华清远见 FARSIGHT 嵌入式培训专家

嵌入式学院 华清远见旗下品牌

嵌入式 Linux Framebuffer 驱动开发

版权

- 华清远见嵌入式培训中心版权所有；
- 未经华清远见明确许可，不能为任何目的以任何形式复制或传播此文档的任何部分；
- 本文档包含的信息如有更改，恕不另行通知；
- 保留所有权利。

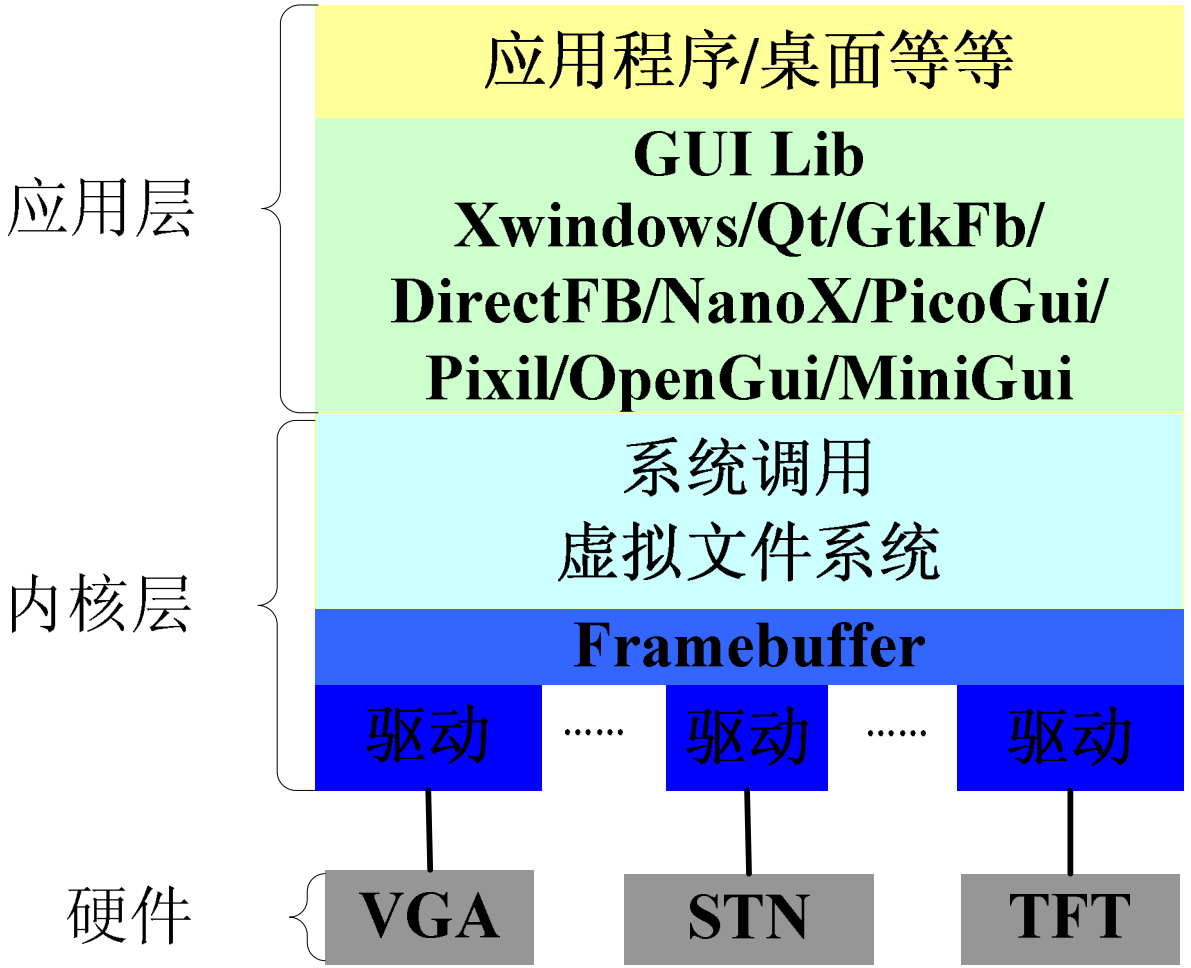
知识点

- *帧缓冲(Framebuffer)显示技术*
- 内核对 Framebuffer 的支持（编译配置选项）
- Framebuffer 的使用
  - 使用 Framebuffer 程序样例
  - 基于 Framebuffer 的 GUI 简介

