开源之夏

openKylin主题框架轻松构建百变样式

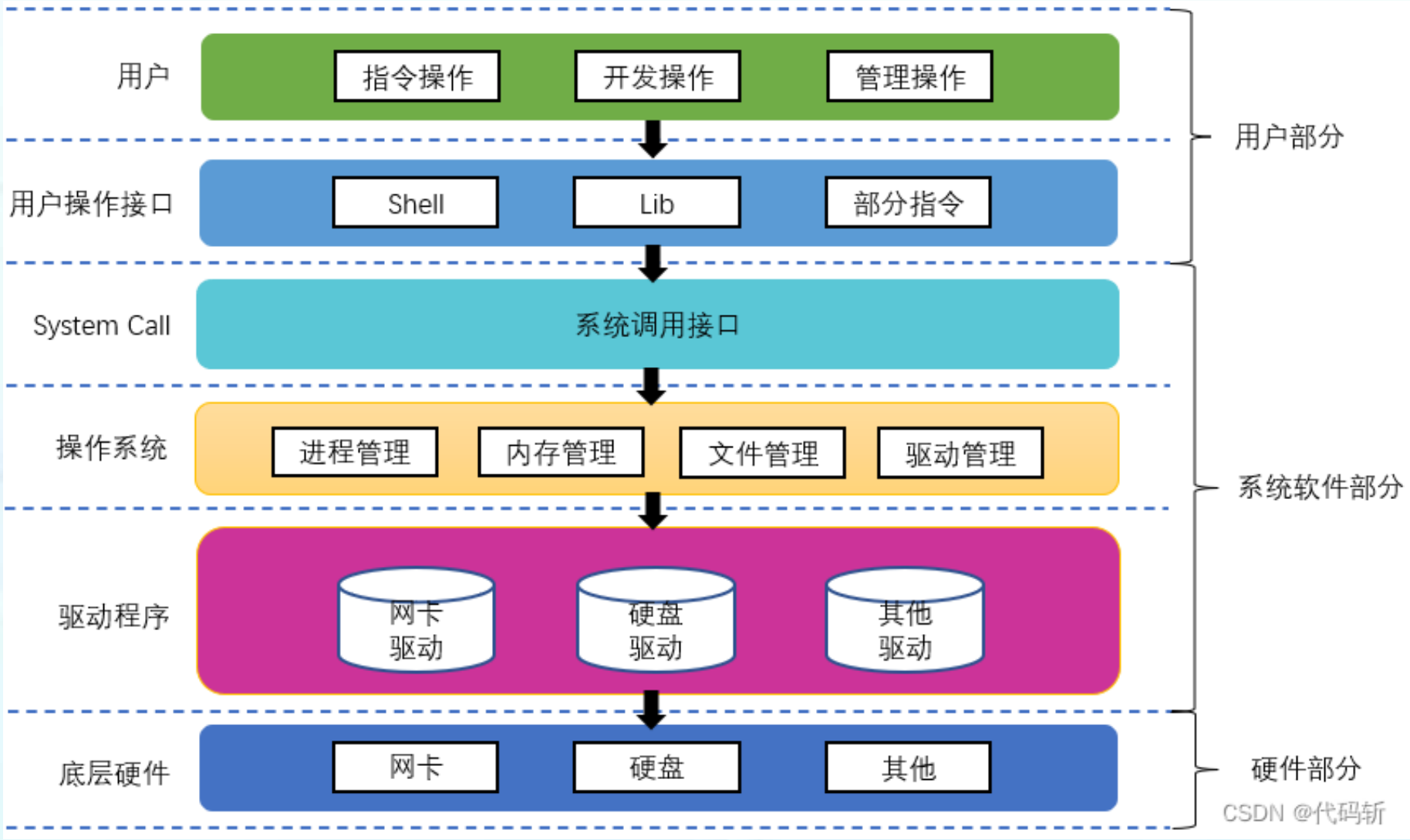
openKylin 段凯文

ISCAS 中国科学院软件研究所 Institute of Software Chinese Academy of Sciences
HUAWEI

如何在openKylin上构建一套心仪的系统主题？

了解系统主题

了解系统


用户
指令操作
开发操作
管理操作
用户操作接口
Shell
Lib
部分指令
System Call
系统调用接口
操作系统
进程管理
内存管理
文件管理
驱动管理
驱动程序
网卡驱动
硬盘驱动
其他驱动
底层硬件
网卡
硬盘
其他
用户部分
系统软件部分
硬件部分
CSDN @代码斩

