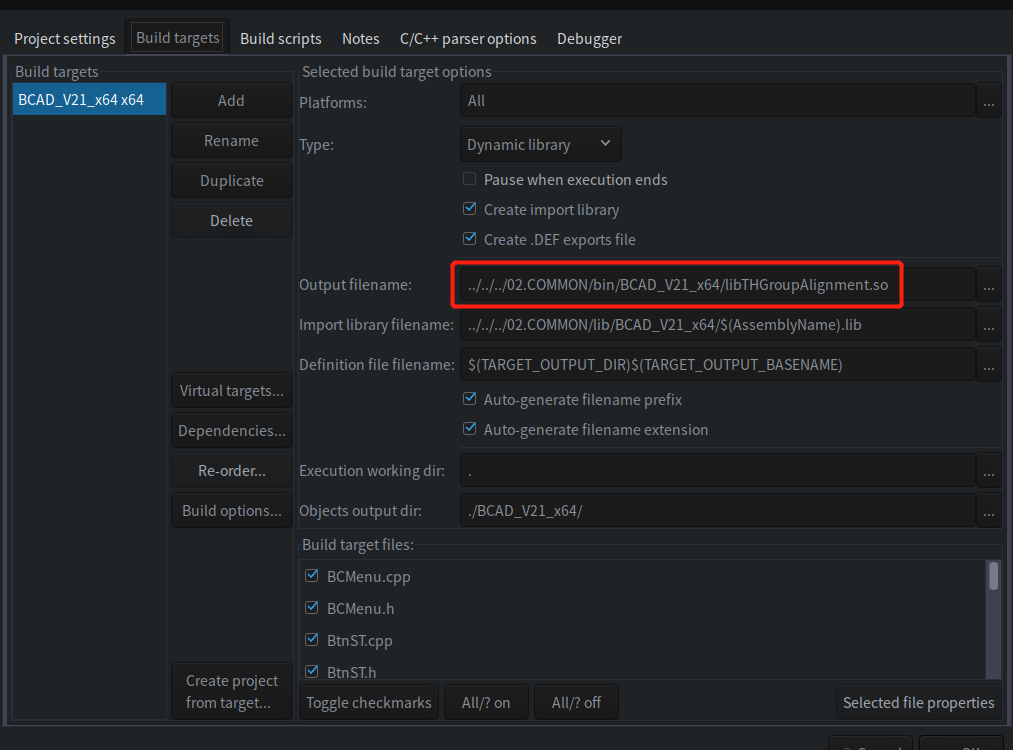
1. 复制/THCAD/THCADV23/TH\_Template到/THCAD/THCADV23/06.BasicApp/01.DrawingTools目录下并改名为TH\_GroupAlignment。





打开TH\_GroupAlignment文件夹，将其中的TH\_Template.cbp文件改为TH\_GroupAlignment.cbp。cbp文件为codeblocks工程文件，双击打开。

1. 修改一些配置如项目名称和生成文件名等

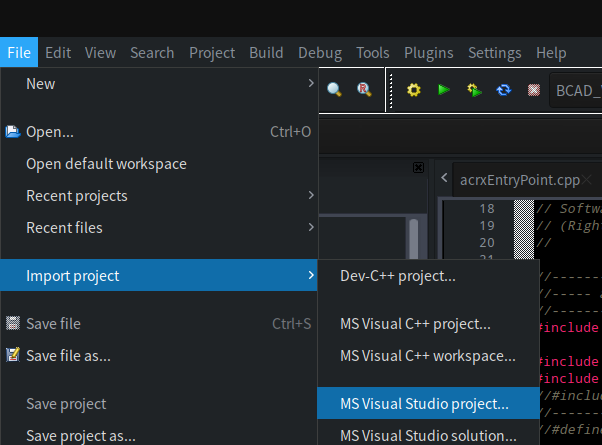


1. 将windows代码复制并添加到项目中。
2. 界面部分替换为Qt([MFC2Qt](#_MFC2Qt))
3. 移植其他部分(遇到问题可查看[Problem and Solution](#_Problem and Solution))。

直接编译，对于报错一一解决。

## ~~方案二~~

1. ~~将windows下的项目复制到项目应该存放的目录~~
2. ~~在codeblocks 中选择导入项目中的MS Visual Studio project~~

~~~~

~~3.修改一些配置如项目名称和生成文件名等([项目配置](#_项目配置))~~

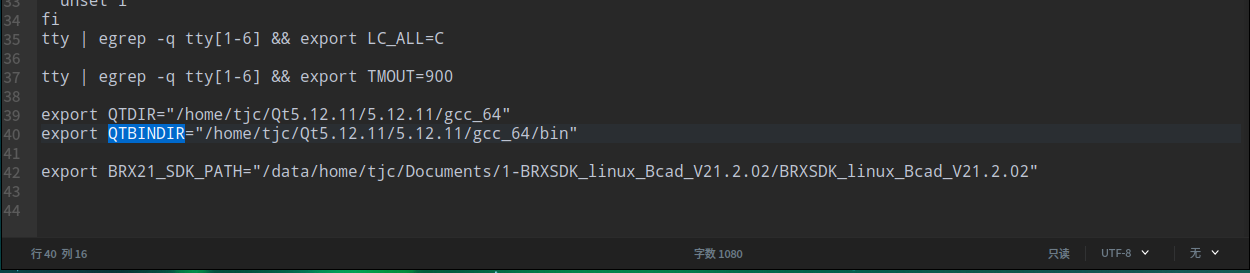
~~4.界面部分替换为Qt(~~**~~[MFC2Qt](#_MFC2Qt)~~**~~)~~

~~5.移植其他部分(遇到问题可查看[Problem and Solution](#_Problem and Solution))~~

# 项目配置

配置可以参考TH\_Template

1. 添加BRX21\_SDK\_PATH ;QTDIR;QTBINDIR,etc 等环境变量



以root身份编辑/etc/profile,在文件最后添加一下几行

export QTDIR="/home/tjc/Qt5.12.11/5.12.11/gcc\_64"

export QTBINDIR="/home/tjc/Qt5.12.11/5.12.11/gcc\_64/bin"

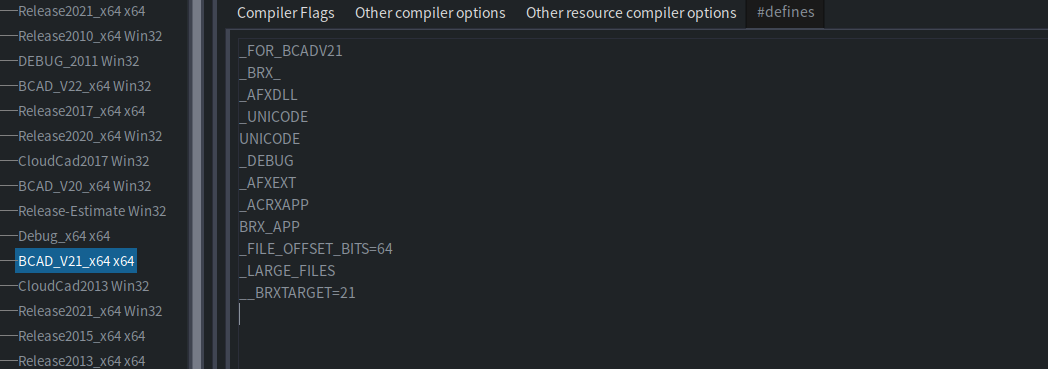
export BRX21\_SDK\_PATH="/data/home/tjc/Documents/1-BRXSDK\_linux\_Bcad\_V21.2.02/BRXSDK\_linux\_Bcad\_V21.2.02"

具体路径以实际为准

1. Build option

单击左边的build target 配置特定的target,当前配置的是BCAD\_V21\_x64

* 1. 添加Defines.



根据需要自行删减

\_BRX\_

\_AFXDLL

\_UNICODE

UNICODE

\_DEBUG

\_AFXEXT

\_ACRXAPP

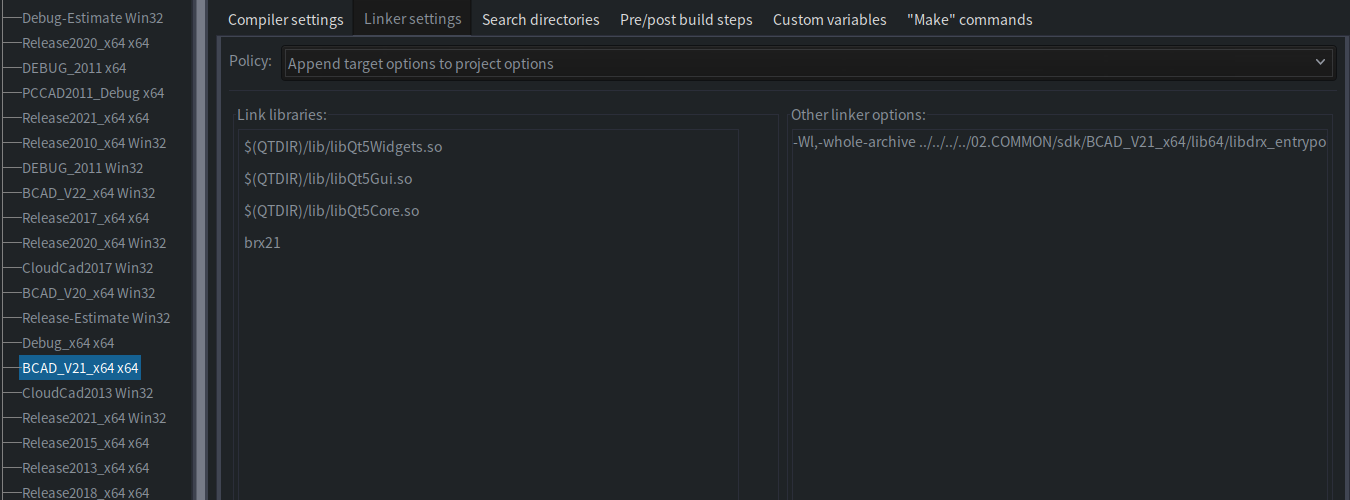
BRX\_APP

\_FILE\_OFFSET\_BITS=64

\_LARGE\_FILES

\_\_BRXTARGET=21

* 1. link配置.