帧缓冲(Framebuffer)显示技术概要

- 不断发展的显示系统
  - 字符界面
  - 图形界面
  - 桌面系统
    - 平面、3D
- Linux 系统是工作在保护模式下，所以用户态进程是无法象 DOS 那样使用显卡BIOS 里提供的中断调用来实现直接写屏
- Framebuffer 是出现在 Linux 2.2（及以后）内核当中的一种驱动程序接口
- Framebuffer 是一项重要的技术，它提供了一种机制使得在 Linux 系统中用户空间程序（应用程序）直接写屏变的非常简单，且不用关心硬件细节问题
- 在嵌入式 Linux 系统中几乎无一例外的都采用了这种接口实现显卡或LCD 驱动

Framebuffer 在Linux系统中的位置



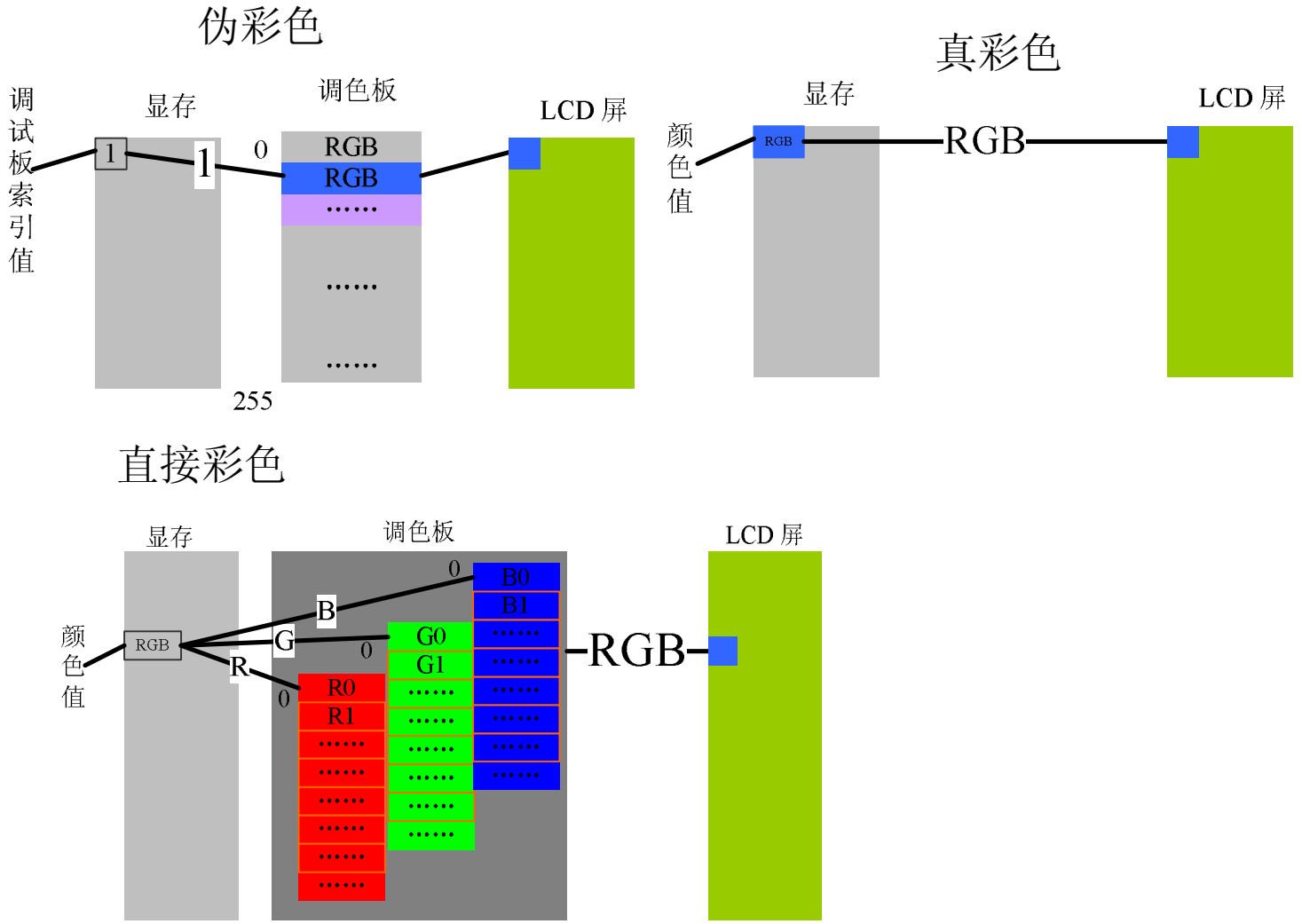
显示设备分类

- 按显示设备所用的显示器件分类，有阴极射线管(CRT)显示器、液晶显示器(LCD)、等离子显示器等。
- 按所显示的信息内容分类，有字符显示器、图形显示器、图像显示器三大类。
- 分辨率：是指显示器所能表示的像素个数。如1024*768，800*600 等
- 灰度级：是指黑白显示器中所显示的像素点的亮暗差别，在彩色显示器中则表现为颜色的不同