了解系统

Qt 和 GTK+ 都是用于创建图形用户界面 (GUI) 的工具包，用于开发跨平台的应用程序。

freedesktop.org 主持自由和开放源码软件的开发，重点是开放源码图形和桌面系统的互操作性和共享技术。我们自己不生产桌面，但我们的目标是帮助他人生产桌面。

了解系统主题

系统主题是指控制桌面环境外观的一组元素，包括窗口装饰、图标、光标、控件、壁纸、开机动画等。

系统主题的设计可以影响用户界面的整体美观性和用户体验。在openKylin中，你可以通过更改系统主题来改变桌面环境的外观。

系统主题与GTK和Qt之间有密切的关系，因为GTK和Qt是两种用于创建图形用户界面（GUI）的跨平台工具包。系统主题通常用于定义应用程序的外观。GTK和Qt都允许开发者使用系统主题来确保应用程序与用户操作系统的外观一致。同时系统主题严格遵守freedesktop规范要求，保障系统主题的通用性。

样式主题加载机制

样式主题加载机制

openKylin中样式主题主要由主题框架控制。主题框架是qt本身具有的一个框架（QPlatformTheme），在qt5之后，它作为qt平台抽象架构的一部分存在，取代了原来的QStyle框架。

样式主题加载机制

QPA简介：

QPA即Qt Platform Abstraction，是Qt5里面的平台抽象层，用以取代Qt for Embedded Linux以及Qt4中的平台接口。QPA插件通过定义QPlatform开头的一系列类的子类实现，其中有两个根类，QPlatformIntegration和QPlatformTheme，前者用于窗口系统的集成，后者用于更深层次的平台主题化和集成。

```
admin@admin-ThinkPad-T490:~$ env|grep PLATFORM
QT_QPA_PLATFORMTHEME=ukui
QT_QPA_PLATFORM=xcb
admin@admin-ThinkPad-T490:~$
```

```
admin@admin-ThinkPad-T490:/usr/lib/x86_64-linux-gnu/qt5/plugins/platforms$ ls
libkxcb.so  libqeglfs.so  libqlinuxfb.so  libqminimalegl.so  libqminimal.so  libqoffscreen.so  libqvnc.so  libqxcb.so
admin@admin-ThinkPad-T490:/usr/lib/x86_64-linux-gnu/qt5/plugins/platforms$
```

```
dmin@admin-ThinkPad-T490:/usr/lib/x86_64-linux-gnu/qt5/plugins/platformthemes$ ls
ibqgtk2.so  libqgtk3.so  libqt5-ukui-platformtheme.so  libqxdgdesktopportal.so
dmin@admin-ThinkPad-T490:/usr/lib/x86_64-linux-gnu/qt5/plugins/platformthemes$
```

