

- 基本对话框
- 布局管理
- 对话框进阶
- QMainWindow
- 图形与图画
- Graphics View
- Model / View
- 网络与通信
- 磁盘文件
- 事件



→ 全书所有实例的

源代码

- 开源工具软件包
- Windows动态链接库
- Linux动态链接库

# Linux窗口程序设计

## —— Qt4精彩实例分析

成洁 卢紫毅 编著



清华大学出版社

# Linux 窗口程序设计

## ——Qt4 精彩实例分析

成 洁 卢紫毅 编著

清华大学出版社

北 京

## 内 容 简 介

Qt 作为 Linux 下 GUI 的强大编程工具,能给用户提供精美的图形界面所需要的所有元素,已经得到了越来越广泛的应用。本书共分 11 章,以循序渐进的方式对 Qt 应用开发进行了介绍,涵盖了界面外观、图像处理、磁盘文件、网络与通信、事件等程序设计中经常涉及的内容。作者希望通过本书为想学习 Qt 编程的读者提供入门的指导,也为从事 Qt 开发应用的读者提供帮助。

本书内容全面,针对性强,叙述言简意赅、清晰流畅,讲解透彻,通俗易懂,图例丰富,所有实例均在 Linux 和 Windows 操作系统下进行了验证。

本书适合于从事或准备从事 Qt 开发的技术人员,也可作为 Linux 窗口应用开发者的参考书。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

图书在版编目(CIP)数据

Linux 窗口程序设计——Qt4 精彩实例分析/成洁,卢紫毅编著. —北京:清华大学出版社,2008.11

ISBN 978-7-302-18158-3

I. L… II. ①成… ②卢… III. Linux 操作系统-程序设计 IV. TP316.89

中国版本图书馆 CIP 数据核字(2008)第 104638 号

责任编辑:涂 荣 张丽萍

封面设计:张 岩

版式设计:赵丽娜

责任校对:王 云

责任印制:王秀菊

出版发行:清华大学出版社

地 址:北京清华大学学研大厦 A 座

<http://www.tup.com.cn>

邮 编:100084

社 总 机:010-62770175

邮 购:010-62786544

投稿与读者服务:010-62776969, [c-service@tup.tsinghua.edu.cn](mailto:c-service@tup.tsinghua.edu.cn)

质 量 反 馈:010-62772015, [zhiliang@tup.tsinghua.edu.cn](mailto:zhiliang@tup.tsinghua.edu.cn)

印 刷 者:北京市清华园胶印厂

装 订 者:三河市李旗庄少明装订厂

经 销:全国新华书店

开 本:185×230 印 张:20.25 字 数:396 千字

(附光盘 1 张)

版 次:2008 年 11 月第 1 版

印 次:2008 年 11 月第 1 次印刷

印 数:1~5000

定 价:38.00 元

---

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题,请与清华大学出版社出版部联系调换。联系电话:(010)62770177 转 3103 产品编号:027649-01

# 前 言

Qt 作为 Linux 下图形用户界面的强大编程工具，能给用户提供了精美的图形界面所需要的所有元素，已经得到了越来越广泛的应用，并且当前多数高端嵌入式设备生产商都选择了 Qt 作为开发工具。

目前，市场上关于 Qt 编程的书籍与其他编程开发工具的参考资料相比，可谓少之又少，这与快速发展的 Qt 不相符合。现有的有关 Qt 的书籍，内容基本上都是面向 Qt 理论，涵盖的内容很多，对 Qt 的发展发挥着强有力的促进作用。本书主要以 Qt 编程实例为基点，将程序设计中经常使用的编程方法和技巧介绍给大家，针对性强，对于初学者来说是一本非常实用的参考书。作者希望通过本书为想学习 Qt 编程的读者提供入门的指导，也为从事 Qt 开发应用的读者提供帮助。

对于应用程序开发者而言，最快捷的学习方式就是对具体实例的分析，本书的编写正是基于这一理念，为从事或准备从事 Qt 开发应用的人员提供一种便捷的学习途径。本书包含 63 个实例，涵盖了界面外观、图像处理、磁盘文件、网络与通信、事件等程序设计中经常涉及的方面，所选实例都是作者结合平时研发工作中的经验精心挑选出来的，内容丰富，具有很强的针对性和实用性。