①link libraries

$(QTDIR)/lib/libQt5Widgets.so

$(QTDIR)/lib/libQt5Gui.so

$(QTDIR)/lib/libQt5Core.so

根据本地安装的Qt自行修改

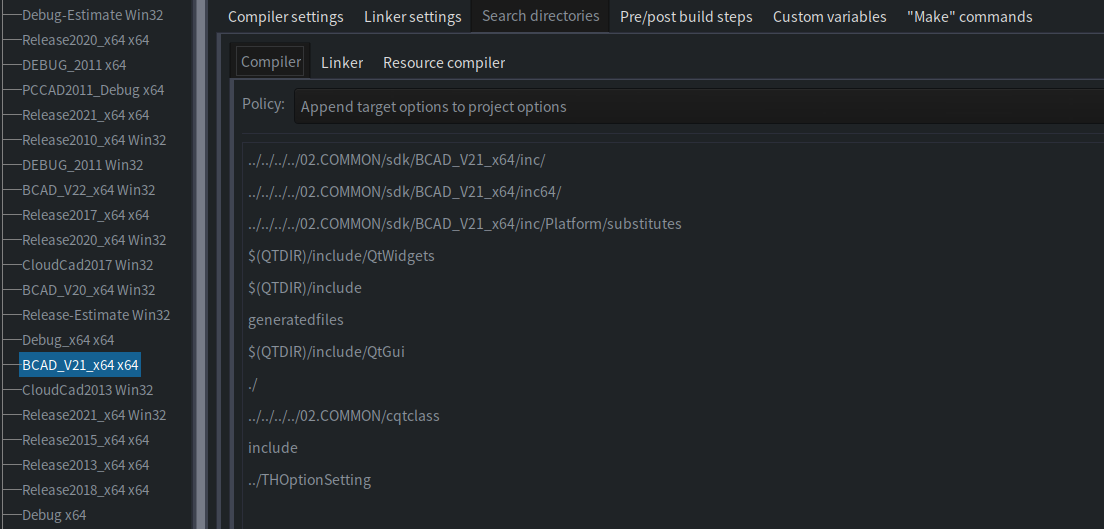
②other link options

-Wl,-whole-archive ../../../../02.COMMON/sdk/BCAD\_V21\_x64/lib64/libdrx\_entrypoint.a -Wl,-no-whole-archive

Note: 上面是一行，这里太短装不下，brx\_linux要求必需有这个选项。

* 1. Search directory.

①compiler path



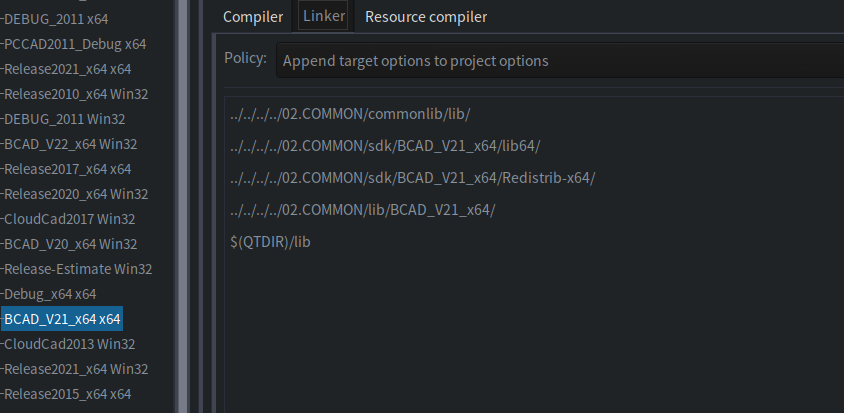
- 1-3行:brx\_sdk 包含目录

- 4-7行：Qt include path

- generatedfiles Qt 生成文件目录

- cqtclass 包含目录

②link path

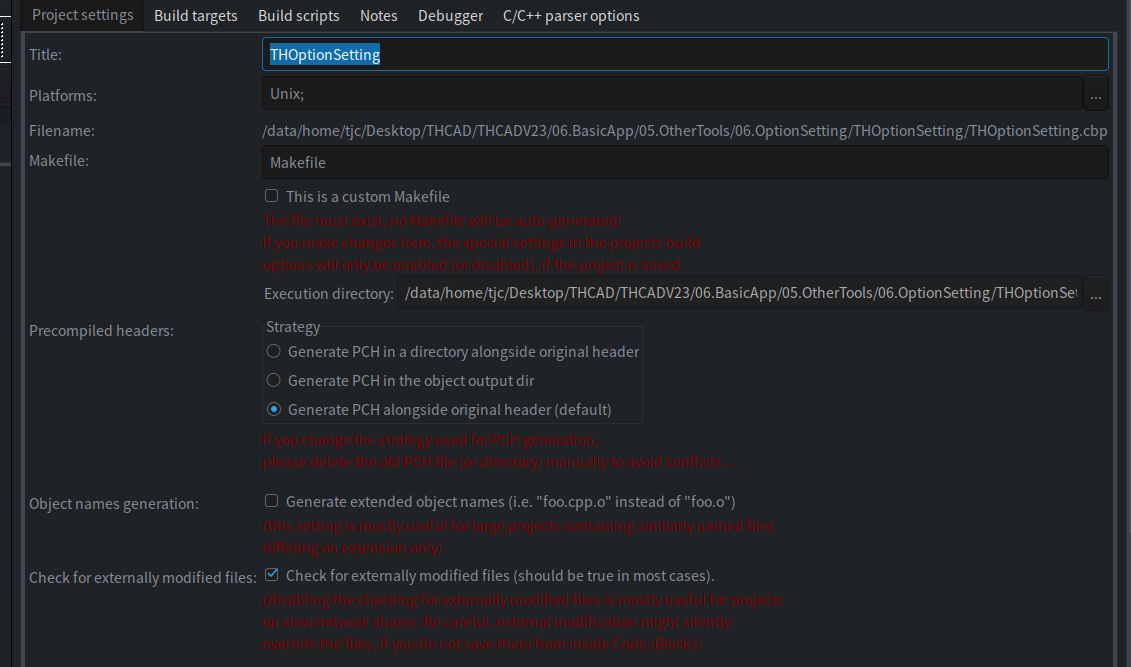


- line 1:commonlib(THCAD lib) path

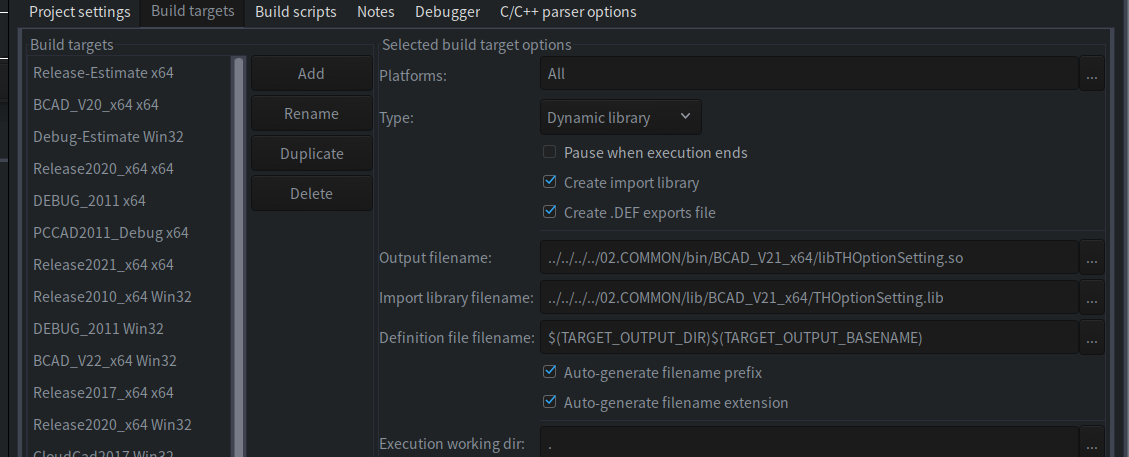
- lines 2-4:brx\_sdk library path

- line 5:Qt library path

1. Properties
   1. Project setting



* 1. Build target



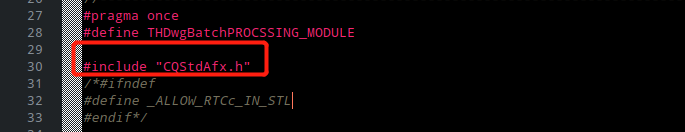
- Type: dynamic library.

- output filename: “../../../../02.COMMON/bin/BCAD\_V21\_x64/libTHOptionSetting.so”,放到整个解决方案的bin目录中

- import library filename:”../../../../02.COMMON/lib/BCAD\_V21\_x64/THOptionSetting.lib”,同上

- 其他如图

1. Stdafx.h 修改
   1. Stdafx.h开始处添加如下内容以兼容linux.



#include "CQStdAfx.h"

# MFC2Qt

## 设计思路

通过Qt来移植MFC程序，GUI 相关的工作量是非常大的，如果能用simulator和automatic脚本，使用现成的代码和资源文件都能极大地减少工作量。我们封装Qt的类，让他具备和MFC类一样的名字和大体相同的功能，这些类统称为CQtClass。

### UI界面

MFC使用文本UI设计 rc+resource.h, Qt ui文件使用 xml 文件，若写出脚本转换rc+resource 为 ui\_xml ，再用uic 转换，最后人工审查一下，可以省去界面设计这一步。最后结果可能与预期有偏差。

目前方案：手动设计，对照原来mfc界面的设计，在Qt designer 中用功能相同的控件替代，缺点是工作量很大。

### UI Class

UI class使用基于Qt的CQtClass simulator，让很多代码可以重用。包括 CWnd ,CDialog ,CButton ,CEdit ...