样式主题加载机制

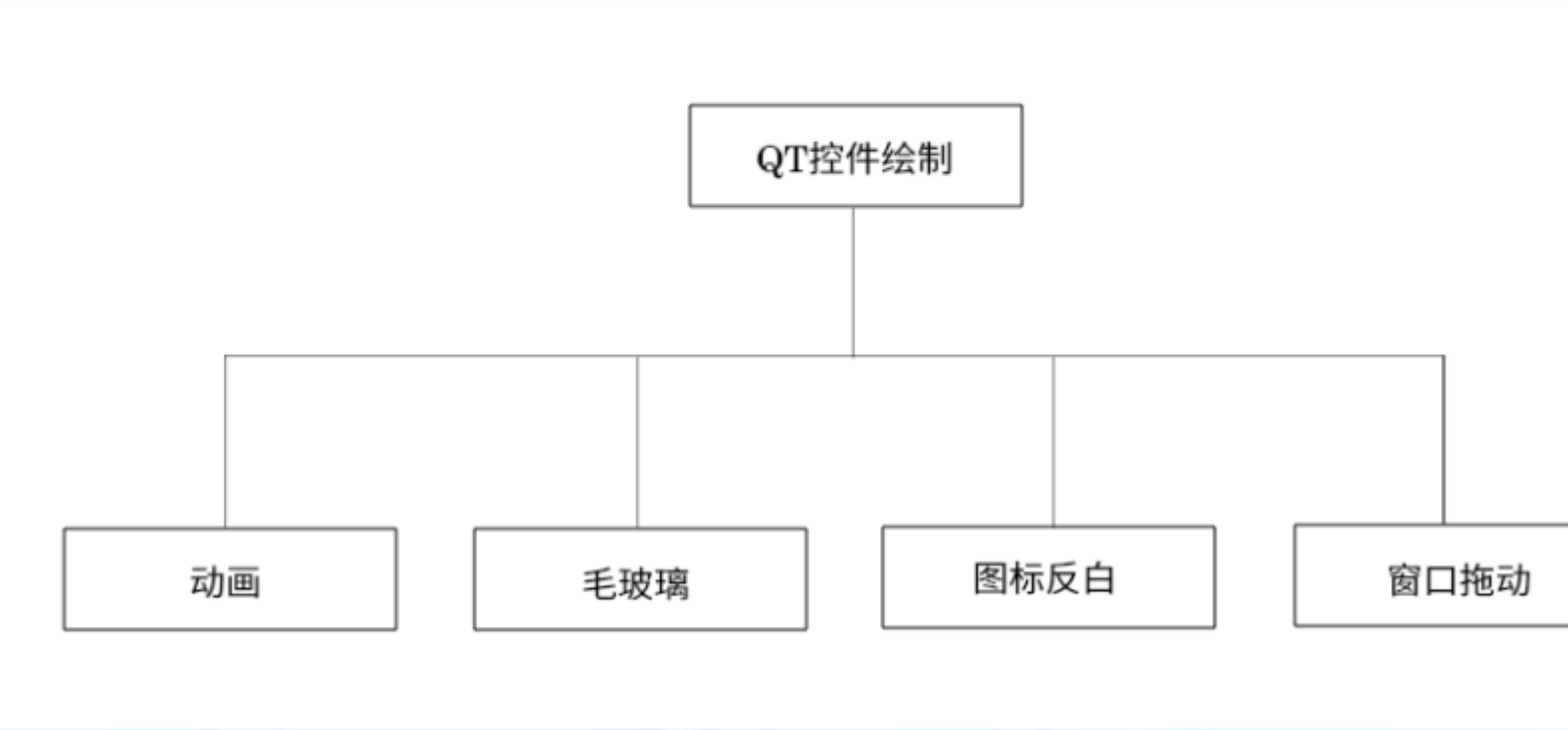
主题框架功能结构组成

主题框架大致分为：控件绘制部分与非控件绘制部分。

样式主题加载机制

主题框架的核心：QT基础控件绘制

QT基础控件的绘制为主题框架最核心的功能，动画、毛玻璃、图标反白、窗口拖动等功能都是基于在控件整个的绘制过程中添加属于我们自己的特性。



样式主题加载机制