本书共分 11 章，以循序渐进的方式对 Qt 应用开发进行了介绍，包括：

- 第 1 章 基本对话框
- 第 2 章 布局管理
- 第 3 章 对话框进阶
- 第 4 章 QMainWindow
- 第 5 章 图形与图画
- 第 6 章 Graphics View
- 第 7 章 Model/View
- 第 8 章 网络与通信
- 第 9 章 磁盘文件
- 第 10 章 事件
- 第 11 章 其他

本书结构没有按照 Qt 类的划分进行介绍，因为 Qt 的在线帮助已经提供了详尽的参



考，而是通过具体实例对涉及的 Qt 类进行了介绍和分析，突出实用性。本书是面向 Qt 程序开发人员的一本入门级参考书，面向实际应用开发，旨在为 Qt 开发人员提供直观的实例参考，针对性强，对 Qt 开发人员具有很好的参考价值。

本书读者对象为从事或准备从事 Qt 开发应用的人员，阅读本书的读者需具备 C++ 面向对象编程方面的知识。

由于时间仓促，作者水平有限，本书无法覆盖 Qt 的全部内容，难免存在缺点和不足之处，敬请读者批评指正。

本书在写作的过程中得到了清华大学出版社的大力支持与帮助，在此，作者向其表示诚挚的谢意！

作 者

2008 年 3 月

# 目 录

|                             |    |
|-----------------------------|----|
| 第 1 章 基本对话框                 | 1  |
| 实例 1 Hello World!           | 2  |
| 实例 2 标准对话框的使用               | 5  |
| 实例 3 各类位置信息                 | 10 |
| 实例 4 使用标准输入框                | 15 |
| 实例 5 各种消息框的使用               | 20 |
| 实例 6 实现 QQ 抽屉效果             | 27 |
| 实例 7 表格的使用                  | 30 |
| 实例 8 使用进度条                  | 31 |
| 实例 9 利用 Qt Designer 设计一个对话框 | 35 |
| 实例 10 在程序中使用 ui             | 40 |
| 实例 11 动态加载 ui               | 43 |
| 第 2 章 布局管理                  | 45 |
| 实例 12 基本布局管理                | 46 |
| 实例 13 多文档                   | 52 |
| 实例 14 分割窗口                  | 56 |
| 实例 15 停靠窗口                  | 58 |
| 实例 16 堆栈窗体                  | 61 |
| 实例 17 综合布局实例                | 63 |
| 第 3 章 对话框进阶                 | 69 |
| 实例 18 可扩展对话框                | 70 |
| 实例 19 利用 QPalette 改变控件颜色    | 73 |
| 实例 20 窗体的淡入淡出效果             | 79 |
| 实例 21 不规则窗体                 | 84 |
| 实例 22 电子钟                   | 87 |
| 实例 23 程序启动画面                | 92 |



|                            |     |
|----------------------------|-----|
| 第 4 章 QMainWindow          | 95  |
| 实例 24 基本 QMainWindow 主窗口程序 | 96  |
| 实例 25 打印文本                 | 103 |
| 实例 26 打印图像                 | 106 |
| 实例 27 图片的缩放与旋转             | 108 |
| 实例 28 在工具栏中嵌入控件            | 112 |
| 实例 29 设置字体、字号等格式属性         | 115 |
| 实例 30 设置文本排序及对齐            | 122 |
| 第 5 章 图形与图画                | 130 |
| 实例 31 利用 QPainter 绘制各种图形   | 132 |
| 实例 32 利用 QPainterPath 进行画图 | 143 |
| 实例 33 渐变效果                 | 149 |
| 实例 34 QPainter 坐标系的变形      | 159 |
| 实例 35 SVG 格式图片的显示          | 162 |
| 实例 36 一个简单的绘图工具            | 169 |
| 实例 37 改变图片的透明度             | 177 |
| 实例 38 橡皮筋线                 | 182 |
| 第 6 章 Graphics View        | 185 |
| 实例 39 地图浏览器                | 187 |
| 实例 40 各种 Graphics Item     | 193 |
| 实例 41 Graphics Item 的各种变形  | 202 |
| 实例 42 飞舞的蝴蝶                | 208 |
| 第 7 章 Model/View           | 212 |
| 实例 43 文件目录浏览器              | 215 |
| 实例 44 利用特定控件进行表项编辑         | 217 |
| 实例 45 自定义 Model            | 222 |
| 实例 46 柱状统计图                | 230 |
| 第 8 章 网络与通信                | 241 |
| 实例 47 获取本机网络信息             | 242 |
| 实例 48 基于 UDP 的网络广播程序       | 244 |



