**qt command line program to start external processes\_Qt's QProcess(1)Run cmd command**

I. Two ways of launching external programs:

(1)void QProcess::start(const QString & program, const QStringList & arguments, OpenMode mode = ReadWrite)

(2)void QProcess::startDetached(const QString & program, const QStringList & arguments, const QString & workingDirectory = QString(), qint64 \* pid = 0

For example:

process->start("F:/AppInst/115.exe");

process->start("C:/ HaoZip.exe");

process->start("C:/Program Files/HaoZip/HaoZip.exe", QStringList("C:/Program Files/HaoZip/HaoZip.exe"));

To start an external program, pass the path and run parameters of the external program. The parameters are brought in using a QStringList.

(1) Set the path: void QProcess::setProgram(const QString & program)

(2) Set the parameters [optional]: void QProcess:: setArguments(const QStringList & arguments)

(3) Initiation: void QProcess::start(OpenMode mode = ReadWrite)

Qt run cmd commandvoid MainWindow::on\_pushButton\_clicked()

{ui->textEdit->clear();

QProcess process(this);

process.setProgram("cmd");

QStringList argument;

argument<lineEdit->text();

process.setArguments(argument);

process.start();

process.waitForStarted();

process.waitForFinished();

QString temp=QString::fromLocal8Bit(process.readAllStandardOutput());

ui->textEdit->setText(temp);}

Process.start(“regedit.exe”)

**QT starts a process**

QT Start A process - an exe file method.

notepad.exe或cmd.exe

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QProcess>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{ Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_pushButton\_clicked();

void showResult();

void showState(QProcess::ProcessState);

void showError();

void showFinished(int,QProcess::ExitStatus);

private:

Ui::MainWindow \*ui;

QProcess my\_Process;

};

#endif // MAINWINDOW\_H

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QDebug>

#include <QTextCodec>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

connect(&my\_Process,&QProcess::readyRead,this,&MainWindow::showResult);

connect(&my\_Process,&QProcess::stateChanged,this,&MainWindow::showState);

connect(&my\_Process,&QProcess::errorOccurred,this,&MainWindow::showError);

connect(&my\_Process,SIGNAL(finished(int,QProcess::ExitStatus)),this,SLOT(showFinished(int,QProcess::ExitStatus)));

}

MainWindow::~MainWindow()

{ delete ui;

}

void MainWindow::on\_pushButton\_clicked()

{// my\_Process.start("notepad.exe");

QString program ="cmd.exe";

QStringList arguments;

arguments<<"/c dir&pause"; // / c Specify the command dir to be executed in cmd my\_Process.start(program,arguments);

}

void MainWindow::showState(QProcess::ProcessState state)

{ qDebug()<<"showState: ";

if(state == QProcess::NotRunning)

{

qDebug()<<"No program running ";

}

else if(state==QProcess::Starting)

{

qDebug()<<"In the beginning ";

}

else

{

qDebug()<<"In operation ";

}

}

void MainWindow::showResult()

{

QTextCodec \*codec=QTextCodec::codecForLocale();

qDebug()<<"Show results:"<<endl<<codec->toUnicode(my\_Process.readAll());

}

void MainWindow::showError()

{

qDebug()<<"Display of error messages："<<endl<<my\_Process.errorString();

}

void MainWindow::showFinished(int exitCode,QProcess::ExitStatus exit\_stattus)

{

qDebug()<<"Show completion："<<endl<<" Exit Code："<<exitCode<<"，Exit status:"<<exit\_stattus;

}

**Example of Qt using QProcess to start a process**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QProcess>

namespace Ui {class MainWindow;}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~MainWindow();

private:

Ui::MainWindow \*ui;

public slots:

void OnFinishProc(int, QProcess::ExitStatus);

};

#endif // MAINWINDOW\_H

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

QProcess \* proc = new QProcess(this);

QStringList lst;

lst<<QString::fromLocal8Bit("D:\\ Report.docx");

proc->start("\"C:\\Program Files\\Microsoft Office\\Office16\\WINWORD.EXE\"", lst);

connect(proc, SIGNAL(finished(int, QProcess::ExitStatus)), this, SLOT(OnFinishProc(int, QProcess::ExitStatus)));

}

MainWindow::~MainWindow()

{ delete ui;

}

void MainWindow::OnFinishProc(int, QProcess::ExitStatus)

{

QProcess \* proc = (QProcess \*)sender();

delete proc;

}

**QT Open External Programs for Windows & Mac**

QString qsPath = QCoreApplication::applicationDirPath();