### 非GUI类

非GUI类(CArray,CString,...)可以直接使用

## 界面设计移植

### 界面移植步骤

对于有ui设计的类，需要将MFC 的设计转换为Qt ui设计，目前采用手工转换。

图2-6 是MFC设计界面，图2-7是Qt设计界面(不要在意颜色，颜色跟随系统主题)。以下是设计Qt界面的步骤：

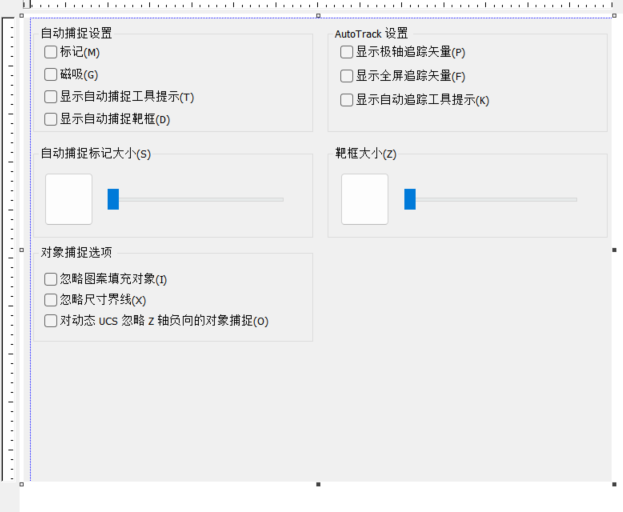
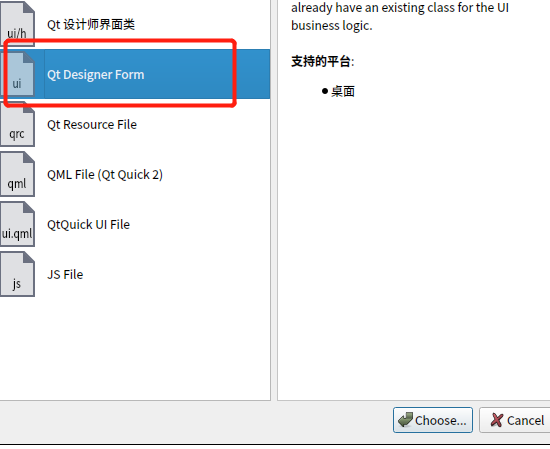


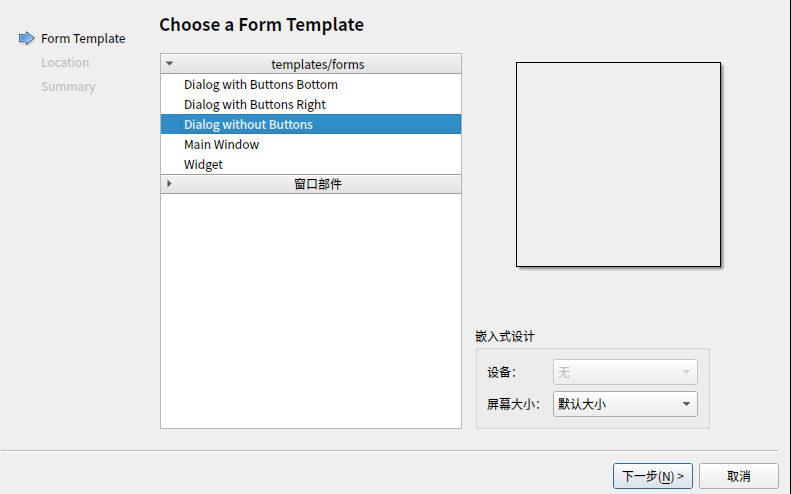
图 2-6

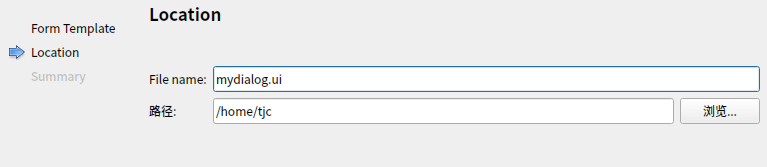


图 2-7

1. 生成一个空的design form 文件，比如mydialog.ui，在具体的移植工作中mydialog改为dialog的类名。打开mydialog.ui文件进行页面设计。







1. 在Qt Designer中找到对应的控件，如MFC 的CCheckBox在Qt中是QCheckBox,CEdit对应QLineEdit等等(详见**[Qt2MFC控件](#_MFC -> Qt类对照表)**)。将控件摆放在合适的位置，完成后可以得到图2-7所示界面。

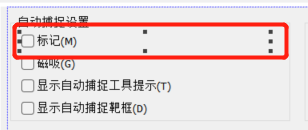


图 2-8



图 2-9

1. 设置好控件名字

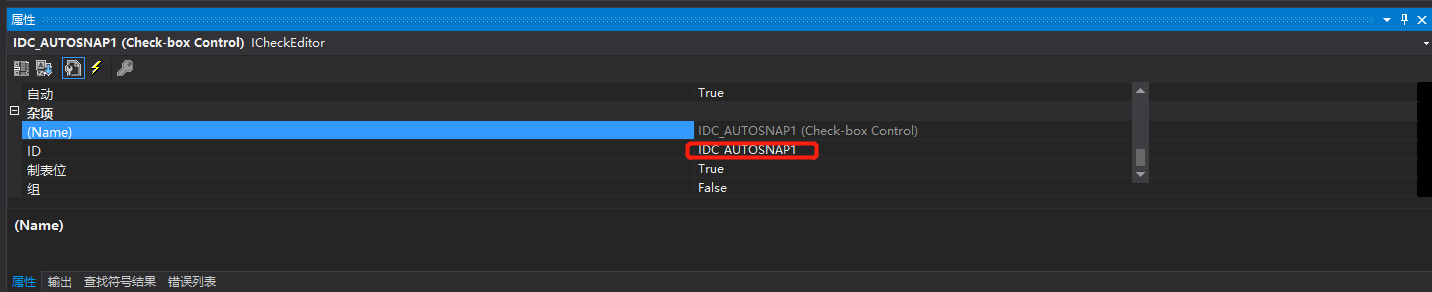


图 2-10



图 2-11

对应的控件ID应该相同，在MFC 中为ID,Qt 中为objectName。

1. 界面设计完成后保存，打开终端，通过uic工具生成ui\_xxx.h文件。



图 2-12

例如，有一个dialog类名为MyDialog，则对应的代码为

uic MyDialog.h -o ui\_MyDialog.h

1. 编辑ui\_qtform.h。添加头文件包含 #include “cqtafxwin.h”，如图2-14所示,将Qt 中的控件类包括(QDialog)改为CQtClass 控件类，如QCheckBox -> CCheckBox ，QLineEdit -> CEdit , ... **[参照附录表3-1](#_MFC -> Qt类对照表)**(可以通过Ctrl+R 快速替换)。~~将Qt 类换成CQtClass 类的目的是建立CQtWnd access tree，使得CWnd 的一些函数如GetDlgItem 功能得以实现~~。

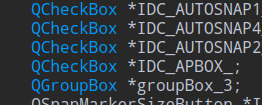


图 2-13

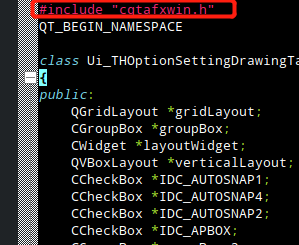


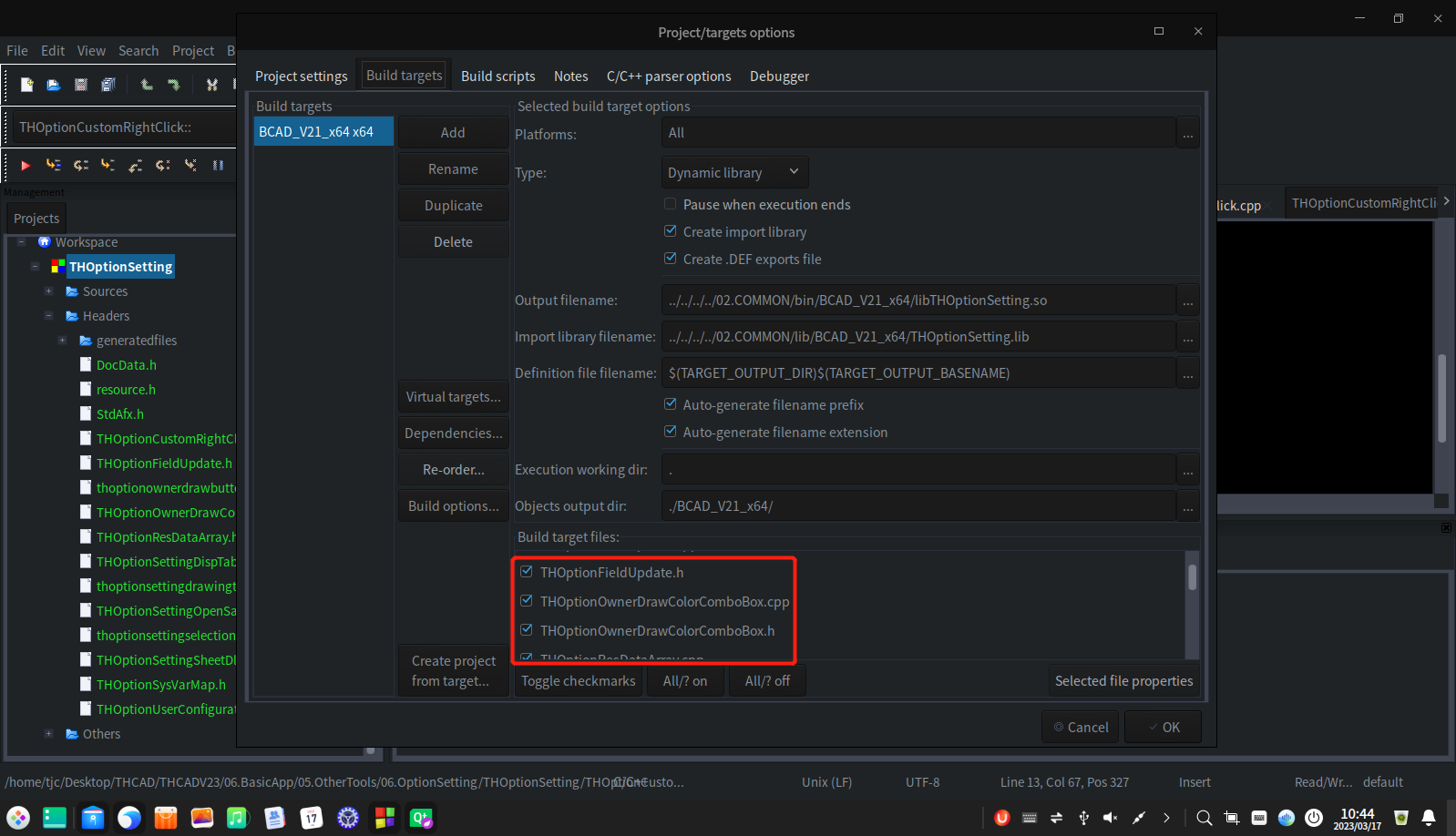
图 2-14

1. 做好上述工作，界面部分移植完成。

## MFC相关代码移植步骤

注：所有生成的文件都需要加到项目中，并且勾选编译。

若文件有修改，切记先保存再用命令行重新生成。



1. 界面部分: xxx.ui 通过命令uic 生成 ui\_xx.h



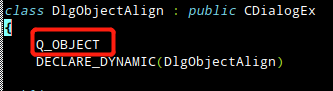
uic mydialog.ui -o ui\_mydialog.h

1. resource.h 使用cqrcc 脚本生成 resource.cpp，在命令行中运行如下命令。

./cqrcc resource.h resource.cpp Module\_Name

Module\_Name为项目名称

3. cqclass 类应添加Q\_OBJECT宏，一般放在类声明第一行。

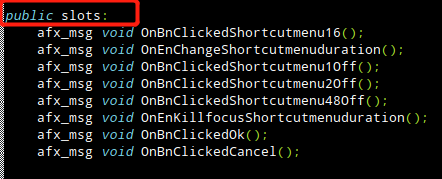


4. 对于没有界面部分的类,注释掉DECLARE\_DYNAMIC 和IMPLEMENT\_DYNAMIC宏，同理对于有界面的窗口类，需要添加这两个宏。





5由于消息机制不同,响应消息的函数需要用public slots: 明确声明。在头文件中，所有的响应函数都要用public slots: 声明。



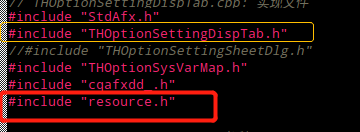
6有宏Q\_OBJECT的类的头文件mydialog.h通过脚本 moc生成 moc\_xx.cpp 添加至项目

moc mydialog.h -o moc\_mydialog.cpp

7 qt 资源文件 qrc 通过 rcc 命令生成qrc\_XXX.cpp 并添加至项目，没有用到qrc可以跳过这一步。

qrc myproject.rc -o qrc\_myproject.cpp

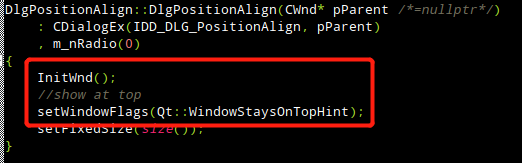
8 所有的 {#include "resource.h"} 放在mydialog.cpp 头件中，不要放在mydialog.h中，将{#include "resource.h"}放在{#include “mydialog.h”}下面,避免错误发生(在 resource.h 中定义的控件ID 为宏,而在ui\_xxx.h 中为变量名，若{#include "resource.h"}在{#include "mydialog.h"}前面，会发生定义报错)。