|                                    |            |
|------------------------------------|------------|
| 实例 49 基于 TCP 的网络聊天室程序 .....        | 247        |
| 实例 50 实现 HTTP 文件下载 .....           | 256        |
| 实例 51 实现 FTP 上传和下载 .....           | 261        |
| <b>第 9 章 磁盘文件 .....</b>            | <b>266</b> |
| 实例 52 获取文件属性 .....                 | 267        |
| 实例 53 文件浏览器 .....                  | 269        |
| <b>第 10 章 事件 .....</b>             | <b>273</b> |
| 实例 54 获得鼠标事件 .....                 | 274        |
| 实例 55 使用键盘控制移动 .....               | 276        |
| 实例 56 事件过滤器实现动态图片按钮 .....          | 281        |
| <b>第 11 章 其他 .....</b>             | <b>284</b> |
| 实例 57 利用 QSettings 保存程序窗口状态 .....  | 285        |
| 实例 58 利用 QDataStream 对文件进行存取 ..... | 290        |
| 实例 59 改变鼠标指针形状 .....               | 293        |
| 实例 60 改变窗体显示风格 .....               | 295        |
| 实例 61 拖拽图标 .....                   | 299        |
| 实例 62 拖拽文字 .....                   | 307        |
| 实例 63 字符串编码格式转换 .....              | 313        |

# 第 1 章 基本对话框

本章的实例对 Qt 编程的基本流程、标准对话框的使用方法以及 Qt Designer 的使用方法等进行了分析, 包括 11 个实例:

- 📁 Hello World!
- 📁 标准对话框的使用
- 📁 各类位置信息
- 📁 使用标准输入框
- 📁 各种消息框的使用
- 📁 实现 QQ 抽屉效果
- 📁 表格的使用
- 📁 使用进度条
- 📁 利用 Qt Designer 设计一个对话框
- 📁 在程序中使用 ui
- 📁 动态加载 ui



## 实例 1 Hello World!

知识点:

- 开发 Qt 程序的基本流程和编译运行方式
- 信号和槽机制 (Signal&Slot)

本实例实现一个“Hello World!”例子, 简单介绍 Qt 编程的基本流程, 以及 Qt 程序的编译运行方式。实例效果图如图 1-1 所示。



图 1-1 Hello World!

这是一个简单的例子, 整个对话框只有一个按钮, 单击该按钮, 对话框关闭, 退出程序。

实现代码如下:

```
1  #include <QApplication>
2  #include <QPushButton>
3  int main(int argc, char *argv[])
4  {
5      QApplication app(argc,argv);
6      QPushButton b("Hello World !");
7      b.show();
8      QObject::connect(&b,SIGNAL(clicked()),&app,SLOT(quit()));
9      return app.exec();
10 }
```

第 1 行包括<QApplication>, 所有 Qt 图形化应用程序都必须包含此文件, 它包含了 Qt 图形化应用程序的各种资源、基本设置、控制流以及事件处理等, 若是 Qt 的非图形化应用程序, 则需包含<QCoreApplication>。

 小贴士: Qt 最初的框架只有关于图形化应用的类, 随着它的一步步发展, Qt 已独立发展出了许多非图形化的类库, 如数据库应用、XML 解析等。

第2行包含了程序中要应用到的按钮控件的头文件。

 小贴士：在 Qt4 中，头文件的包含可以采用类似 `<QApplication>` 和 `<QPushButton>` 的形式，也可以写成 `<qapplication.h>` 和 `<qpushbutton.h>` 的形式。

第3行为应用程序的入口，所有 Qt 程序都必须有一个 `main()` 函数，以 `argc` 和 `argv` 作为入口参数。

第4行新创建了一个 `QApplication` 对象，每个 Qt 应用程序都必须有且只有一个 `QApplication` 对象，采用 `argc`、`argv` 作为参数，便于程序处理命令行参数。

第5行创建了一个 `QPushButton` 对象，并设置它的显示文本为“Hello World!”，由于此处并没有指定按钮的父窗体，因此以自己作为主窗口。

第6行调用 `show()` 方法，显示此按钮。控件被创建时，默认是不显示的，必须调用 `show()` 函数来显示它。

第7行的 `QObject::connect()` 方法是 Qt 最重要的特征，即信号与槽的机制。当按钮被按下则触发 `clicked()` 信号发射，与之相连的 `QApplication` 对象的槽 `quit()` 响应按钮信号，执行退出应用程序的操作。关于信号与槽机制在本实例最后将进行详细的分析。