Framebuffer支持的颜色方式

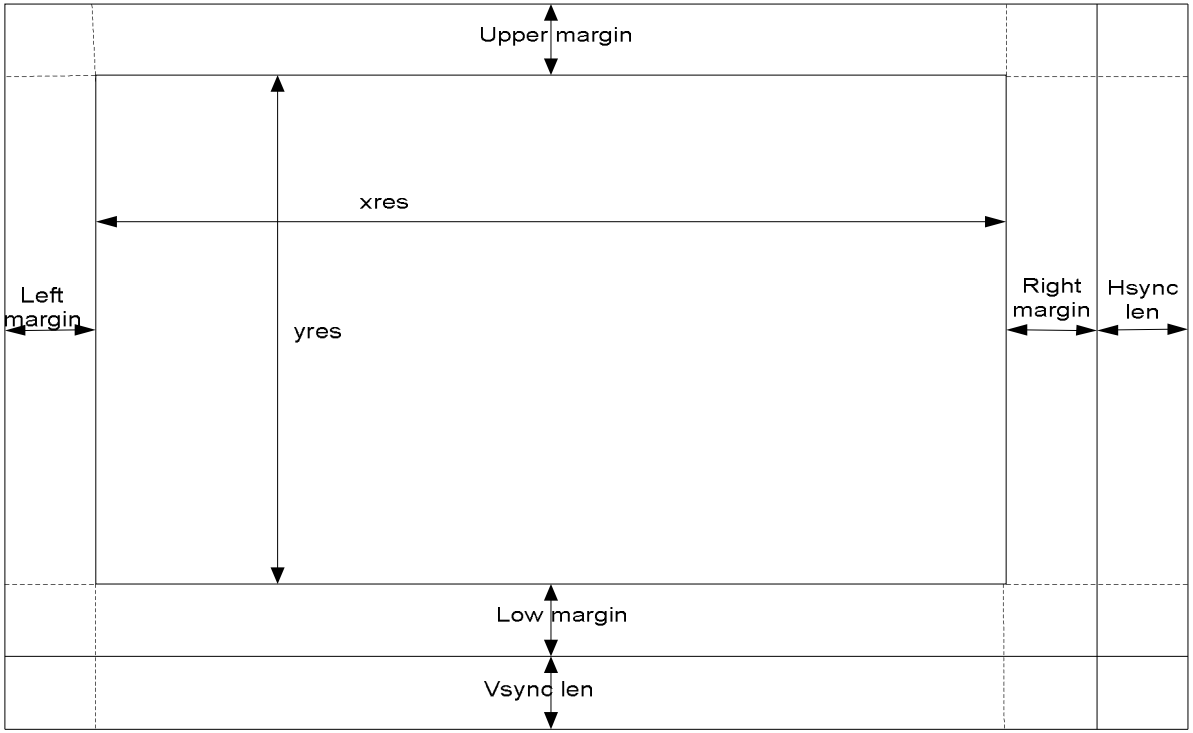
- Framebuffer支持多种颜色显示方式：
  - 单色（Monochrome）
  - 伪彩色（Pseudo color）
  - 真彩色（True color）
  - 直接彩色（Direct color）
  - 灰度（Grayscale displays）

伪彩色、真彩色、直接彩色异同



帧缓冲(Framebuffer)显示设备时序

- 标准化，保证显卡和显示设备的兼容性



帧缓冲驱动程序的数据结构

- fb_info
- fb_fix_screeninfo
- fb_var_screeninfo
- fb_ops

fb_info 结构（1）

```
struct fb_info {
    struct fb_var_screeninfo var;
    struct fb_fix_screeninfo fix;
    struct fb_monspecs monspecs;
    struct fb_cmap cmap;
    struct fb_ops *fbops;
    ……
}
```