9 在MyDialog类的构造函数的开始处添加如下代码：

InitWndUI();// initial ui of this dialog.

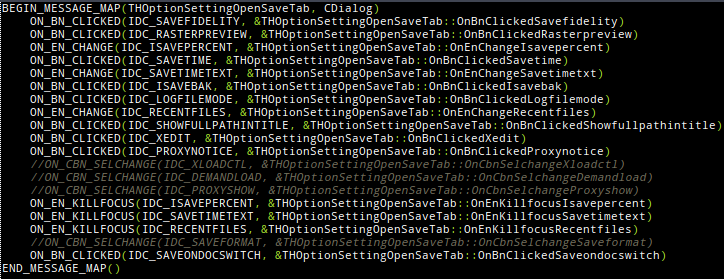
setWindowFlags(Qt::WindowStaysOnTopHint); //show at top



10 所有的CAcModuleResourceOverride tempResourceHere();改为：

CAcModuleResourceOverride tempResourceHere(CURRES);//要求包含resource.h 且resource.cpp 已经添加到项目中。

11 在MESSAGE\_MAP 中，对于暂不支持的事件进行注释(编译时报错：ON\_CBN\_SELCHANGE未定义)。



12 做好上述工作，界面移植部分完成。

# CQClass项目

## Cqafx.h

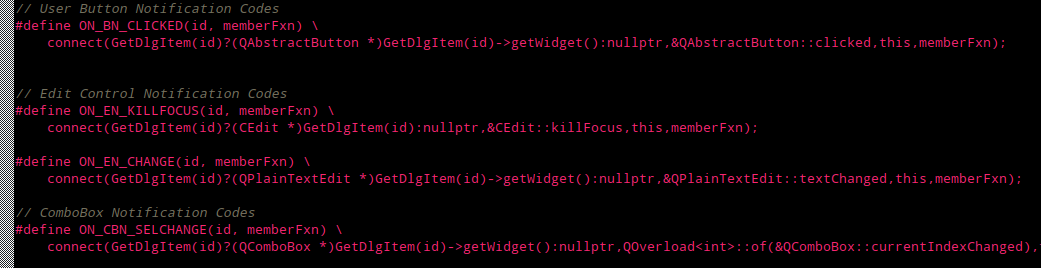
对应于windows 上的afx.h 文件

## Cqafxdd\_.h

对应于windows 上的afxdd\_.h 文件

## Cqafxmsg\_.h

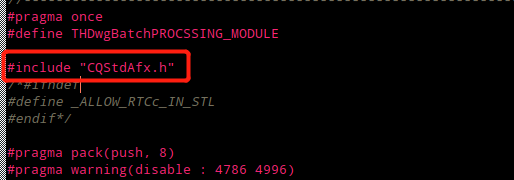
对应于windows 上的afxmsg\_.h文件



其中主要定义了MESSAGE\_MAP 中相应的宏,通过id找到对象，再通过宏名确定相应的事件信号，由qt中的connect将其关联，从而实现信号和处理函数的绑定。

## Cqstdafx.h

将该文件放在StdAfx.h 中的开始处来获取cqclass对项目的支持

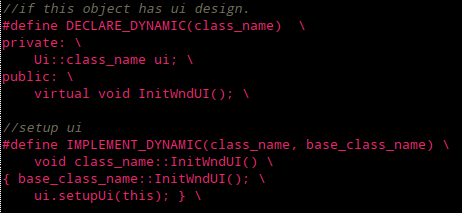


## Cqwinuser.h

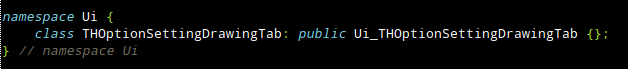
对应于windows 上的winuser.h文件,定义了winuser.h中相同的宏，使编译能正常通过。

## Cqafxwin.h

### DECLARE\_DYNAMIC和IMPLEMENT\_DYNAMIC

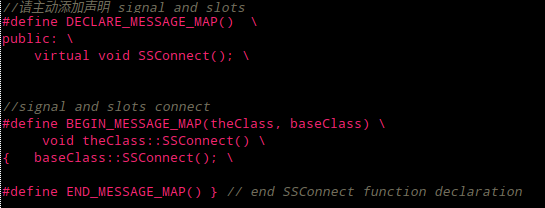


如果一个类有对应的ui界面文件，在类的头文件中使用DECLARE\_DYNAMIC声明，在cpp文件中使用IMPLEMENT\_DYNAMIC实现。在DECLARE\_DYNAMIC中，声明了成员变量Ui::class\_name ui，这个变量的类型在相应类的ui\_xxx.h文件中声明，如下图

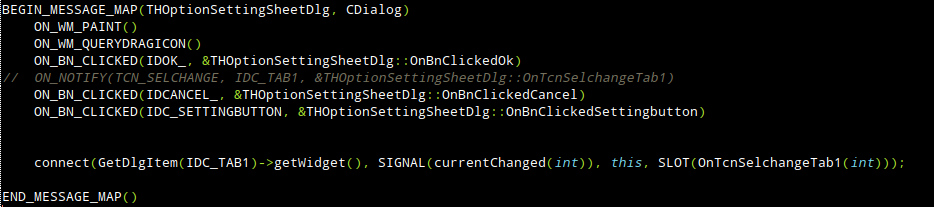


同时声明重写继承于CWnd的InitWndUI函数。在IMPLEMENT\_DYNAMIC中，通过 {ui.setupUi(this)}设置类的ui界面。

### DECLARE\_MESSAGE\_MAP BEGIN\_MESSAGE\_MAP END\_MESSAGE\_MAP



DECLARE\_MESSAGE\_MAP声明重写继承于CWnd的SSConnect函数。BEGIN\_MESSAGE\_MAP END\_MESSAGE\_MAP共同构成了SSConnect() 的实现。



### global function declations.

int GetWidgetType(QObject\* pObj);

此函数输入一个QObject对象，返回值是一个枚举类QtWidgetType，定义了qt控件的类型。

const CRC\* GetCurRes();// get Qt Widget Type

void SetCurRes(const CRC\* pRc);// set current resource

这两个函数获取和设置当前的资源，当前资源由一个全局的指针变量管理。它在cqafxwin.cpp 中声明。

const CRC\* g\_curRes = nullptr;

bool ID2StrList(int nID, QStringList& strList);// id to string list

该函数通过当前资源读取到这个整数对应的所有字符串，返回一个字符串列表。

std::map<std::string, const CRC\*> g\_mapRes; //管理所有的资源

Key: 模块名，value:资源对象

void AddNewRes(std::string strModuleName, const CRC\* pRes);//add new resource to res map

const CRC\* GetModuleRes(std::string strModuleName);//get module resource by module name

### CRC 类

此类为资源类，用于封装一个项目的资源。

Header: #include “cqafxwin.h”

**Properties:**

//key: resource.h ID ; value: resource.h 中对应的宏的字符串;

std::multimap<int, std::string> m\_mapRC;

**Public functions:**

//return m\_mapRC.