QPushbutton::paintevent源码

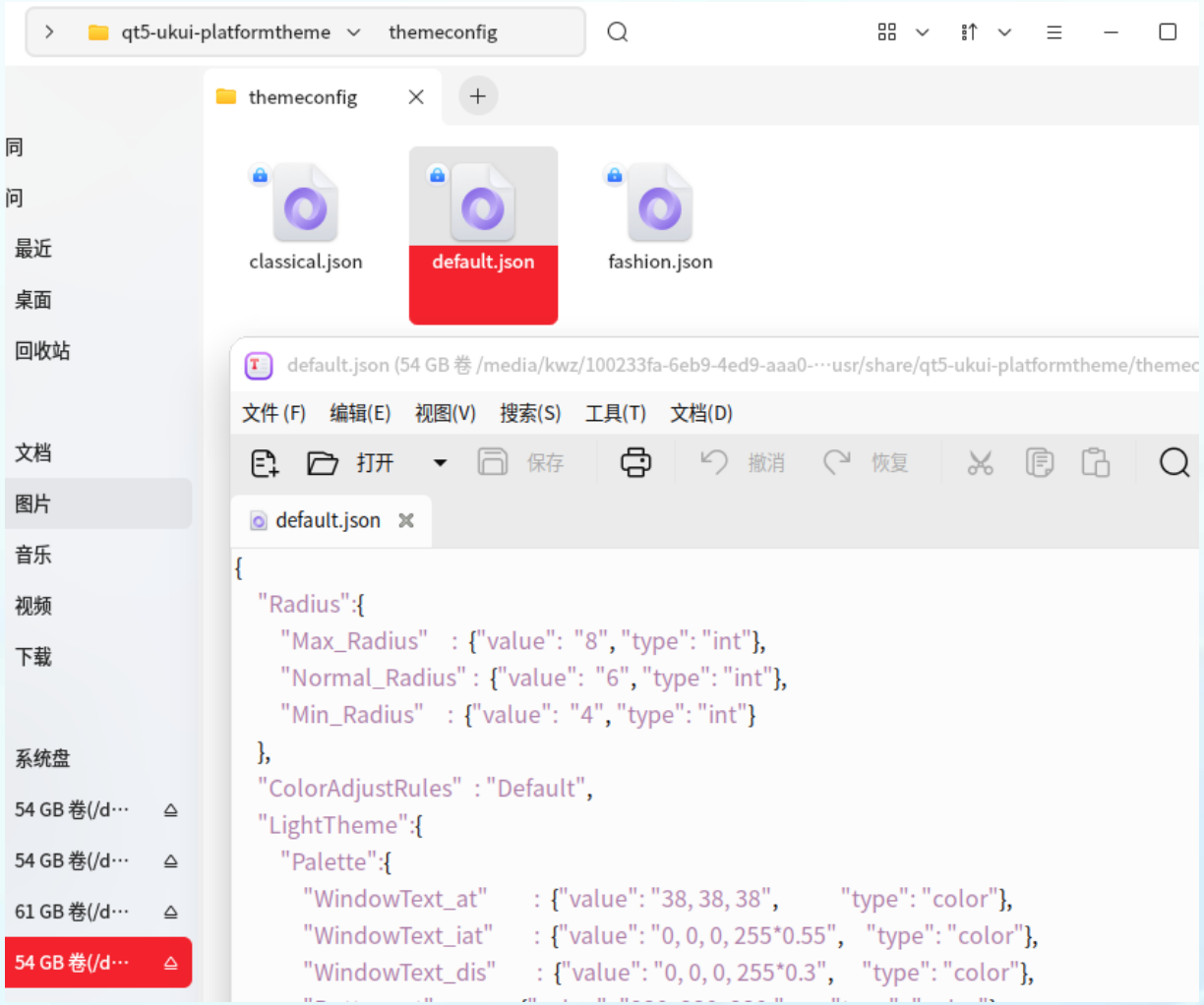
在QPushbutton的paintevent函数中，可以看到它调用了drawControl(QStyle::CE_PushButton, option);来进行绘制，通过配置相关插件，使得p.drawcontrol指向我们自己定义的Style，从而走我们ukui来对控件进行绘制处理。