- fb_info结构是Linux为帧缓冲设备定义的驱动层接口，用于用户在内核空间的调用。它不仅包含了底层函数，而且还有记录设备状态的数据。每个帧缓冲设备都有一个fb_info结构相对应。

fb_info 结构（2）

- 主要数据项包括
  - struct fb_var_screeninfo var;描述当前的可变参数
  - struct fb_fix_screeninfo fix;描述当前的固定参数
  - struct fb_monspecs monspecs;描述当前显示器的特有固定信息
  - struct fb_cmap cmap;描述当前颜色映射表
  - struct fb_ops *fbops;指向驱动设备工作所需的函数集，fb_ops 用户应用可以使用ioctl()系统调用来操作设备

fb_fix_screeninfo结构(1)

```
struct fb_fix_screeninfo {
    unsigned long smem_start;
    __u32 smem_len;
    __u32 type;
    __u32 visual;
    ……
};
```

- 该结构用来描述设备无关，不可变更的信息。用户可以使用FBIOGET_FSCREENINFO命令来获得这些信息

fb_fix_screeninfo结构（1）

- 结构包含主要的项
  - unsigned long smem_start;描述缓冲区起始地址（物理地址）
  - __u32 smem_len;描述缓冲区长度
  - __u32 type;描述fb类型，比如TFT或STN类型
  - __u32 visual;描述显示颜色是真彩色，伪彩色还是单色

fb_var_screeninfo结构(1)

```
struct fb_var_screeninfo{
__u32 xres;
__u32 yres;
__u32 xres_virtual;
__u32 yres_virtual;
__u32 xoffset;
__u32 yoffset;
……
}
```

- 该结构描述设备无关的，可更改的配置信息。应用程序可以使用FBIOGET_VSCREENINFO命令获得这些信息，使用FBIOPUT_VSCREENINFO命令写入这些信息

fb_var_screeninfo结构（2）

- 主要的数据项包含
  - __u32 xres;　　//可见分辨率
  - __u32 yres;
  - __u32 xres_virtual;　// 虚拟分辨率
  - __u32 yres_virtual;
  - __u32 xoffset;　　//从虚拟到可见分辨率的偏移
  - __u32 yoffset;
  - 以及屏幕四周的margin，像素时钟，同步等时序信息

帧缓冲驱动程序的API（1）

- 对帧缓冲设备的操作在Linux内核中是由fb_ops 结构定义的，用户可以使用 ioctl 系统调用来操作设备，该结构就是用来支持 ioctl 的这些操作的。
- 帧缓冲设备的驱动主要就是编写这些接口函数。

帧缓冲驱动程序的API（2）

- 常见的操作有：
  - int (*fb_open)(struct fb_info *info, int user);
  - int (*fb_release)(struct fb_info *info, int user);
  - ssize_t (*fb_read)(struct file *file, char __user *buf, size_t count, loff_t *ppos);
  - ssize_t (*fb_write)(struct file *file, const char __user *buf, size_t count, loff_t *ppos);
  - int (*fb_check_var)(struct fb_var_screeninfo *var, struct fb_info *info);
  - int (*fb_set_par)(struct fb_info *info);

  ……

帧缓冲驱动程序的API（3）

- int (*fb_open)(struct fb_info *info, int user);
  打开帧缓冲设备，类似文件操作的open方法
- int (*fb_release)(struct fb_info *info, int user);
  关闭帧缓冲设备，类似文件操作的release方法
- ssize_t (*fb_read)(struct file *file, char __user *buf, size_t count, loff_t *ppos);
  帧缓冲设备的读操作，类似文件操作的read方法
- ssize_t (*fb_write)(struct file *file, const char __user *buf, size_t count, loff_t *ppos);
  帧缓冲设备的写操作，类似文件操作的write方法
- int (*fb_check_var)(struct fb_var_screeninfo *var, struct fb_info *info);检查帧缓冲设备的可变参数，如果不合法，然后修正这些参数