const std::multimap<int, std::string>& getRC() const;

//insert a pair <int,string> to resource map

void insert(int nID, const std::string& str);

//return a string list which contains those strings corresponed given id

bool id2StrList(int nID, QStringList& strList)const;

Detailed Description：

此类为资源类，用于封装一个项目的资源,将 mfc的 resource.h 中的宏字符串和数字对应, 一个项目一个CRC实例。通过cqrcc 脚本解析并修改项目的resource.h，生成resource.cpp。

Resource.h 中声明了一个CRC 实例，在resource.cpp中实现。

CRC 中管理一个成员变量std::multimap<int, std::string> m\_mapRC，其中包含resource.h 中值与宏名字的对应关系。

### CDataExchange

for data exchange and validation

Header: #include “cqafxwin.h”

**Properties:**

// TRUE => save and validate data

BOOL m\_bSaveAndValidate;

// container usually a dialog

CWnd\* m\_pDlgWnd;

UINT m\_idLastControl; // last control used (for validation)

BOOL m\_bEditLastControl; // last control was an edit item

// data exchange lock

BOOL m\_bLocked;

**Public functions:**

//Constructor

CDataExchange(CWnd\* pDlgWnd, BOOL bSaveAndValidate = 0);

Detailed Description：

用于保存数据交换时基本的参数。

### CWnd

同Windows CWnd。

Header: #include “cqafxwin.h”

**Properties:**

QWidget\* m\_pWnd;//使用该指针指向的对象实现窗口功能

CString m\_str;//窗口文本

CWnd\* m\_pParentWnd;// parent window.

std::vector<CWnd\*> m\_vecChildren;//Children

bool m\_bNewWgtFlag; //纪录m\_pWnd是否为构造函数 new 出

CDataExchange\* m\_pDataExchanger;//用于变量和界面数据的交换

**Public Function:**

//Constructor

CWnd(QWidget\* pWnd, CWnd\* pParentWnd);

// cqt ui support

virtual void InitWndUI();

// cqt msg map support

virtual void SSConnect();

// cqt system initialize

virtual void InitWnd();

Detailed Description:

CQtUI class 中最基础的类是CWnd,它几乎具备一个窗口的所有功能，CWnd 不继承任何类，它的成员和方法大部分与MFC中的CWnd 一样，不同的是它通过封装Qt来实现。以下是它的成员和方法。

- 构造函数

CWnd(QWidget\* pWnd, CWnd\* pParentWnd); //构造函数

QWidget\* m\_pWnd;//使用该指针指向的对象实现窗口功能

CWnd的成员m\_pWnd指向一个窗口QWidget,CWnd的所有功能都是通过该指针指向的对象实现，图2-2是它的构造函数。

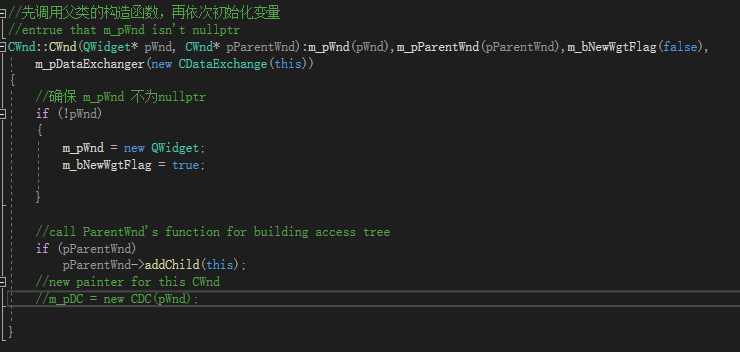


图 2-1

在构造函数中需要传入一个QWidget 用于修饰，m\_pParentWnd是父窗口,如果传入的QWidget 为空，则new 一个对象用于修饰，并将标识m\_bNewWgtFlag设为true。此外，还new 了其他需要的对象，如负责数据交换的DataExchanger。如果父窗口不为空，调用父窗口的addChild 来建立MFC access tree。

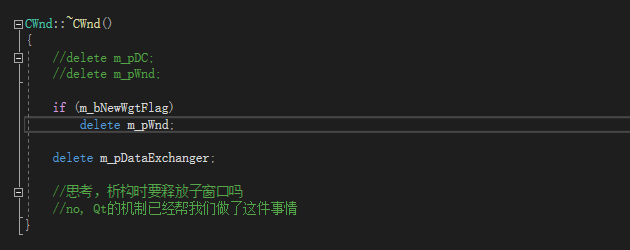


图 2-2

图2-2是CWnd的析构函数，一般控件类都需要声明为虚析构，这样做是考虑到父类指针释放子类对象的情况。在~CWnd中，如果m\_bNewWgtFlag为true,说明pWnd是new出来的需要释放，此外还需要释放其他在构造函数中new 出来的对象。

1. CWnd method

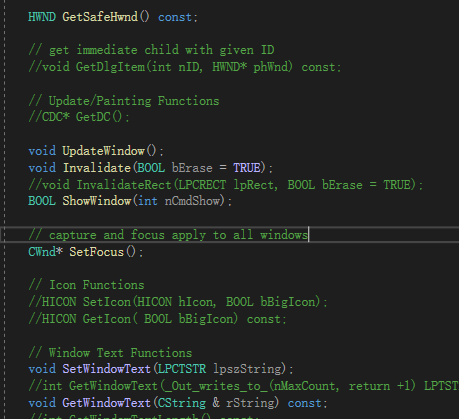


图 2-3

图2-3 展示的是和MFC 同名同参的method，实现这些函数可以让MFC CWnd 在非windows操作系统正常编译运行。

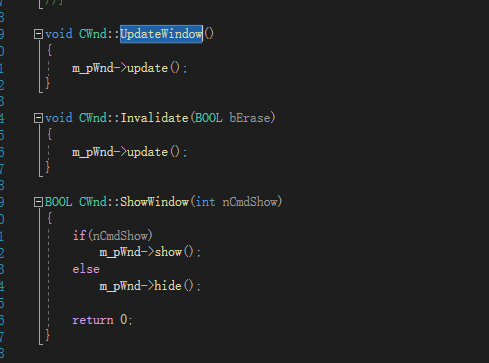


图 2-4

图2-4为图2-3 method的实现，可以看到CWnd是通过修饰一个QWidget对象实现的。

1. 特殊的method

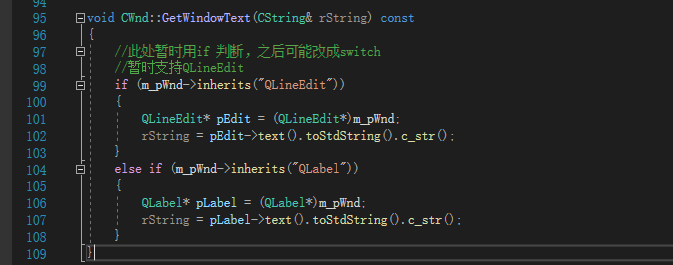


图 2-5

MFC CWnd中有一个CString text成员，用于保存窗口文本，而QWidget 并没有这个成员，如果在CQt CWnd要实现这个CString text成员,会很繁琐，所以这里用了一个朴素的办法，通过判断CWnd对象所修饰的QWidget的类型来实现GetWindowText,SetWindowText。

### CDialog

同Windows CDialog。

Header: #include “cqafxwin.h”

Inherits: QDialog, Cwnd

**Properties:**

bool m\_bInitialed; // 初始化标记

**Public Function:**

//Constructor

explicit CDialog(UINT nIDTemplate, CWnd\* pParentWnd = NULL);

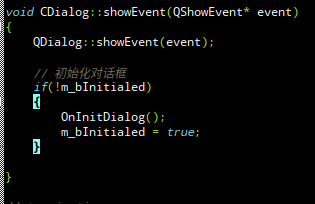
//override showevent for calling OnInitDialog() before first show.

virtual void showEvent(QShowEvent\* event);

Detailed Description:

CDialog继承于QDialog和CWnd类来实现具体函数的功能。

重写了QDialog的virtual void showEvent(QShowEvent\* event);



用于在第一次show的时候调用初始化函数OnInitDialog.

### CEdit

同Windows CEdit。

Header: #include “cqafxwin.h”

Inherits: QPlainTextEdit, Cwnd

**Public Function:**

//Constructor

CEdit(CWnd\* pParentWnd = nullptr);

void setLineEdit(bool);//true if edit control behave as a line edit.