样式主题加载机制

```
2798        case CE_PushButton:
2799        {
2800            if (const QStyleOptionButton *button = qstyleoption_cast<const QStyleOptionButton *>(option)) {
2801                proxy()->drawControl(CE_PushButtonBevel, option, painter, widget);
2802                QStyleOptionButton subopt = *button;
2803                subopt.rect = proxy()->subElementRect(SE_PushButtonContents, option, widget);
2804                proxy()->drawControl(CE_PushButtonLabel, &subopt, painter, widget);
2805    //            if (option->state & State_HasFocus || button->features & QStyleOptionButton::DefaultButton) {
2806    //                painter->save();
2807    //                painter->setPen(QPen(highLight_Click(option), 2, Qt::SolidLine, Qt::RoundCap, Qt::RoundJoin));
2808    //                painter->setBrush(Qt::NoBrush);
2809    //                painter->setRenderHint(QPainter::Antialiasing, true);
2810    //                int x_Radius = 4;
2811    //                int y_Radius = 4;
2812    //                painter->drawRoundedRect(button->rect.adjusted(1, 1, -1, -1), x_Radius, y_Radius);
2813    //                painter->restore();
2814    //            }
2815                return;
2816            }
2817            break;
2818        }
```

因此要想熟悉主题框架，必须也要熟悉QT源码，熟悉各个控件的绘制流程，还包括控件的size的处理，复杂控件的每个部分的处理，控件的继承关系。

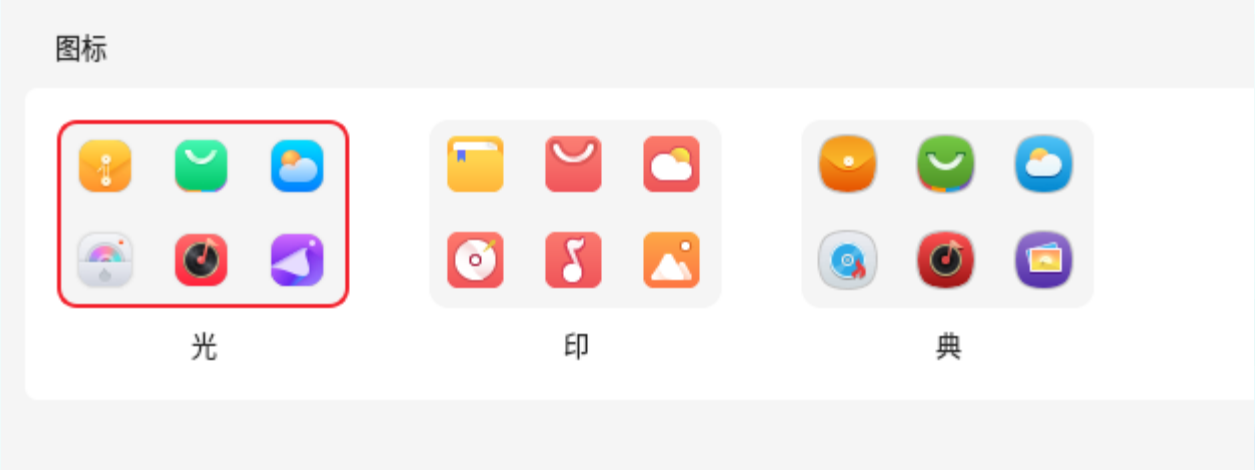
样式主题加载机制



在openKylin 2.0版本中，即将开放更多的主题样式配置，通过这些系统级配置文件，可以实现更加多元的样式主题。

图标主题加载机制

图标主题加载机制



openKylin中图标主题包含了大量的系统中会用的到图标资源，文件、文件夹、应用程序图标等的图标文件。这些图标主题位于系统 /usr/share/icons/ 目录下。

图标主题主要是提供文件资源，Qt和Gtk对系统图标各自封装了对应函数：

- Qt 提供了一套称为 Qt 图标主题（QIcon Theme）的机制，其中 QIcon:fromtheme 常用于加载和显示图标；
- GTK+ 提供了一些用于加载和显示图标的函数，如 gtk_image_new_from_icon_name() 和 gtk_image_new_from_file()

图标主题加载机制

```
QIcon QIcon::fromTheme(const QString &name)
{
    QIcon icon;

    if (qtIconCache()->contains(name)) {
        icon = *qtIconCache()->object(name);
    } else if (QDir::isAbsolutePath(name)) {
        return QIcon(name);
    } else {
        QPlatformTheme * const platformTheme = QGuiApplicationPrivate::platformTheme();
        bool hasUserTheme = QIconLoader::instance()->hasUserTheme();
        QIconEngine * const engine = (platformTheme && !hasUserTheme) ? platformTheme->createIconEngine(name)
                                                                      : new QIconLoaderEngine(name);
        QIcon *cachedIcon  = new QIcon(engine);
        icon = *cachedIcon;
        qtIconCache()->insert(name, cachedIcon);
    }

    return icon;
}
```