最后调用 `QApplication` 的 `exec()` 方法，程序进入消息循环，等待可能输入进行响应。Qt 完成事件处理及显示的工作，并在应用程序退出时返回 `exec()` 的值。

Qt 程序的编译运行很简单，利用 Qt 提供的 `qmake` 工具能够很方便地对程序进行编译，编译流程如下：

```
qmake -project
qmake
make
./hello
```

其中，`qmake -project` 命令用于生成程序的项目文件（\*.pro）；`qmake` 用于生成程序的 Makefile 文件；`make` 编译 Makefile 文件得到可执行文件；最后执行程序即可出现图 1-1 所示的对话框，一个简单的 Hello World! 例子完成。

 小贴士：确保 Qt 的环境变量路径设置正确，程序编译时若出现连接错误，请首先检查有关 Qt 的环境变量，保证调用的 `qmake` 为相应的 Qt 版本。

信号与槽机制（signal&slot）作为 Qt 最重要的特性，提供了任意两个 Qt 对象之间的通信机制。其中，信号会在某个特定情况或动作下被触发，槽是用于接收并处理信号的函数。例如，要将一个窗口中的变化情况通知给另一个窗口，则一个窗口发送信号，另一个窗口的槽接收此信号并进行相应的操作，即可实现两个窗口之间的通信。这比传统



的图形化程序采用回调函数的方式实现对象间通信要简单灵活得多。每个 Qt 对象都包含预定的信号和槽，当某一特定事件发生时，一个信号被发射，与信号相关联的槽则会响应信号完成相应的处理。

信号与槽机制常用的连接方式为：

```
connect( Object1, SIGNAL(signal), Object2, SLOT(slot) );
```

signal 为对象 Object1 的信号，slot 为对象 Object2 的槽，Qt 的窗口部件都包含若干个预定义的信号和若干个预定义的槽。当一个类被继承时，该类的信号和槽也同时被继承。开发人员也可以根据需要进行定义自己的信号和槽。

信号与槽机制可以有多种连接方式，图 1-2 中描述了信号与槽的多种可能连接方式。

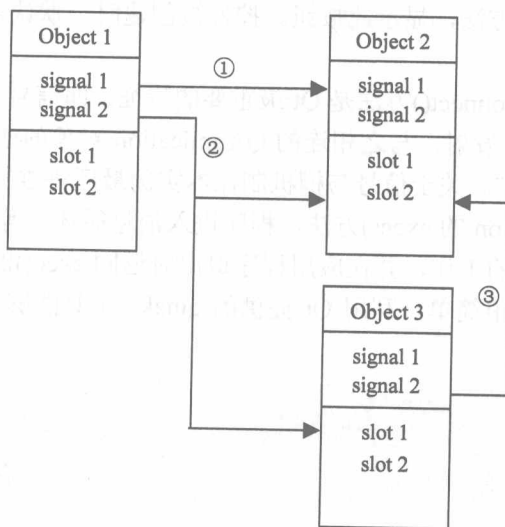


图 1-2 信号与槽的连接方式

① 一个信号可以与另一个信号相连。

```
connect(Object1, SIGNAL(signal 1), Object2, SIGNAL(signal 1));
```

即表示 Object1 的信号 1 发射可以触发 Object2 的信号 1 发射。

② 表示同一个信号可以与多个槽相连。

```
connect(Object1, SIGNAL(signal 2), Object2, SIGNAL(slot 2));  
connect(Object1, SIGNAL(signal 2), Object3, SIGNAL(slot 1));
```

③ 表示同一个槽可以响应多个信号。

```
connect(Object1, SIGNAL(signal 2), Object2, SIGNAL(slot 2));
connect(Object3, SIGNAL(signal 2), Object2, SIGNAL(slot 2));
```

## 实例2 标准对话框的使用

知识点:

- 📁 标准通用文件对话框的使用
- 📁 标准通用颜色对话框的使用
- 📁 标准通用字体对话框的使用

和大多数操作系统一样, Linux 也提供了一系列的标准对话框, 如文件选择、字体选择、颜色选择等, 这些标准对话框为应用程序提供了一致的观感。Qt 对这些标准对话框都定义了相关的类, 这些类让使用者能够很方便地使用标准对话框进行文件、颜色以及字体的选择。标准对话框在软件设计过程中是经常需要使用的。

