# Qt 之 qInstallMessageHandler（输出详细日志）

# 简述

安装之前已定义的消息处理程序，返回一个指向前一个消息处理程序。

消息处理程序是一个函数，用于打印调试信息、警告信息、严重错误和致命的错误的消息。Qt库（debug模式）包含成百上千的警告信息打印，当发生内部错误时（通常是无效的函数参数）。Qt在release模式下构建不包含这些警告，除非在编译时设置QT\_NO\_WARNING\_OUTPUT和/或QT\_NO\_DEBUG\_OUTPUT。如果你实现自己的消息处理程序，就可以完全控制这些消息。

缺省的消息处理程序向标准输出打印消息。如果是一个致命的消息，应用程序就会立即中止。

仅可以定义一个消息处理程序，因为这通常是在应用范围的基础上进行的，以控制调试输出。

调用qInstallMessageHandler(0)可以恢复消息处理程序。

# 调试级别

Qt包含用于警告和调试文本的全局宏：

* qDebug()   
  调试消息
* qInfo()   
  信息消息
* qWarning()   
  警告消息和可恢复的错误
* qCritical()   
  关键错误和系统错误
* qFatal()   
  致命错误

如果包含头文件<QDebug>，就可以将所述qDebug()宏用作一个输出流。

例如：

qDebug() << "Widget" << widget << "at position" << widget->pos();

Windows中，如果是一个控制台应用程序，文本将被发送到控制台；否则，被发送到调试器。

# 正常使用

一般情况下，在调试程序的时候，我们会输出一些调试信息，便于程序跟踪。

例如：

int main(int argc, char \*\*argv)

{

QApplication app(argc, argv);

// 打印信息

qDebug("This is a debug message.");

qWarning("This is a warning message.");

qCritical("This is a critical message.");

qFatal("This is a fatal message.");

...

return app.exec();

}

输出如下：

This is a debug message.   
This is a warning message.   
This is a critical message.   
This is a fatal message.

# 

# 自定义消息处理

一般情况下，一个大型项目会出现很多这样类似的调试信息，很不利于查找，因为我们根本不清楚输出信息所在的文件、函数、行号等信息。

下面，我们来根据需要自定义消息处理程序。

#include <QApplication>

#include <stdio.h>

#include <stdlib.h>

*// 自定义消息处理程序*

void myMessageOutput(QtMsgType type, const QMessageLogContext &context, const QString &msg)

{

QByteArray localMsg = msg.toLocal8Bit();

switch (type) {

case QtDebugMsg:

fprintf(stderr, "Debug: %s (%s:%u, %s)\n", localMsg.constData(), context.file, context.line, context.function);

break;

case QtInfoMsg:

fprintf(stderr, "Info: %s (%s:%u, %s)\n", localMsg.constData(), context.file, context.line, context.function);

break;

case QtWarningMsg:

fprintf(stderr, "Warning: %s (%s:%u, %s)\n", localMsg.constData(), context.file, context.line, context.function);

break;

case QtCriticalMsg:

fprintf(stderr, "Critical: %s (%s:%u, %s)\n", localMsg.constData(), context.file, context.line, context.function);

break;

case QtFatalMsg:

fprintf(stderr, "Fatal: %s (%s:%u, %s)\n", localMsg.constData(), context.file, context.line, context.function);

abort();

}

}

int main(int argc, char \*\*argv)

{

QApplication app(argc, argv);

*// 安装消息处理程序*

qInstallMessageHandler(myMessageOutput);

*// 打印信息*

qDebug("This is a debug message.");

qWarning("This is a warning message.");

qCritical("This is a critical message.");

qFatal("This is a fatal message.");

...

return app.exec();

}

这时，我们不仅可以输出调试信息，而且可以很直观、很方便的得到输出代码所在的文件、函数、行号等信息。

输出如下：

Debug: This is a debug message. (..\MessageOutput\main.cpp:90, int \_\_cdecl main(int,char \*[]))   
Warning: This is a warning message. (..\MessageOutput\main.cpp:91, int \_\_cdecl main(int,char \*[]))   
Critical: This is a critical message. (..\MessageOutput\main.cpp:92, int \_\_cdecl main(int,char \*[]))   
Fatal: This is a fatal message. (..\MessageOutput\main.cpp:93, int \_\_cdecl main(int,char \*[]))