Qt 程序设计

Xteam(中国)软件技术有限公司　编著

清华大学出版社

(京) 新登字 158 号

内 容 简 介

Qt 是 Linux 系统中最流行的开发工具之一，它是开发 X Window 系统下应用程序的利器。本书全面地介绍了 Troll Tech 公司基于 C++的 CUI 开发工具——Qt。不仅介绍了如何编写 X Window 系统下的应用程序，还详细描述了 Qt 的开发环境、编程方法、关键技术和编程实例。最后，介绍了如何应用 KDE 本身提供的接口进行 X Window 系统编程。本书由中国第一家专业化的 Linux 发行商(中国)软件技术有限公司编写，适合初、中级程序员以及广大计算机编程的爱好者阅读。

书　　名：Qt 程序设计
作　　者：Xteam(中国)软件技术有限公司　编著
出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)
http://www.tup.tsinghua.edu.cn
责任编辑：张秋香
印 刷 者：北京大中印刷厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　**印张**：20.75　**字数**：492 千字
版　　次：2002 年 3 月第 1 版　2002 年 3 月第 1 次印刷
书　　号：ISBN 7-302-05231-X/TP・3076
印　　数：0001～4000
定　　价：30.00 元

前　言

本书内容

Qt 是 Linux 下的一种正在不断成熟，并且功能十分强大、灵活的开发工具。本书将介绍如何使用 Qt 在 X Window 上创建应用程序。通过对本书的学习，读者可以得到很大的收获。

读者对象

本书面向初、中级程序员及广大计算机编程的爱好者。读者应该了解 C++语言，但并不需要有任何在 X Window 下编程的背景和经验。通过本书的学习，读者将对 Qt 编程有一定的了解，并能够独立开发 X Window 的应用程序。

本书结构

本书分为 8 章：

- 第 1 章是“X Window 编程入门”，介绍 X Window 的一些基本的概念，并对 X Window 编程的技术进行必要的阐述。
- 第 2 章是“Qt 基础”，介绍 Qt 的系统概念，以及 Qt 与 X11，KDE 和 Motif 之间的关系。
- 第 3 章是“Qt 开发环境”，介绍了 Qt 开发环境的建立，以及如何开始编写 Qt 程序。
- 第 4 章是“Qt 编程方法”，以例程结合分析的方式对 Qt 编程的具体内容加以阐述。
- 第 5 章是“Qt 关键技术”，介绍 MOC，Signal 和 Slot 以及 drag 和 drop 等内容。
- 第 6 章是“Qt 类”，介绍 Qt 中类的继承结构及 Qt 中的一些重要和常用的类，并对它们的用处和几乎每一个成员函数都做了详尽的描述。
- 第 7 章是“编程实例”，通过一组实例程序的详解来由浅入深地介绍 Qt 编程方法。
- 第 8 章是“KDE 程序设计介绍”，介绍如何使用 KDE 本身提供的接口进行 X Window 编程。

由于时间紧迫，加之作者水平有限，书中的错误在所难免，敬请读者批评指正。

目　　录

第1章　X Window 编程入门

X Window 编程是 Qt 编程的基础，了解 X Window 编程的技术对理解 Qt 编程有很大的帮助。本章将对 X Window 的一些基本概念作详细的介绍，并对 X Window 编程的技术进行必要的阐述。

本章内容主要包括：

- X11 编程
- X Window 系统
- X11 与 Motif

1.1　X11 编程

随着 Xt，Motif 和 Qt 等高级 GUI 工具包的出现，用 Xlib 编写复杂的程序显得没有必要。但是一些基本的例程有时还会用到，并且所有工具包都是建立在 Xlib 基础之上。这里将会给出一个简单的模板和基本结构来介绍如何使用 Xlib 编一个 X Windows 程序。