 小贴士: Qt 提供的标准对话框除了本实例提到的, 还有 `QErrorMessage`、`QInputDialog`、`QMessageBox`、`QPrintDialog`、`QProgressDialog` 等, 这些标准对话框的使用在本书的后续部分将会陆续介绍。

本实例主要演示上面几种标准对话框的使用, 如图 1-3 所示。



图 1-3 标准对话框

在图 1-3 中, 单击“文件对话框”按钮, 会弹出文件选择对话框, 选中的文件名将显示在右边; 单击“颜色对话框”按钮, 会弹出颜色选择对话框, 选中的颜色将显示在右边; 单击“字体对话框”按钮, 会弹出字体选择对话框, 选中的字体将更新右边显示的字符串。

具体实现代码如下所示。

头文件 `standarddialogs.h`:

```
class StandardDialogs : public QDialog
```



```
{
    Q_OBJECT
public:
    StandardDialogs( QWidget *parent=0, Qt::WindowFlags f=0 );
    ~StandardDialogs();
public:
    QGridLayout *layout;
    QPushButton *filePushButton;
    QPushButton *colorPushButton;
    QPushButton *fontPushButton;
    QLineEdit *fileLineEdit;
    QLineEdit *fontLineEdit;
    QFrame *colorFrame;
private slots:
    void slotOpenFileDialog();
    void slotOpenColorDlg();
    void slotOpenFontDlg();
};
```

头文件定义了实例中需要用到的各种窗体控件以及各种操作的槽函数。

实现文件 standarddialogs.cpp:

```
StandardDialogs::StandardDialogs( QWidget *parent, Qt::WindowFlags f )
    : QDialog( parent, f )
{
1   setWindowTitle(tr("Standard Dialogs"));
2   layout = new QGridLayout( this );
3   filePushButton = new QPushButton;
4   filePushButton->setText(tr("File Dialog"));
5   colorPushButton = new QPushButton;
6   colorPushButton->setText(tr("Color Dialog"));
7   fontPushButton = new QPushButton;
8   fontPushButton->setText(tr("Font Dialog"));
9   fileLineEdit = new QLineEdit;
10  colorFrame = new QFrame;
11  colorFrame->setFrameShape( QFrame::Box );
12  colorFrame->setAutoFillBackground(true);
```

```

13  fontLineEdit = new QLineEdit;
14  fontLineEdit->setText(tr("Hello World"));

15  layout->addWidget( filePushButton, 0, 0 );
16  layout->addWidget( fileLineEdit, 0, 1 );
17  layout->addWidget( colorPushButton, 1, 0 );
18  layout->addWidget( colorFrame, 1, 1 );
19  layout->addWidget( fontPushButton, 2, 0 );
20  layout->addWidget( fontLineEdit, 2, 1 );
21  layout->setMargin(15);
22  layout->setSpacing(10);

23  connect(filePushButton,SIGNAL(clicked()),this,SLOT(slotOpenFileDialog()));
24  connect(colorPushButton,SIGNAL(clicked()),this,SLOT(slotOpenColorDlg()));
25  connect(fontPushButton,SIGNAL(clicked()),this,SLOT(slotOpenFontDlg()));
}

```

第 1 行设置主窗体的标题。

第 2~8 行分别创建各个按钮控件。

第 9 行创建一个 QLineEdit 类实例 fileLineEdit，用来显示选择的文件名。

第 10~12 行创建一个 QFrame 类实例 colorFrame，当用户选择不同的颜色时，colorFrame 会根据用户选择的颜色更新其背景。

第 13、14 行创建一个 QLineEdit 类实例 fontLineEdit，当用户选择不同的字体时，fontLineEdit 会根据用户选择的字体更新其内容。

第 15~22 行将各个控件进行布局。

第 23~25 行将各个按钮的 clicked 事件与相应的槽进行连接。

```

void StandardDialogs::slotOpenFileDialog()
{
    QString s = QFileDialog::getOpenFileName(
        this,
        "open file dialog",
        "/",
        "C++ files (*.cpp);;C files (*.c);;Head files (*.h)");

    lineEditFile->setText( s.toAscii() );
}