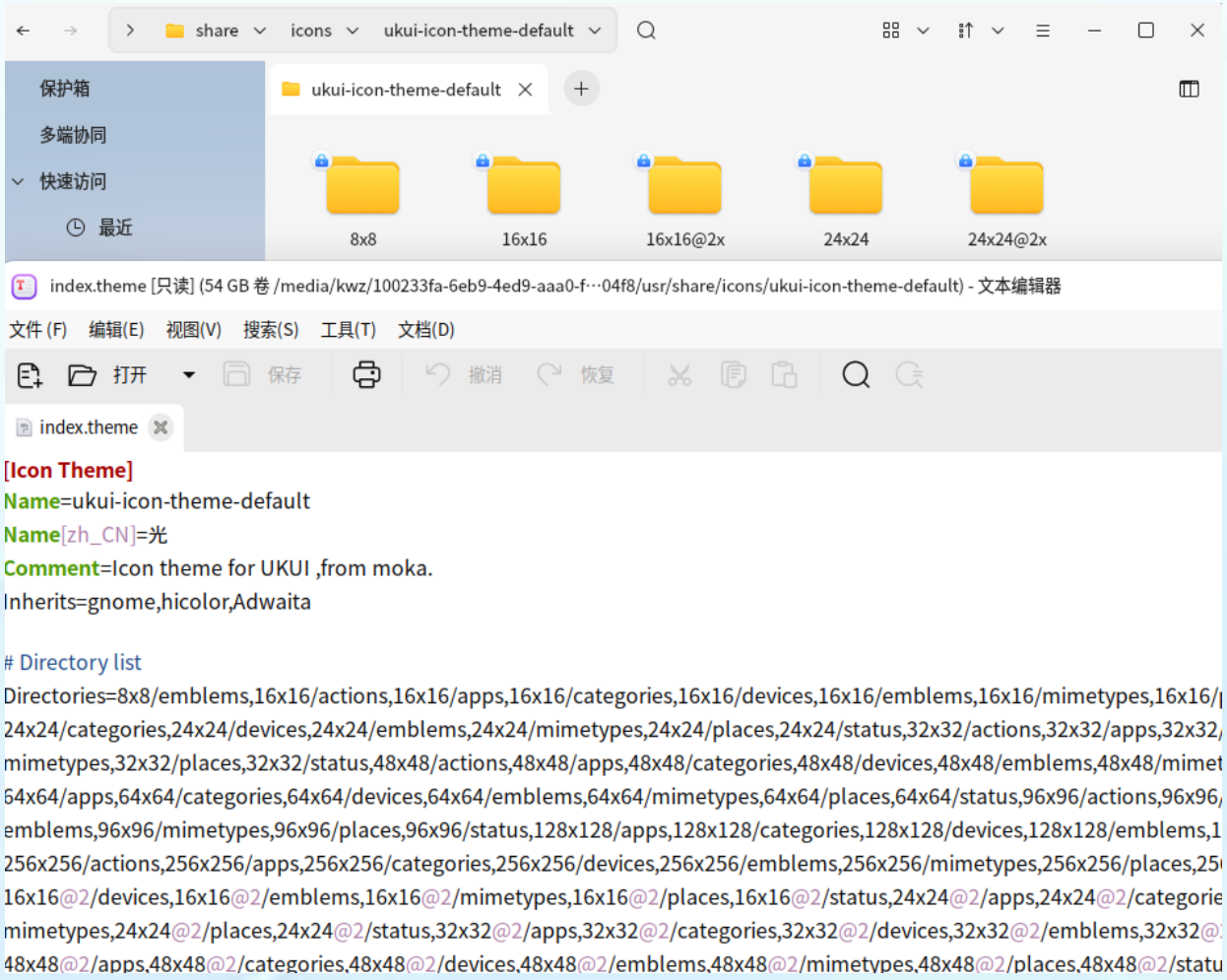
QIcon::fromTheme函数的具体流程如下：

- 该函数会调用QIconLoader::instance()获取一个QIconLoader实例。
- 它会调用QIconLoader::themeName()获取当前主题的名称。
- 它会调用QIconLoader::loadIcon函数，该函数会根据主题名称和图标名称加载图标。
- 它会返回一个QIcon对象，该对象包含了加载的图标。

图标主题加载机制

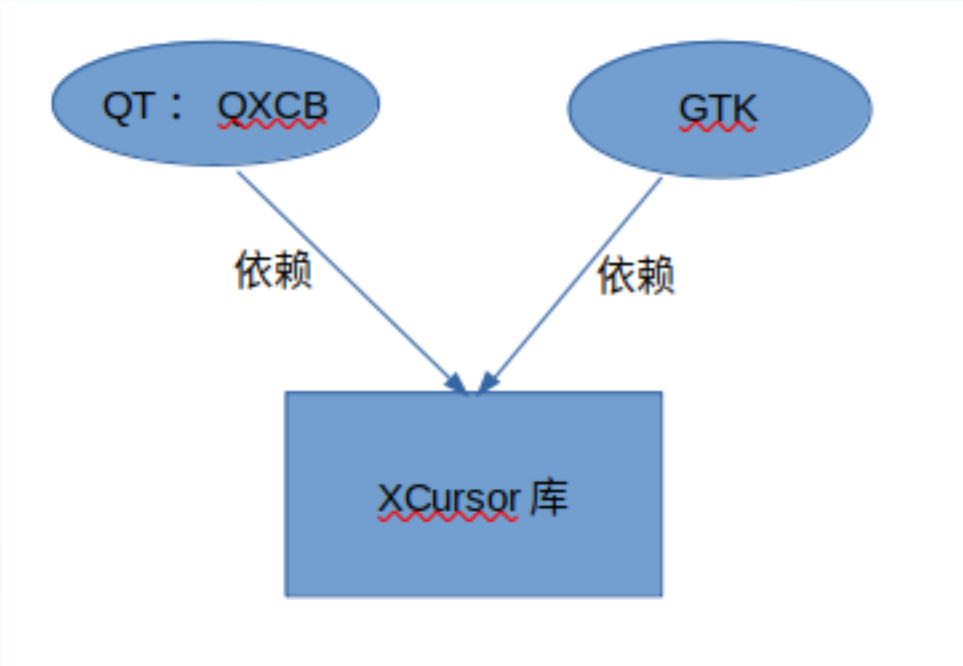
在获取到系统图标主题名称后，会优先去系统中/usr/share/icons/对应主题目录中寻找图标。