帧缓冲驱动程序的API（4）

- int (*fb_set_par)(struct fb_info *info);
  设置视频模式根据info->var成员
- int (*fb_blank)(int blank, struct fb_info *info);
  帧缓冲bank模式显示
- void (*fb_fillrect) (struct fb_info *info, const struct fb_fillrect *rect);
  画一个矩形区域
- void (*fb_copyarea) (struct fb_info *info, const struct fb_copyarea *region);
  拷贝数据从一个区域到另一个区域

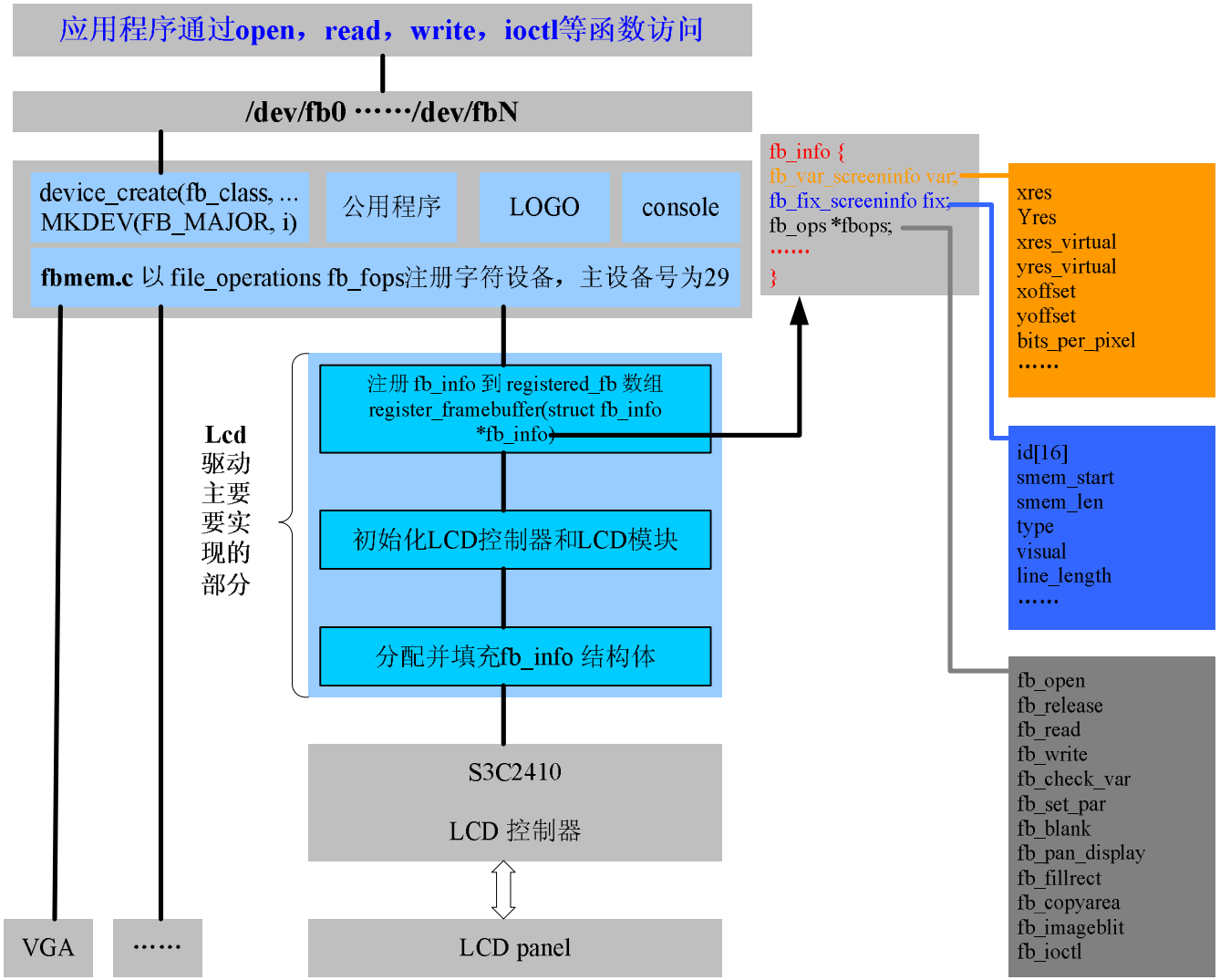
帧缓冲驱动程序的API（5）

- void (*fb_imageblit) (struct fb_info *info, const struct fb_image *image);
  把控制台过来的字符等信息画在lcd上
- int (*fb_cursor) (struct fb_info *info, struct fb_cursor *cursor);
  用于控制光标的位置，是和鼠标相关
- int (*fb_ioctl)(struct fb_info *info, unsigned int cmd, unsigned long arg);
  帧缓冲设备的具体ioctl操作
- int (*fb_mmap)(struct fb_info *info, struct vm_area_struct *vma);
  帧缓冲设备的内存映射操作