```

getOpenFileName()是 QFileDialog 类的一个静态函数，返回用户选择的文件名，如果用户选择取消（Cancel），则返回一个空串。函数形式如下：

```
QString QFileDialog::getOpenFileName ( QWidget * parent = 0, const QString & caption = QString(),
const QString & dir = QString(), const QString & filter = QString(), QString * selectedFilter = 0, Options
options = 0 )
```

调用 `getOpenFileName()` 函数将创建一个模态的文件对话框，如图 1-4 所示。dir 参数指定了默认的目录，如果 dir 参数带有文件名，则该文件将是默认选中的文件；filter 参数对文件类型进行过滤，只有与过滤器匹配的文件类型才显示，filter 可以同时指定多种过滤方式供用户选择，多种过滤器之间用“::”隔开，用户选择的过滤器通过参数 `selectedFilter` 返回。

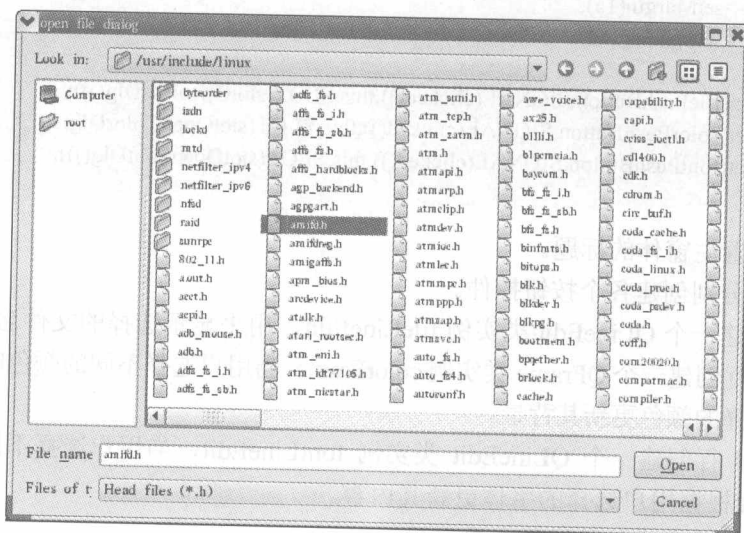


图 1-4 标准文件对话框

文件对话框的文件类型选择如图 1-5 所示。

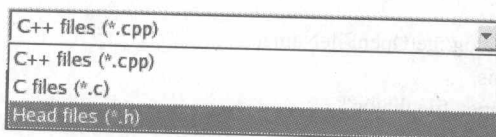


图 1-5 文件对话框的文件类型选择

`QFileDialog` 类还提供了类似的其他静态函数，如表 1-1 所示，通过这些函数，用户可以很方便地定制自己的文件对话框。

表 1-1 QFileDialog 的静态函数

| 静态函数                 | 说明             |
|----------------------|----------------|
| getOpenFileName      | 获得用户选择的文件名     |
| getSaveFileName      | 获得用户保存的文件名     |
| getExistingDirectory | 获得用户选择的已存在的目录名 |
| getOpenFileNames     | 获得用户选择的文件名列表   |

```

void StandardDialogs::slotOpenColorDlg()
{
    QColor color = QColorDialog::getColor (Qt::blue);

    if(color.isValid())
    {
        frameColor->setPalette(QPalette(color));
    }
}

```

getColor()是 QColorDialog 的一个静态函数，返回用户选择的颜色值，函数形式如下：

```
QColor getColor ( const QColor & initial = Qt::white, QWidget * parent = 0 )
```

调用 getColor()函数将创建一个模态的颜色对话框，如图 1-6 所示。initial 参数指定了默认选中的颜色，默认为白色。通过 QColor::isValid()可以判断用户选择颜色是否有效，若用户选择取消（Cancel），QColor::isValid()将返回 false。

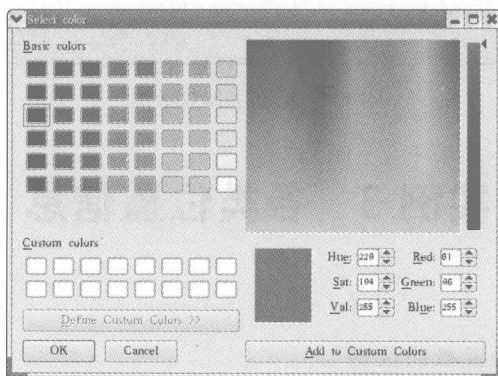


图 1-6 标准颜色对话框

```
void StandardDialogs::slotOpenFontDlg()
```