下面给出一个例程，来逐步介绍 X Window 的编程方法。

1.1.1　头文件

大部分 X 程序都需要几个包含文件，比如<X11/Xlib.h>，<X11/Xutil.h>和<X11/Xos.h> 。其中，Xlib.h 包含了 Xlib 函数的结构定义。在<X11/Xlib.h>中又包含了一些其他的包含文件，如<X11/X.h>等；<X11/Xutil.h>中包含了一些 Xlib 函数的结构定义和常量；<X11/Xos.h>通过包含编译过程中依赖操作系统的一些文件，使 X 程序具有可移植性。除此之外，程序还应该根据情况增加对应的头文件。如<stdio.h>—— 标准 C 语言的输入输出库文件，<X11/Xresource.h>—— 包含 X 程序的一些资源定义结构和变量。

1.1.2　变量

在 X 程序中，除了一些 C 或 C++的标准变量外，还有 X 程序特有的变量，这些变

量一般在<X11/Xlib.h>中定义，比如：

● Display 它是一个结构，包含有关 X 服务器与屏幕的信息，当应用程序与 X 服务器相连时，才给该结构赋值。变量声明如 Display *display。

● Screen 显示屏幕信息，变量声明如 int screen。

● Window 用来标识一个 X 窗口，由 XCreateWindow()或 XCreateSimpleWindow()函数产生并返回。一个应用程序仅有一个窗口，如 Window win。

● XEvent 是一个存储有关事件信息的集合，根据事件的类型，可归为某一类结构中的一个，如 XButtonEvent 变量声明和 XEvent event。

● GC 是一种数据结构，代表图形上下文标识符，其中包含了一些画图的信息。变量声明如 GC gc1，gc2。

● XGCValues 表示图形上下文设置变量，如 XGCValues gs1，gs2。

1.1.3 连接服务器

在应用程序中首先要做的是将程序与 X 服务器相连，这意味着将显示器初始化，调用 XOpenDisplay()函数将完成这个工作。同样也可以使用默认的屏幕作为程序中的屏幕。XOpenDisplay()的用法如下：

```
Display XOpenDisplay(name)
char   * name
```

其中，name 为要连接的服务器的显示器名，形式如 host：server.screen，它可以是网络上任意的服务器。如果设置为 NULL，XOpenDisplay 将自动采用用户的 UNIX 环境变量 DISPLAY 来定义。通过 echo $ DISPLAY 命令就可以了解当前 DISPLAY 的内容。如果想改变默认的 DISPLAY 内容，可利用 setenv DISPLAY=name 等 Shell 命令。如果程序中 DISPLAY 内容为空，则在程序中提示错误信息，否则连接好服务器后设置屏幕信息如下：

```
if ((display = XOpenDisplay(NULL)) == NULL) {
    perror("Can't connect to server");
    exit(1);}
 screen = DefaultScreen(display);
```

1.1.4 窗口

在 X Window 程序中，创建窗口的基本的函数有两个：XCreateWindow()，

XCreateSimpleWindow()，后者的参数较前者的少，并且较为方便。可以利用XchangeWindowAttributes()函数来修改窗口属性。XCreateWindow()的用法如下：

```
Window   XCreateWindow(display,parent,x,y,width,height,border_width,
                                Depth,class,visual,valuemask,attributes)
        Display   * display;
        Window    parent;
        int   x,y,depth;
        unsigned   int   width,height,border_width,class;
        visual    * visual;
        unsighed   long   valuemask;
        XsetWindowAttributes    *attributes;
```

其中，display 是程序连接的显示；parent 为窗口的父窗口，程序最上级窗口的父窗口应是系统的父窗口；class 代表创建窗口的类，可以是输入输出型、只输入型，或从父窗口继承其类别；XSetWindowAttributes 中定义了窗口需设置的各种特性，比如背景、显示的光标等多种参数。利用上述函数创建窗口，用户需要定义很多参数，如XSetWindowAttributes 中的属性，以及 XCreateWindow()函数中的参数设置。

如果采用 XCreateSimpleWindow()函数创建窗口则相对简单些，其用法如下：

```
Window   XCreateSimpleWindow(display,parent,x,y,width,height,
                                border_width,border,background)
        Display    *display;
        Window  parent;
        int    x,y;
        unsigned   int   width,height,border_width;
        unsigned   long   border,background;
```

