**qt命令行程序启动外部进程\_Qt之QProcess(一)运行cmd命令**

一、启动外部程序的两种方式：

(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

比如：

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"));

启动一个外部程序，须要传递外部程序的路径和运行參数。參数用QStringList来带入。

(1)设置路径： void QProcess::setProgram(const QString & program)

(2)设置參数[可选]： void QProcess:: setArguments(const QStringList & arguments)

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

Qt运行cmd命令

void 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启动 一个进程

QT启动 一个进程-一个exe文件方法。

启动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指定命令dir在cmd中执行

my\_Process.start(program,arguments);

}

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

{ qDebug()<<"showState: ";

if(state == QProcess::NotRunning)

{

qDebug()<<"没有程序运行";

}

else if(state==QProcess::Starting)

{

qDebug()<<"开始中";

}

else

{

qDebug()<<"运行中。。。";

}

}

void MainWindow::showResult()

{

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

qDebug()<<"显示结果:"<<endl<<codec->toUnicode(my\_Process.readAll());

}

void MainWindow::showError()

{

qDebug()<<"显示错误信息："<<endl<<my\_Process.errorString();

}

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

{

qDebug()<<"显示完成："<<endl<<"退出代码："<<exitCode<<"，退出状态:"<<exit\_stattus;

}

# Qt利用QProcess启动进程实例

#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:\\报告.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打开外部程序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

windows下bat命令打开外部程序

// 选取一个能写入位置的路径

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

// 删除旧的文件

if (QFile(batFile).exists())

{

QFile(batFile).remove();

}

// 写入新的文件

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();

// 执行脚本

QProcess p;

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

if (p.waitForStarted())

{

p.waitForFinished(2000);

return;

}

@echo off

start /d

exit

# QT打开外部程序并嵌入Qt界面

一 启动程序

通过调用系统函数 system

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

## 二、通过[QProcess](https://so.csdn.net/so/search?q=QProcess&spm=1001.2101.3001.7020),阻塞调用QProcess::execute

QProcess \*pro = new QProcess;

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

FindWindow句法

检索其类名和窗口名与指定字符串匹配的顶级窗口的句柄。此函数不搜索子窗口。此函数不执行区分大小写的搜索。

在pro文件里面添加：

win32{LIBS += -luser32}

源文件添加：

#include <Windows.h>

HWND FindWindow（LPCTSTR IpClassName，LPCTSTR IpWindowName）;

IpClassName ：指向一个指定了类名的空结束字符串，或一个标识类名字符串的成员的指针。假设该參数为一个成员，则它必须为前次调用theGlobafAddAtom函数产生的全局成员。该成员为16位，必须位于IpClassName的低 16位，高位必须为 0。

IpWindowName：指向一个指定了窗体名（窗体标题）的空结束字符串。假设该參数为空，则为全部窗体全匹配。

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

返回值类型：HWND

如果函数成功，则返回值是具有指定类名和窗口名的窗口句柄。如果函数失败，则返回值为NULL。要获取扩展的错误信息,使用GetLastError

QWindow\*m\_window;

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

m\_widget = QWidget::createWindowContainer(m\_window, this->ui->widget); //第二个参数是作为window的父类，也可以用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)

{

//启动

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 = "无标题 - 记事本";

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;

//嵌入

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与HWND的互相转换

## HWND转QWidget

QWidget \*myWidget;

HWND hwnd;

myWidget=QWidget::find(hwnd);

QWidget \*myWidget;

HWND hwnd;

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

在windows编程中类型名前面加H的基本是句柄

常用句柄 :

HBITMAP 保存位图信息的内存域的句柄

HBRUSH 画刷句柄

HCTR 子窗口控件句柄

HCURSOR 鼠标光标句柄

HDC 设备描述表句柄

HDLG 对话框句柄

HFONT 字体句柄

HICON 图标句柄

HINSTANCE 应用程序实例句柄

HMENU 菜单句柄

HMODULE 模块句柄

HPALETTE 颜色调色板句柄

HPEN 笔的句柄

HWND 窗口句柄