//operator=

CEdit& operator=(const CEdit&);

virtual void keyPressEvent(QKeyEvent \*e);//override key press event.

virtual void focusOutEvent(QFocusEvent \*e);//for kill focus event.

signals:

void killFocus();// emit when focus out

**Properties:**

bool m\_bLineEdit;// true if edit control behave as a line edit.

Detailed Description:

CEdit inherits QPlainTextEdit and CWnd to implement the function of the Edit control.

### CCheckBox

同Windows CCheckBox。

Header: #include “cqafxwin.h”

Inherits: QCheckBox, Cwnd

**Public Function:**

//Constructor

CCheckBox(CWnd\* pParentWnd = nullptr);

//operator=

CCheckBox& operator=(const CCheckBox &);

重载= 操作符以实现数据交换。

Detailed Description:

CCheckBox继承于QCheckBox和CWnd类来实现相应的功能。

重载= 操作符以实现数据交换。

### CSliderCtrl

同Windows CSliderCtrl。

Header: #include “cqafxwin.h”

Inherits: QSlider, Cwnd

**Properties:**

NMHDR m\_NMHDR; //

LRESULT m\_Result;

**Public Function:**

//operator=

CSliderCtrl& operator=(const CSliderCtrl&);

重载= 操作符以实现数据交换。

**signals:**

void valueChangedLink(NMHDR\* pNMHDR, LRESULT\* pResult);//

**public slots:**

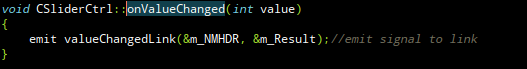
void onValueChanged(int value);

Detailed Description:

为了兼容Windows上的CSliderCtrl 的响应函数，添加了如下代码:

在构造函数中





当QSlider emit valueChanged 信号的同时会emit valueChangedLink信号。

### CComboBox

同Windows CComboBox。

Header: #include “cqafxwin.h”

Inherits: QComboBox, Cwnd

**Public Function:**

//set content

void setContent(LPCTSTR lpszString);

传入以;分隔的宽字符串，将复选的内容设为各子字符串。

//operator=

CSliderCtrl& operator=(const CSliderCtrl&);

重载= 操作符以实现数据交换。

### CAcModuleResourceOverride

Header: #include “cqafxwin.h”

**Properties:**

const CRC\* m\_oldRes;

**Public Function:**

//Constructor

CAcModuleResourceOverride(const CRC\*);

传入CRC对象，将当前资源设为该对象。

//Destructor

~CAcModuleResourceOverride();

将资源设为原先值。

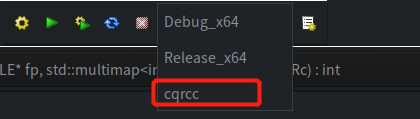
Detailed Description:

使用如下代码将资源设为当前项目值。

CAcModuleResourceOverride tempResourceHere(CURRES);

## cqrcc

将解决方案配置修改为cqrcc单独生成cqrcc脚本程序。



Cqrcc 通过解析resource.h 来生成resource.cpp 并在resource.h中做一些修改。

对于生成的cqrcc程序，你可以在终端中输入如下类似的命令：

cqrcc resource.h resource.cpp Module\_Name

来解析resource.h。使用该命令后,resource.h中追加了几行代码：

#include"cqafxwin.h"

class CRCModuleName:public CRC{public:CRCModuleName();};

#define CURRES &g\_ResModuleName

extern CRCModuleName g\_ResModuleName;

声明了一个 CRC的子类:CRCModuleName，并在该项目的resource.h 中定义CURRES 为 &g\_ResModuleName。

Resource.cpp中定义了CRCModuleName的一个实例（作为全局变量），这个实例在构造函数中插入了resource.h 中值与宏名的对应关系，使得通过ID找到对象这个功能得以实现。此外，在CRCModuleName的构造函数中通过

AddNewRes("CRCModuleName",this);

将该实例的指针添加资源管理器中，之后便可以通过全局函数GetModuleRes获得资源。

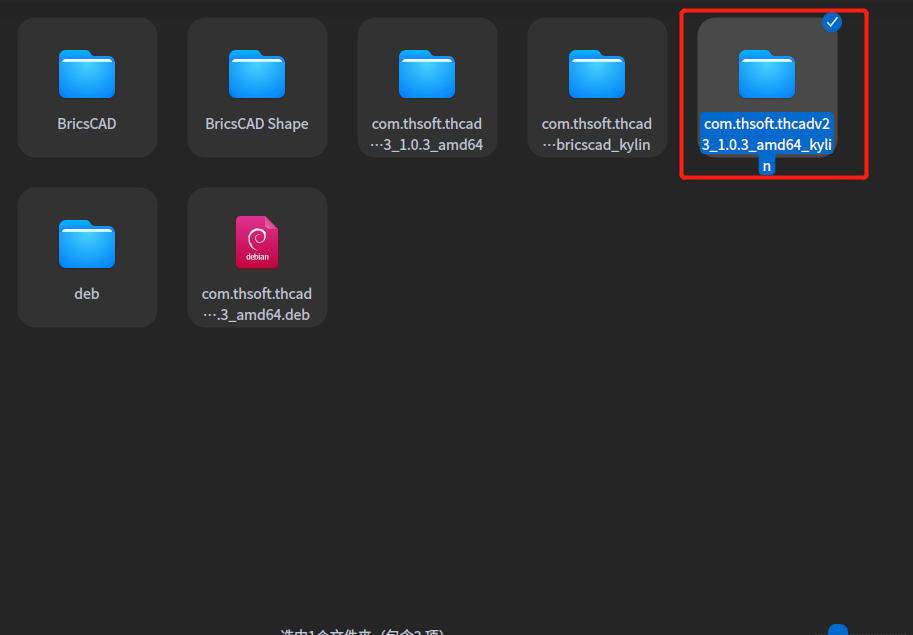
# 软件安装包

## 打包命令

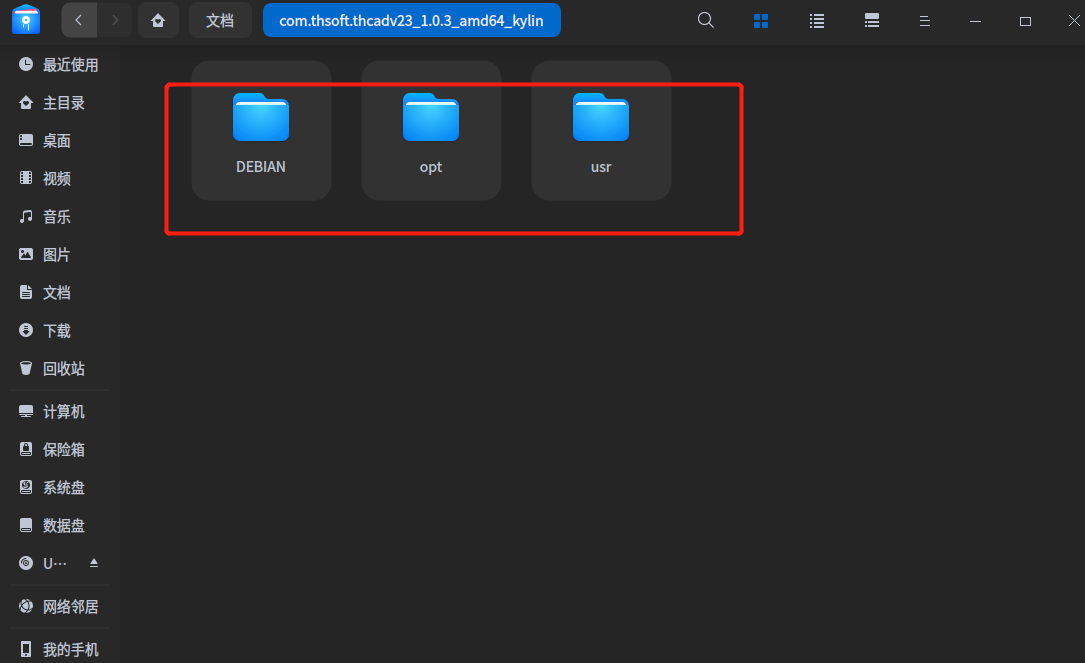
在uos 和 kylin 系统上使用 dpkg 命令生成deb 软件安装包。可参考链接：[Ubuntu中deb包详解及打包教程](https://blog.csdn.net/xiaoyafang123/article/details/123985726)。

## 天河cad软件目录

所有文件在打包成deb之前放在com.thsoft.thcadv23\_xxx\_xxx 文件夹中，需要注意的是，com.company.appname\_version\_architecture 是软件包的命名规范。

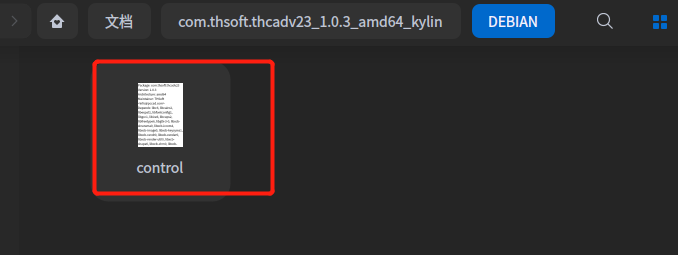


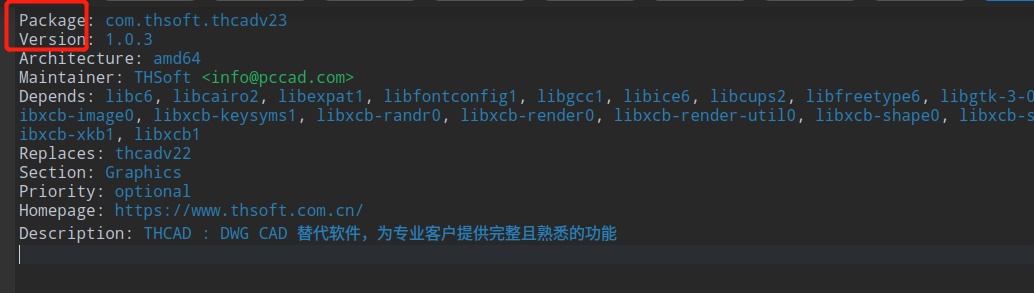
对于麒麟安装包其中有三个子文件夹，如图。对于UOS系统，usr文件夹需要删除掉。



### DEBIAN

此文件夹存放打包配置相关文件，当前只有 control 文件。





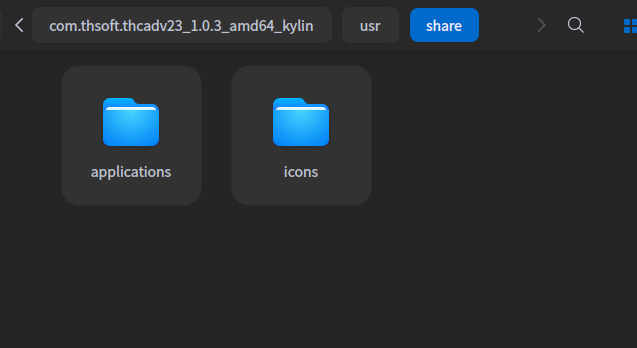
安装包中的命名要规范统一，当前安装包各文件命名符合要求，如非必要请勿更改。以下列出需要统一规范之处：