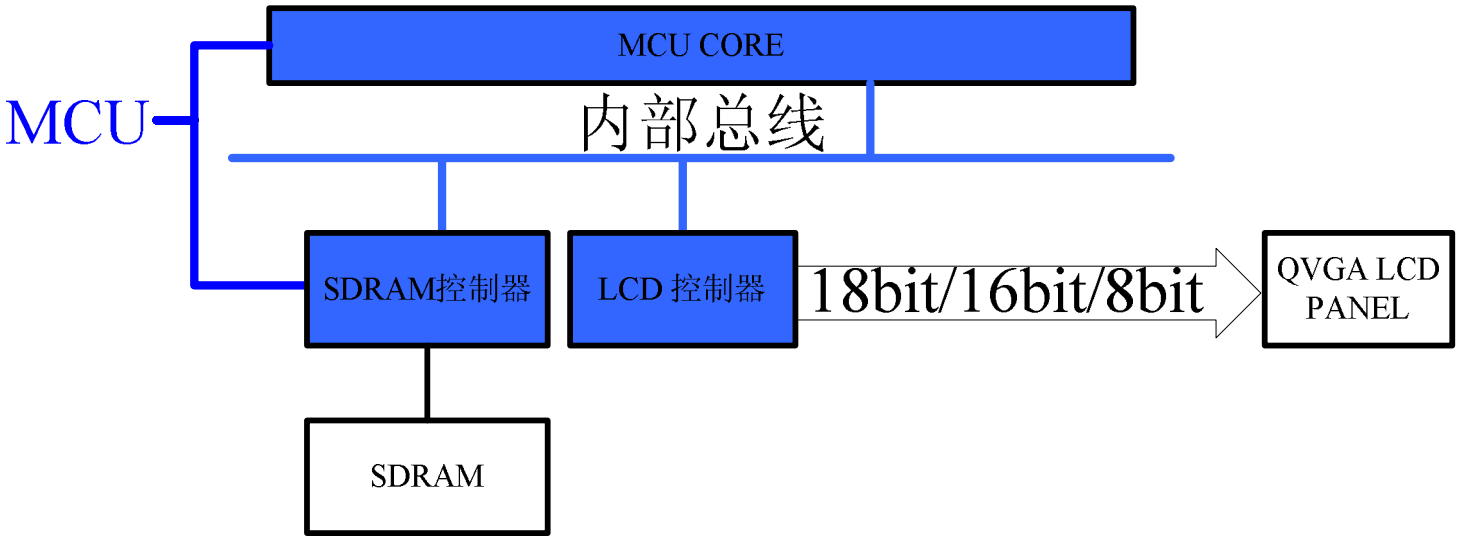
帧缓冲模块文件组成介绍

- 帧缓冲驱动程序核心实现部分
- drivers/video/
  - fbmem.c文件实现了帧缓冲驱动公共的调用 (核心)
    - **帧缓冲设备的注册函数**
      - int register_framebuffer(struct fb_info *fb_info);
    - **帧缓冲设备的注销函数**
      - int unregister_framebuffer(struct fb_info *fb_info);
  - modedb.c文件包含了所有的VESA标准显示模式信息
  - fbcmap.c文件实现了和调色板相关的调用
  - fbmon.c文件用来解析显示器的 EDID 并计算时序参数
  - cfbcopyarea.c文件实现了缓冲区中区域拷贝所需的函数
  - cfbfillrect.c文件实现了向缓冲区中指定矩形区域进行填充的方法
  - cfbimgblt.c文件实现了位图clipping操作
  - softcursor.c文件实现了软件光标
  - console/ 终端显示函数
  - logo/ LOGO相关代码和位图
  - 具体的显卡驱动（如：s3c2410.c 2410 lcd 驱动）