#ifdef \_WIN32

qsPath += + "/user.exe";

qsPath.replace("/", "\\");

QProcess::startDetached(qsPath, QStringList());

#else // \_WIN32

qsPath += + "/user.app";

QProcess proc;

proc.start(qsPath, QStringList());

proc.waitForFinished(-1);

#endif

Open external programs with the bat command under windows

// Select a path to a location that can be written to

QString batFile = QStandardPaths::writableLocation(QStandardPaths::HomeLocation)+ QDir::separator() + "StartDaemon.bat";

// Delete old documents

if (QFile(batFile).exists())

{

QFile(batFile).remove();

}

// Write a new file

QFile file(batFile);

if (file.open(QFile::ReadWrite))

{

QString strParam = QString("@echo off \n start /d \"%1\" HiDaemond.exe \"%2\" \n exit").arg(QCoreApplication::applicationDirPath()).arg(QFileInfo(QCoreApplication::applicationFilePath()).fileName());

file.write(strParam.toLocal8Bit());

file.close();

// Execute the script

QProcess p;

p.start("cmd.exe", QStringList() << "/c" << batFile);

if (p.waitForStarted())

{

p.waitForFinished(2000);

return;

}

@echo off

start /d

exit

**QT opens external programs and embeds them in the Qt interface**

I Start-up procedure

By calling the system function system

system("e:\\1.exe");

***Second, through QProcess, blocking calls to QProcess::execute***

QProcess \*pro = new QProcess;

pro->start("/opt/myApp");

FindWindow句法

Retrieves the handle of the top-level window whose class name and window name match the specified string. This function does not search for sub-windows. This function does not perform a case-sensitive search.

Add inside the pro file：

win32{LIBS += -luser32}

Source document added:

#include <Windows.h>

HWND FindWindow（LPCTSTR IpClassName，LPCTSTR IpWindowName）;

IpClassName ：A pointer to a null-terminated string specifying the class name, or to a member identifying the class name string. Assuming this parameter is a member, it must be a global member generated by a previous call to theGlobafAddAtom function. The member is 16 bits and must be located in the lower 16 bits of IpClassName and the higher bits must be 0.

IpWindowName：Points to an empty closing string specifying the form name (form title). Assuming this parameter is empty, it is a full match for all forms.

HWND hWnd = ::FindWindow(L"CalcFrame", L" Calculators ");

Return value type: HWND

If the function succeeds, the return value is a window handle with the specified class name and window name. To get the extended error message, use GetLastError

QWindow\*m\_window;

m\_window= QWindow::fromWinId((WId)hWnd);

m\_widget = QWidget::createWindowContainer(m\_window, this->ui->widget); // The second parameter is used as the parent of window, or you can use layout

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QProcess>

#include <QDir>

#include "windows.h"

#include "stdio.h"

#include <QDebug>

#include <QMainWindow>

#include <QWindow>

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

// Launch

QDir::setCurrent("C:\\windows\\system32");

QProcess \*pro = new QProcess(this);

pro->start("notepad.exe");

ui->setupUi(this);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::integrate()

{

QString Name = "No title - Notepad ";

QString Name2 = "MainWindow";

WId hwnd = (WId)FindWindow(L"Notepad",(LPCTSTR)Name.unicode());

WId m\_hWnd = (WId)FindWindow(L"Qt5QWindowIcon",(LPCTSTR)Name2.unicode());

qDebug()<<hwnd;

qDebug()<<m\_hWnd;

if (hwnd > 0)

{

QWindow \*m\_window;

m\_window = QWindow::fromWinId(WId(hwnd));

QWidget \*m\_widget;

m\_widget = QWidget::createWindowContainer(m\_window,this);

setCentralWidget(m\_widget);

}

}

void MainWindow::on\_pushButton\_clicked()

{

integrate();

}

# QWidget to HWND interconversion

# HWND to QWidget

# QWidget \*myWidget;

HWND hwnd;

myWidget=QWidget::find(hwnd);

QWidget \*myWidget;

HWND hwnd;

hwnd=( HWND )myWidget->winId();

A type name preceded by an H is basically a handle in windows programming

Common handles .

HBITMAP Handle to the memory field where the bitmap information is stored

HBRUSH Handle to a brush

HCTR Handle to child window control

HCURSOR Handle to mouse cursor

HDC Device Description Table handle

HDLG Dialog box handle

HFONT Font handle

HICON Icon handle

HINSTANCE application instance handle

HMENU menu handle

HMODULE Module handle

HPALETTE Handle to colour palette

HPEN Pen handle

HWND Window handle