这时图标目录中的index.theme文件就起到向导作用，通过DIrectories、Inherits等标签帮助程序定位到使用图片的具体目录。



光标主题加载机制

样式主题加载机制



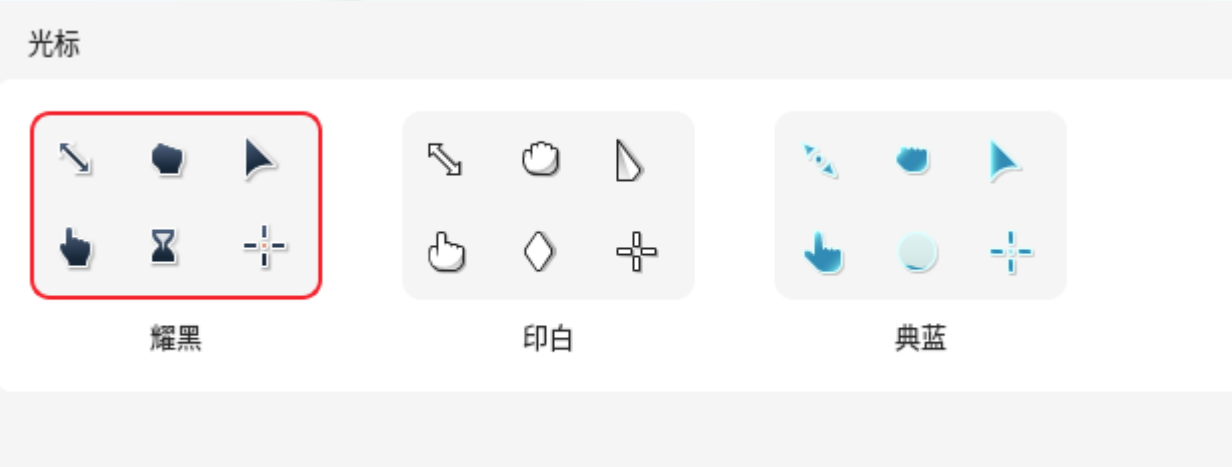
openKylin中光标绘制由 Xcursor 提供，Xcursor 是一个简单的库，其目的在于帮助定位和加载光标。

XCursor 提供了很多的接口，方便其他的组件的调用。
其中QT与GTK在X11中关于光标绘制就对XCursor库进行了依赖。

Xcursor 的功能是由几层代码组成：

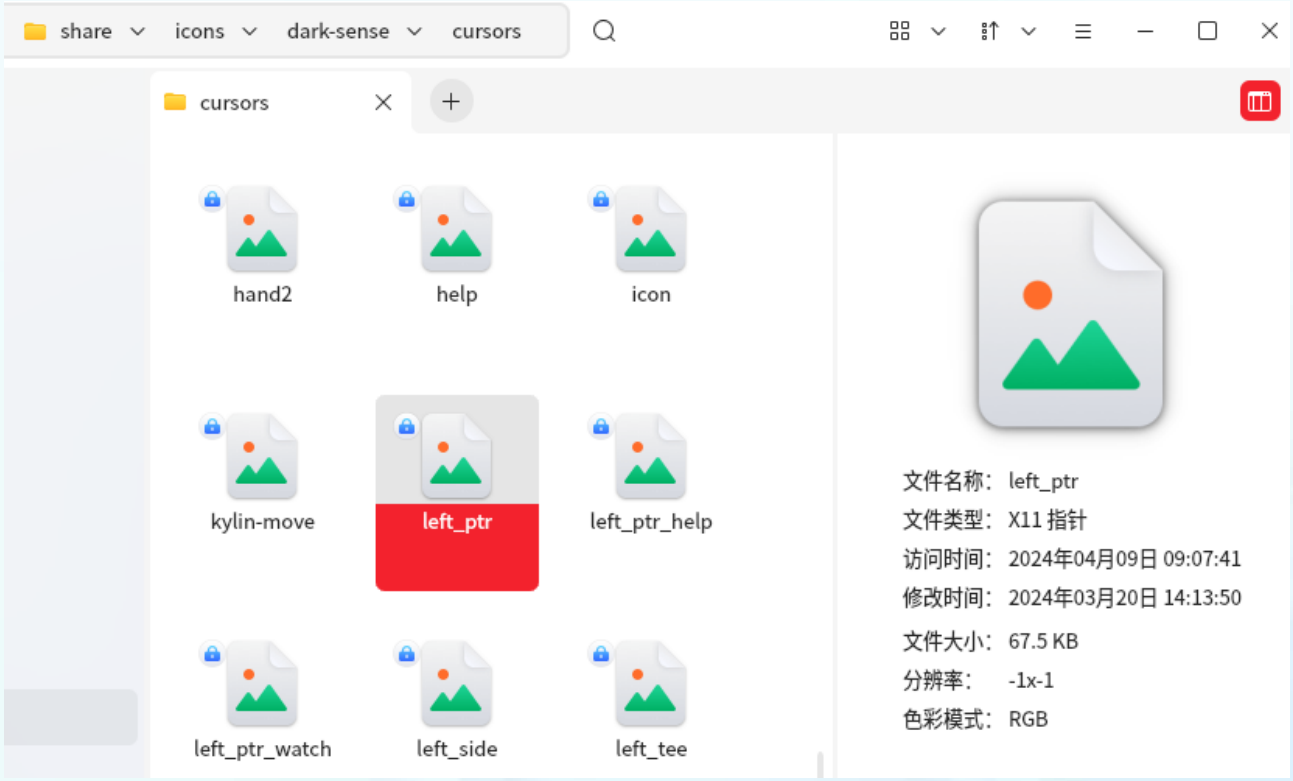
- 最上层代码用于创建光标，它可以从光标文件中加载的图像或标准的 X 光标来创建光标。
- 中间层会根据库路径和主题来定位光标资源文件位置；
- 底层代码是从资源文件中加载光标文件；