帧缓冲驱动核心层次结构



基于s3c2410的显示系统硬件组成



用户编写帧缓冲设备驱动（1）

- drivers/video/s3c2410fb.c
  - 实现了和硬件相关的方法，并填充必要的 fb_info 结构
- 编写驱动必须做的工作
  - 编写初始化函数：
    - 初始化LCD控制器
    - 设置显示模式和显示颜色数
    - 分配 LCD 显示缓冲区（DMA方式），缓冲区通常分配在片外SDRAM中，起始地址保存在LCD控制器寄存器中
  - 分配并初始化一个fb_info结构，填充其中的成员变量，并调用 register_framebuffer(&fb_info) 将 fb_info 注册到 framebuffer 系统中

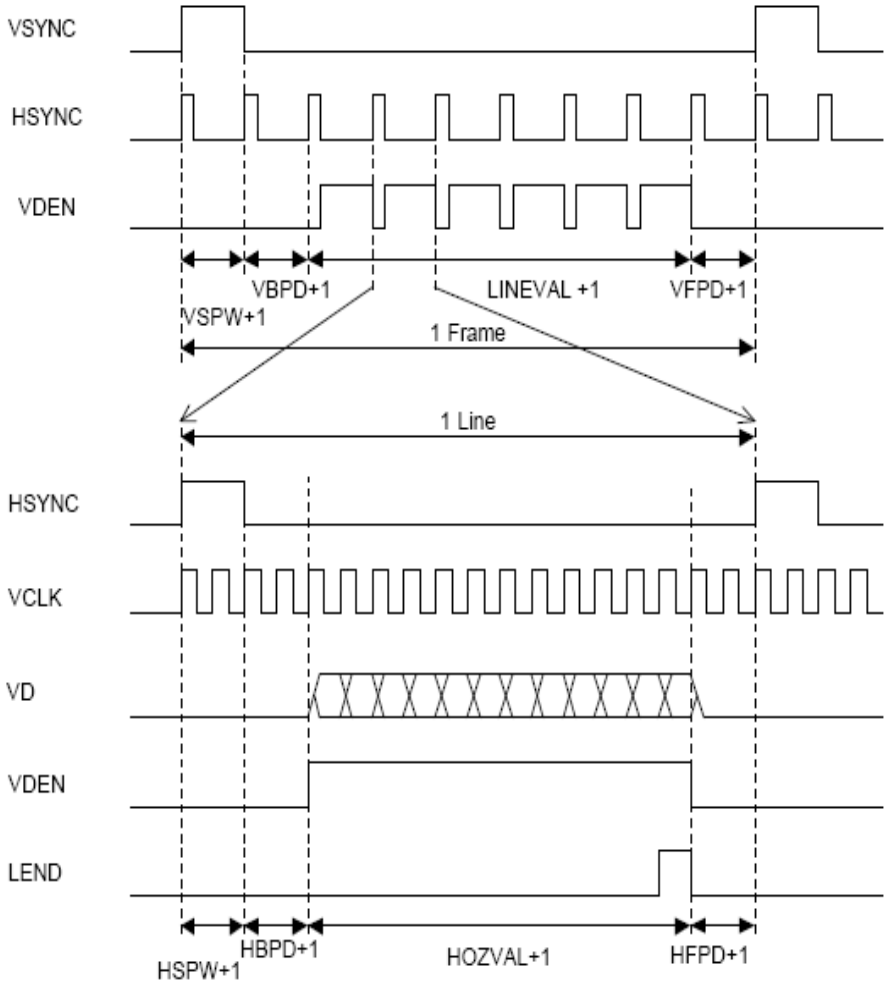
用户编写帧缓冲设备驱动（2）

- 编写结构fb_info中函数指针fb_ops对应的成员函数：例如：

```
static struct fb_ops s3c2410fb_ops = {
    .owner                  = THIS_MODULE,
    .fb_check_var   = s3c2410fb_check_var,
    .fb_set_par     = s3c2410fb_set_par,
    .fb_blank       = s3c2410fb_blank,
    .fb_pan_display = s3c2410fb_pan_display,
    .fb_setcolreg   = s3c2410fb_setcolreg,
    .fb_fillrect    = cfb_fillrect,
    .fb_copyarea    = cfb_copyarea,
    .fb_imageblit   = cfb_imageblit,
    .fb_cursor      = soft_cursor,
    .fb_ioctl       = s3c2410fb_ioctl,
};
```