1. DEBIAN/control 中的package 字段命名规范:com.company.appname.
2. 桌面文件命名为: [package].desktop.
3. 图标命名为 [package].png or [package].xxx.
4. 版本号要统一。

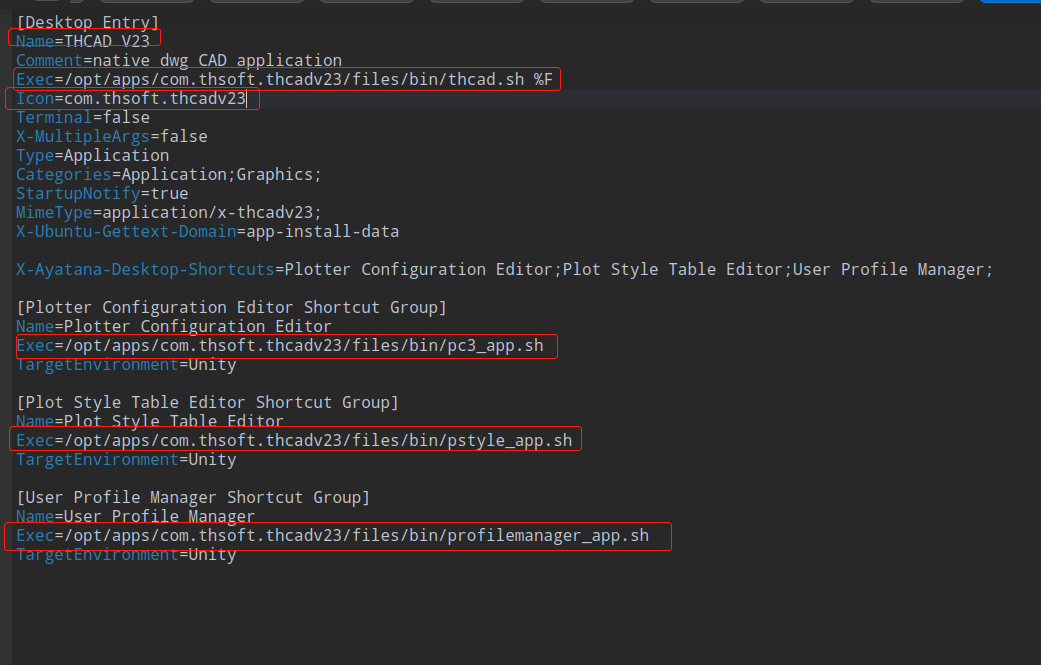
### 软件安装目录

除了DEBIAN 这个配置文件夹外，其它文件夹都会在安装时解压到根目录下。如opt文件夹会安装在/opt 文件目录，usr 目录会安装在/usr 目录下。opt文件夹里存放软件的运行文件。usr 目录存放图标和桌面文件。

在usr 目录下有两个子文件夹:applications 和 icons。



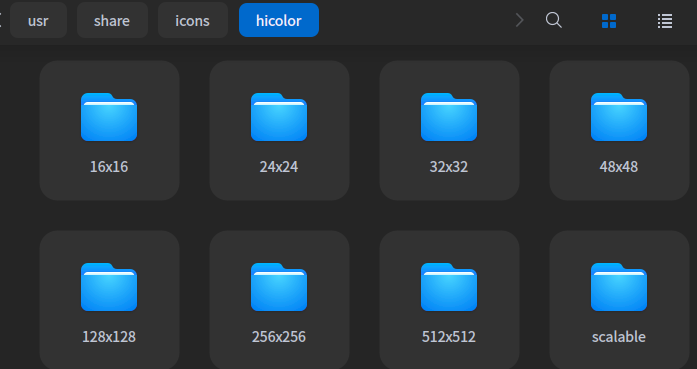
在applications 中存放linux 桌面文件，以文本形式打开applications 中的com.thsoft.thcadv23.desktop 桌面文件。



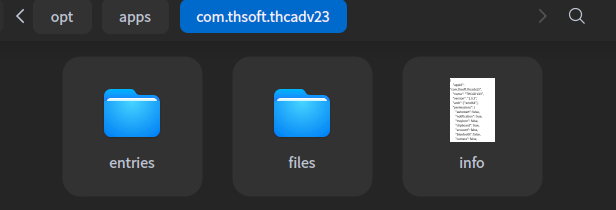
具体个字段意义参考： [Linux下Desktop文件入门](https://blog.csdn.net/YiferHuang/article/details/86634589)。需注意字段要统一规范。以下说明常用字段：

1. Name: app name
2. Exec: 点击快捷方式运行的命令，这里表示运行thcad.sh
3. Icon:桌面图标

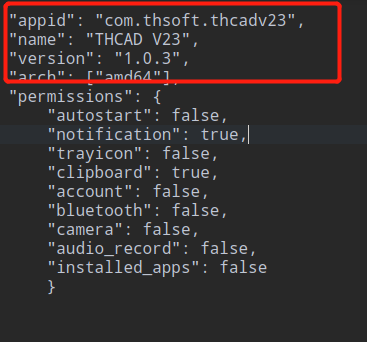
icons 存放图标各个尺寸的图标。



在/opt/apps/com.thsoft.thcadv23 中有两个文件夹和info文件。entries是安装包入口文件，里面的内容同usr/share 。有些系统如UOS会在安装时读取其中文件的信息而不是直接解压usr/share，所以在打包UOS版本时，删除usr 文件夹。



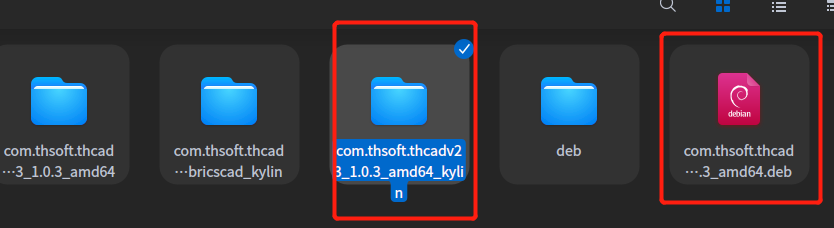
Info 文件保存权限信息。注意在更新时保持版本号一致。

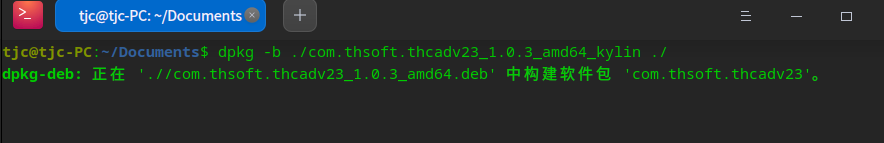


## 打包

修改完目录的文件使用如下命令打包：

dpkg -b ./com.thsoft.thcadv23\_1.0.3\_amd64\_kylin ./





命令运行完成后生成的deb安装包。

# 附录

## MFC -> Qt类对照表

|  |  |
| --- | --- |
| 数据类 |  |
| MFC | Qt |
| CString | QString |
| CMap | QMap |
| CList | QList |
| ... | ... |
|  |  |
| 窗口类 |  |
| CCheckBox | QCheckBox |
| CEdit | QLineEdit |
| CCombobox | QCombobox |
| CGroupBox | QGroupBox |
| CSliderCtrl | QSlider |
| CButton | QPushButton |
| CStatic | QLabel |
| CDialog | QDialog |

表 3-1

## Problem and Solution

symbol lookup error: undefined symbol: \_ZdlPvm, version Qt\_5

Reason: cannot find qt lib

Sln: 通过修改/etc/profile，添加环境变量，在profile文件最后一行添加如下代码：

export LD\_LIBRARY\_PATH=/opt/qt5/5.12.11/gcc\_64/lib

根据Qt的安装路径进行修改。

Failed to find "GL/gl.h" in "/usr/include/libdrm".

Sln: sudo apt install mesa-common-dev libglu1-mesa-dev

Cannot find Cmake , GNU compiler

Sln: sudo apt install build-essential

/lib/x86\_64-linux-gnu/libQt5Core.so.5: version `Qt\_5.12' not found

Reason: (找不到库)

Sln(同1): export LD\_LIBRARY\_PATH=/opt/qt5/5.12.11/gcc\_64/lib

UOS 安装Qt 请使用UOS 推荐办法

[开发者文档-UOS开发者平台 (chinauos.com)](https://developer.chinauos.com/" \l "/pages/document/info?id=42e5c0af-4b32-40f2-ac1a-2308bbdb78cf&admin=)

Prob: “stdafx.h” not found

Reason: linux is a case\_strict system.