光标主题加载机制



Xcursor 创造了一种新的光标文件格式（.cur/X11 cursor）。每个光标文件是由一个或多个光标图像（.png）组成。每个光标图像都有一个尺寸大小的标签，这样 Xcursor 就可以自动选择最合适的尺寸大小。当一个光标文件由多个相同尺寸大小的光标图像组成时，光标可以逐个加载，会形成一个动画（watch光标）。

光标主题加载机制



Xcursor 遵循 freedesktop.org 关于主题图标的规范。它使用的默认搜索路径是 $HOME/.icons, /usr/share/icons, /usr/share/pixmaps, /usr/X11R6/lib/X11/icons。在这些目录中，它会使用当前设置主题名称进行目录搜索。在对应设置主题名的目录中，它会在 "cursors" 子目录中寻找光标文件并使用沿着路径找到的第一个光标文件。

如果选中的主题中无所需光标文件，Xcursor 还会在每个主题目录中寻找一个 "index.theme" 文件，以找到继承的主题，并沿着路径搜索这些主题。如果没有设置主题，或者没有找到指定主题的光标，Xcursor 会从 "Default" 主题获取光标文件。

光标主题加载机制

```
Cursor
XcursorLibraryLoadCursor (Display *dpy, const char *file)
{
    int         size = XcursorGetDefaultSize (dpy);
    char        *theme = XcursorGetTheme (dpy);
    XcursorImages   *images = XcursorLibraryLoadImages (file, theme, size);
    Cursor      cursor;

    if (!file)
        return 0;

    if (!images)
    {
    int id = XcursorLibraryShape (file);

    if (id >= 0)
        return _XcursorCreateFontCursor (dpy, id);
    else
        return 0;
    }
    cursor = XcursorImagesLoadCursor (dpy, images);
    XcursorImagesDestroy (images);
#if defined HAVE_XFIXES && XFIXES_MAJOR >= 2
    XFixesSetCursorName (dpy, cursor, file);
#endif
    return cursor;
}
```

XcursorLibraryLoadCursor函数中根据传入数据去调用XcursorImageLoadCursor函数

Xcursor 提供的加载光标 API 为 XcursorLibraryLoadCursor，该 API 会根据传入 X 特有的数据类型 Display，去设置环境光标大小、主题、使用光标名称。Display 数据结构类型会传递包含视觉信息、屏幕的信息、等等有深度的信息。

通过获取主题相关信息，去系统中找到对应光标文件以及光标文件中适当大小的图像文件，并最终通过 XcursorImageLoadCursor 函数去创建光标。

社区参与

openKylin主题制作工坊能够合理协助用户制作个性化系统主题，其中包含壁纸、光标、图标、系统样式、开机动画、grub界面、全局主题等。

用户根据个人需求选择新建全局、图标、光标、开机动画、grub界面主题，其中全局主题包含图标、光标、开机动画、grub界面、系统样式和壁纸等主题，其他四个类型仅包含各自对应类型主题。

社区参与

UKUITheme SIG组致力于帮助社区用户进行与系统主题相关的软件包的参与使用，且帮助用户实现个性化自主打包。

社区用户可以将通过主题制作工坊生成的个性化主题deb包发送至UKUITheme SIG的邮件列表，我们将筛选合适的主题进行集成，期待未来软件商店中有你的个性化主题包。

社区参与

2023年开源之夏的活动中，openKylin社区上线3个项目，提交结项报告2份，最终通过组委会审核公示2个项目。

今年社区预计上线4+项目，欢迎大家提交项目申请。

2023年openKylin项目结项证书

加入openKylin城市用户组

openKylin城市用户组：简称OKUG,是为方便区域开源爱好者交流openKylin系统版本及用户体验、Linux技术及生态建设，探讨开源操作系统产业趋势及开源技术贡献而成立的区域型城市组织。

- 开源爱好者的本地圈子
- Linux开源技术及生态支持者的私享会
- openKylin社区爱好者的学习、交流大本营
-

扫码加入广州用户组

开源之夏

Thanks

ISCAS 中国科学院软件研究所 Institute of Software Chinese Academy of Sciences

HUAWEI