其中，(*x*，*y*)是窗口在父窗口中的位置坐标；width，height 和 border_width 分别是窗口的宽、高和边框宽度；border 和 background 为窗口的边框和背景颜色。例如，创建一个叫 win 的窗口，并利用 XMapWindow() 映射为可以显示出来的窗口即显示窗口。

```
win = XCreateSimpleWindow(display, RootWindow(display, screen),
100, 200, 400, 300, 1, BlackPixel(display, screen),
WhitePixel(display, screen));
/*映射为可以显示出来的窗口*/
XMapWindow(display, win);
```

1.1.5 位图

在 X 程序中将会用到很多位图或图标，此时，可调用 XCreateBitmapFromData()函数来创建 X 窗口中的 Pixmap 类型的数据。它的调用格式为：

```
Pixmap stipple;
    stipple = XCreateBitmapFromData(display, RootWindow(display, screen),
     stipple_bits, stipple_width, stipple_height));
```

例如，创建一个填充椭圆的位图。

```
Pixmap stipple;
if ((stipple = XCreateBitmapFromData(display, RootWindow(display, screen),
              stipple_bits, stipple_width, stipple_height)) == 0) {
            perror("Can't create bitmap");
            exit(1);
  }
```

其中，stipple_bits，stipple_width 和 stipple_height 分别是定义好的位图表和位图尺寸常量，定义如下：

```
#define stipple_width 3
#define stipple_height 3
      static char stipple_bits[] = {0x02, 0x05, 0x02};
```

1.1.6 事件

X 系统中，事件是一个基本的概念，表现为用户对键盘或鼠标的操作、窗口管理器对程序窗口的管理。事件可分为 3 种：暴露事件、改变尺寸事件和用户操作事件。事件的处理过程一般有以下几个步骤：

(1) 等待事件

(2) 读取事件

(3) 检查事件类型

(4) 处理事件

(5) 判断是否退出事件处理循环

选择事件可以用 XselectInput()函数，以便告诉窗口只接受 EventMask 指定的事件。用法如下：

```
XselectInput(display, window, EventMask);
Display   *display;
Window    window;
long    EventMask;
```

其中，EventMask 是对选择的事件的屏蔽，可以取值为 ExposureMask 表示暴露事件、KeyPressMask 表示键盘按键事件、ButtonPressMask 表示鼠标按键事件，可通过它来设置需要选择一种或多种事件。

```
EventMask=ExposureMask|KeyPressMask|ButtonPressMask;
```

上述语句表示接受暴露事件、键盘按键事件和鼠标按键事件。

进入事件处理循环，获取事件可以采用 XNextEvent( )函数。

```
XNextEvent(display, &event);
```

用 switch 语句可针对不同事件作出处理，如：

```
switch (event.type) {
        case Expose: drawInToWindow(); break; /*  重画/刷新屏幕*/
        case ButtonPress: exit(0); break; /*  在鼠标按钮按下退出*/
        default: break;

}
```

1.1.7　图形上下文

GC(Graphics Contexts，图形上下文)可用来画图，并指定画图中的一些细节选项，它控制了所画图形的前景色、背景色、线条宽度和线条形式等要素。

首先，创建和设置 GC，采用 XcreateGC( )函数：

```
XcreateGC(display, drawable, valuemask, values)
Display   * display;
Drawable   drawable;
unsigned long    valuemask;
XGCValues   *values;
```

其中，drawable 可以是一个窗口或一个像素图；values 是 XGCValues 类型的结构；valuemask 是一个过滤，用来定义被读取 XGCValues 中的成员。

例如，gc1 用来画消息和矩形，gc2 则用来画椭圆。

```
XGCValues gs1,gs2; /*  图形/上下文设置  */
```

```
gc1 = XCreateGC(display, win, 0, &gs1); /* 0 ->使用默认值*/
gc2 = XCreateGC(display, win, 0, &gs2); /* 0 ->使用默认值*/
```

可以用 XSetForeground()设置前景，用 XSetBackground()设置背景等一些 GC 基本的设置函数，如：

```
/* Now override some of the defaults */
XSetForeground(display, gc1, BlackPixel(display, screen));   //设置前景色
XSetBackground(display, gc1, WhitePixel(display, screen)); //设置背景色
XSetLineAttributes(display, gc1, 2, LineSolid, CapRound, JoinRound);   //设置线
XSetStipple(display, gc2, stipple);                          //设置点
XSetFillStyle(display, gc2, FillOpaqueStippled);   //设置填充格式
```