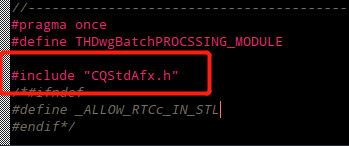
Sln: replace “stdafx” with “StdAfx.h”

Prob:Afxwin.h not found

Reason:

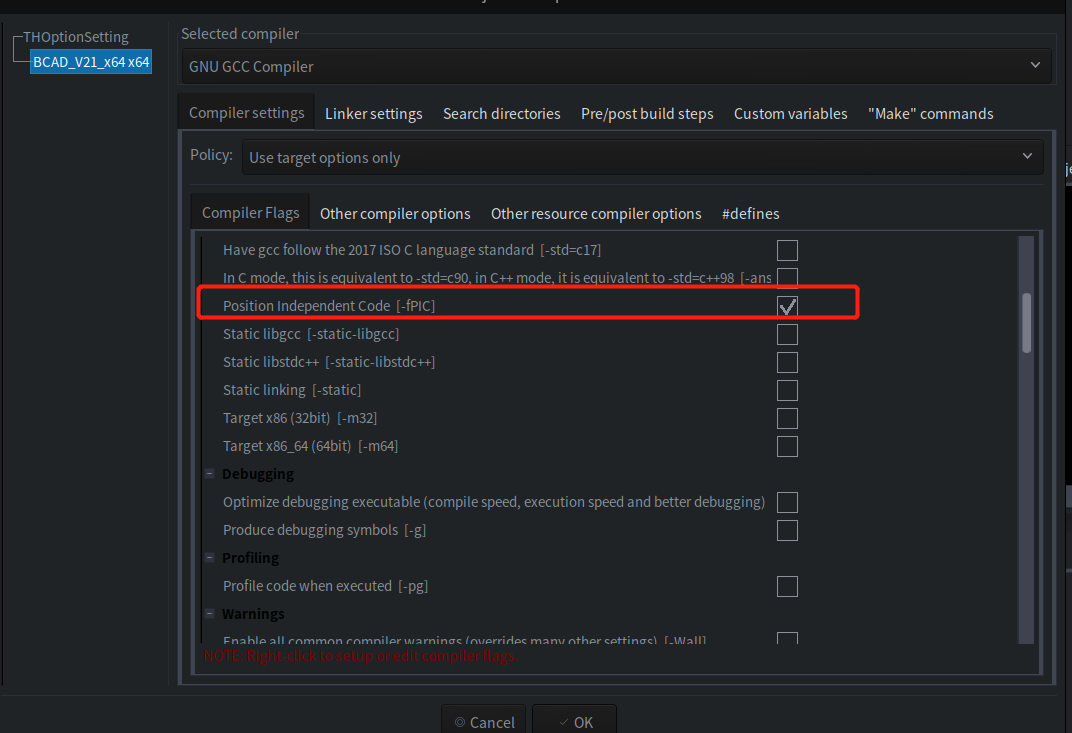
Linux have no winapi here,so you have to include a simulate winapi which locate at brx\_linux\_sdk.

Sln: Add #include “CQStdAfx.h” to the head of StdAfx.h.



You must build your code with position independent code if ...

Sln: 打开project build options, add “-fPIC” to other Compile option

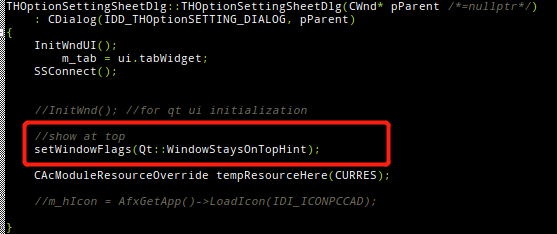


对话框没有显示在最前面，或者闪一下便显示在cad后面

Reason:Qt linux bug

Sln:在窗口类的构造函数中添加

setWindowFlags(Qt::WindowStaysOnTopHint);



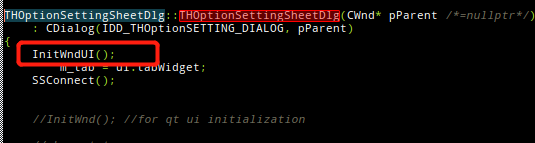
1. 对话框显示出来了，但是一片空白

Reason：界面没有初始化

Sln: 在在窗口类的构造函数的开始处添加如下代码：

InitWndUI();

同时要声明[DECLARE\_DYNAMIC 和IMPLEMENT\_DYNAMIC](#_DECLARE_DYNAMIC和IMPLEMENT_DYNAMIC) 宏



Undefind symbol : XXX

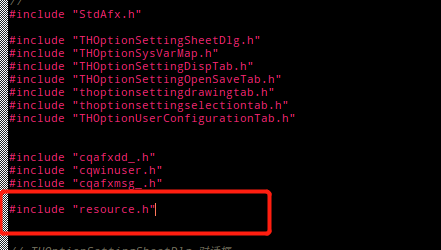
Reason:函数只定义但未实现或者窗口类头文件修改了而moc\_xxx.cpp 没有重新生成，记得修改完 CQClass窗口类的头文件后保存，再[重新生成moc\_xxx.cpp](#_MFC相关代码移植步骤).

Expected unqualified-id before numeric constant，报错跳转到resource.h中。

Reason:ui\_xxx.h文件中的控件变量名与resource.h 中定义的宏名冲突

Sln: 约定所有 .h头文件不包含 resource.h，而是包含在cpp文件中，且

#include “resource.h” 写在最后。



uos黑屏以及其他故障

Sln:使用livecd 修复。

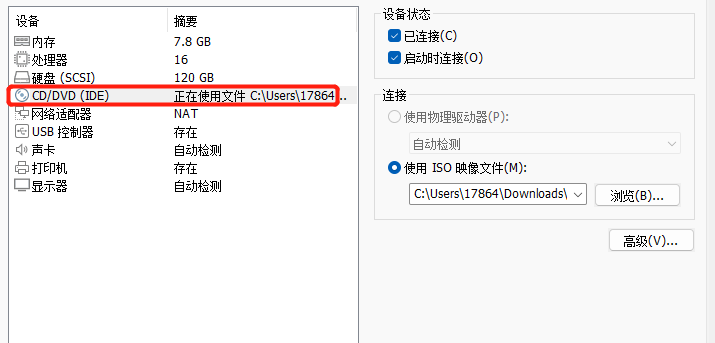
修复步骤：

1. 制作相应系统的安装启动盘

对于UOS详细步骤参考：[《统信UOS》如何制作UOS系统安装启动盘-攀升知识库 (ipason.com)](https://knowledge.ipason.com/ipKnowledge/knowledgedetail.html/1277)

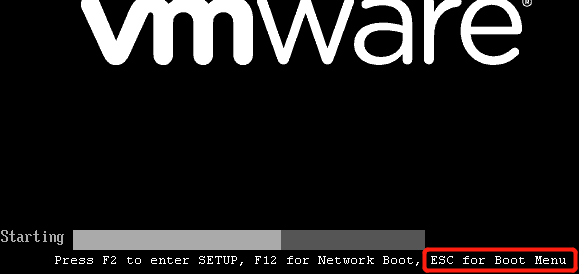
配置光驱

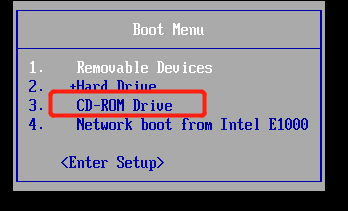
1. 虚拟机通过U盘载入需要配置光驱，打开需要修复的虚拟机硬件设置，选择使用ISO映像文件，浏览选择对应虚拟机的ios文件。



1. 通过u盘启动虚拟机

开机需要修复的虚拟机，在启动页面按esc(amd 系统)进入boot menu，选择 CD-ROM Drive ，enter 进入。

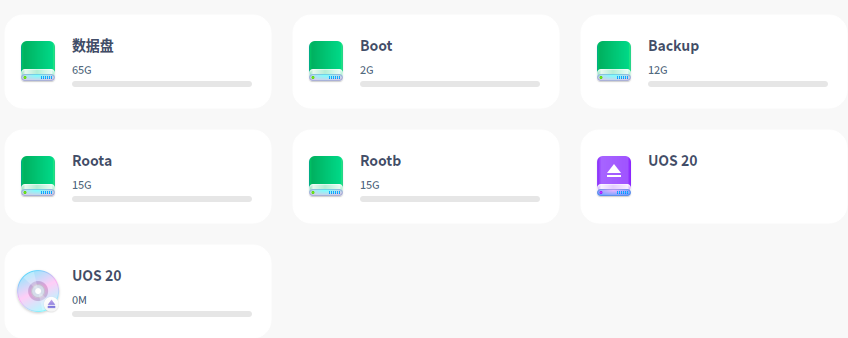




稍等片刻，在如下界面按Tab(进入编辑模式)，光标移动到.linux 开头那行，删除 livecd-installer 后enter进入([《统信UOS》如何进入livecd救援模式-攀升知识库 (ipason.com)](https://knowledge.ipason.com/ipKnowledge/knowledgedetail.html/1274))。



进入系统之后可以将故障之前的操作进行回退。或者将数据转移到u盘。



14.  
cad 命令行报错

qt.qpa.xcb: QXcbConnection: XCB error: 3

Reason:Qt linux 在uos系统存在bug，目前没有发现功能上的影响。

Sln:设置Qt调试信息为false，将如下代码添加到thcad.sh:

export QT\_LOGGING\_RULES="\*.debug=false;\*.info=false;\*.warning=false;\*.critical=false"