通过 GC，可以画很多图形，比如文字、矩形以及填充一些图案。

```
XDrawString(display, win, gc1, 130, 20, message, strlen(message));
DrawRectangle(display, win, gc1, 40, 40, 320, 220);
FillArc(display, win, gc2, 50, 50, 300, 200, 0, 360*64);
```

1.1.8 完整的例子

```
/* win.c --- a very basic Xlib application, Jeff Pitchers, LUT 1993 */

/* The basic set of header files for any Xlib program */
#include <stdio.h>
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Xresource.h>
#include <X11/keysym.h>

Display *display;   /* Server or display to which a connection will be made */
int screen;         /* Screen number to use on that display */
Window win;         /* The applications one and only window */
XEvent event;       /* The event data structure */
GC gc1, gc2;        /* Graphics context data structures */
XGCValues gs1,gs2; /* Graphics context settings */
char *message;      /* Message printed in window */
static char progDefaultMessage[] = {"A Simple Xlib Program"};

main(argc, argv)
int argc; char **argv;
```

```
{
  /* 尝试连接服务器并使用默认屏幕*/
  if ((display = XOpenDisplay(NULL)) == NULL) {
    perror("Can't connect to server");
    exit(1);
  }
  screen = DefaultScreen(display);

  /*创建一个窗口*/
  win = XCreateSimpleWindow(display, RootWindow(display, screen),
          100, 200, 400, 300, 1, BlackPixel(display, screen),
          WhitePixel(display, screen));

  /* Set up which event types the window will handle */
  XSelectInput(display, win, ExposureMask | ButtonPressMask);

  /* 图形上下文*/
  setUpGCs();

  /* 尝试从用户定义资源设置中获得消息*/
  if ((message = XGetDefault(display, "win", "message")) == NULL)
    message = progDefaultMessage;

  /* 映射为可显示出来的窗口*/
  XMapWindow(display, win);

  /* Enter an infinite loop, awaiting and responding to the chosen events */
  while (1) {
    XNextEvent(display, &event);
    switch (event.type) {
      case Expose: drawInToWindow(); break; /* Draw/re-draw everything */
      case ButtonPress: exit(0); break; /* Quit on mouse button press */
      default: break;
    }
  }
}

setUpGCs()
/* Set up 2 graphics context resources in the display server */
/* gc1 for message and rectangle; gc2 for ellipse */
```

```
{
/* Define a bitmap for the stipple pattern used to fill the ellipse */
#define stipple_width 3
#define stipple_height 3
   static char stipple_bits[] = {0x02, 0x05, 0x02};
   Pixmap stipple;

   /* Create the stipple for ellipse fill pattern of gc2 */
   if ((stipple = XCreateBitmapFromData(display, RootWindow(display, screen),
      stipple_bits, stipple_width, stipple_height)) == 0) {
      perror("Can't create bitmap");
      exit(1);
   }

   gc1 = XCreateGC(display, win, 0, &gs1); /* 0 -> use defaults */
   gc2 = XCreateGC(display, win, 0, &gs2); /* 0 -> use defaults */
   /* Now override some of the defaults */
   XSetForeground(display, gc1, BlackPixel(display, screen));
   XSetBackground(display, gc1, WhitePixel(display, screen));
   XSetLineAttributes(display, gc1, 2, LineSolid, CapRound, JoinRound);
   XSetStipple(display, gc2, stipple);
   XSetFillStyle(display, gc2, FillOpaqueStippled);
}

drawInToWindow()
{
   /* Print message, draw restangle and filled ellipse as defined */
   /* by the GC */
   XDrawString(display, win, gc1, 130, 20, message, strlen(message));
   XDrawRectangle(display, win, gc1, 40, 40, 320, 220);
   XFillArc(display, win, gc2, 50, 50, 300, 200, 0, 360*64);
}
```

1.2 X Window 系统

1.2.1 X Window 系统的基本概念

X Window 系统始于 1984，它是 MIT 在发展雅典娜计划(Project Athena)时，为适应