- 然后调用int register_framebuffer(struct fb_info *fb_info);将帧缓冲驱动注册到系统中

S3C2410液晶显示时序



S3C2410液晶驱动核心数据结构（1）

```
struct pxafb_mach_info S3C2410_320X240_info = {
        .pixclock               = 270000,
        .xres           = 320,
        .yres           = 240,
        .bpp            = 16,
        .hsync_len      = 18,
        .left_margin    = 4,
        .right_margin   = 13,
       .vsync_len       = 4,
      .upper_margin     = 4,
      .lower_margin     = 4,
       .sync            = FB_SYNC_HOR_HIGH_ACT |
FB_SYNC_VERT_HIGH_ACT,
```

S3C2410液晶驱动核心数据结构（2）

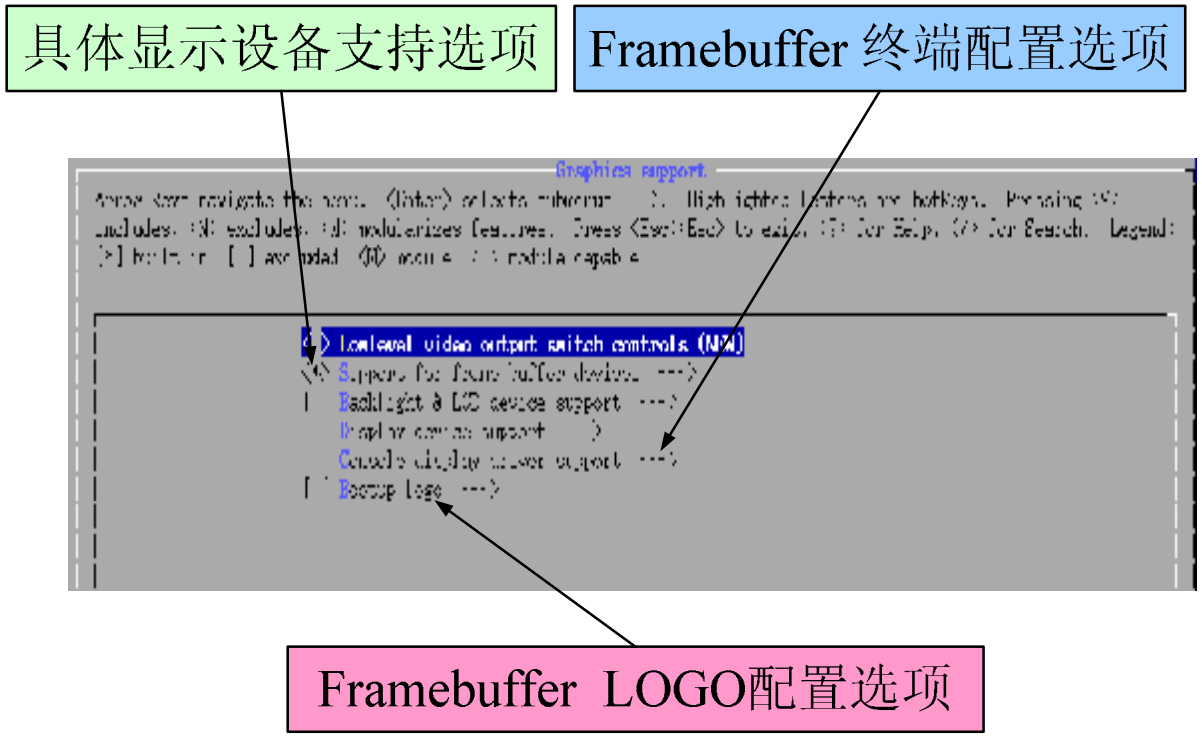
```
 .cmap_greyscale        = 0,
 .cmap_inverse          = 0,
 .cmap_static   = 0,
 .reg                   = {
.lcdcon1 = (7<<8)|(0<<7)|(3<<5)|(12<<1),
.lcdcon2 = (4<<24) | (239<<14) | (4<<6) | (4),
.lcdcon3 = (13<<19) | (319<<8) | (4),
.lcdcon4 = (13<<8) | (18),
.lcdcon5 = (1<<11) | (1<<10) | (1<<9) | (1<<8) | (0<<7) | (0<<6) |
   (1<<3)  |(0<<1) | (1),          }};
```

知识点

- 帧缓冲(Framebuffer)显示技术
- *内核对 Framebuffer 的支持（编译配置选项）*
- Framebuffer 的使用
  - 使用 Framebuffer 程序样例
  - 基于 Framebuffer 的 GUI 简介

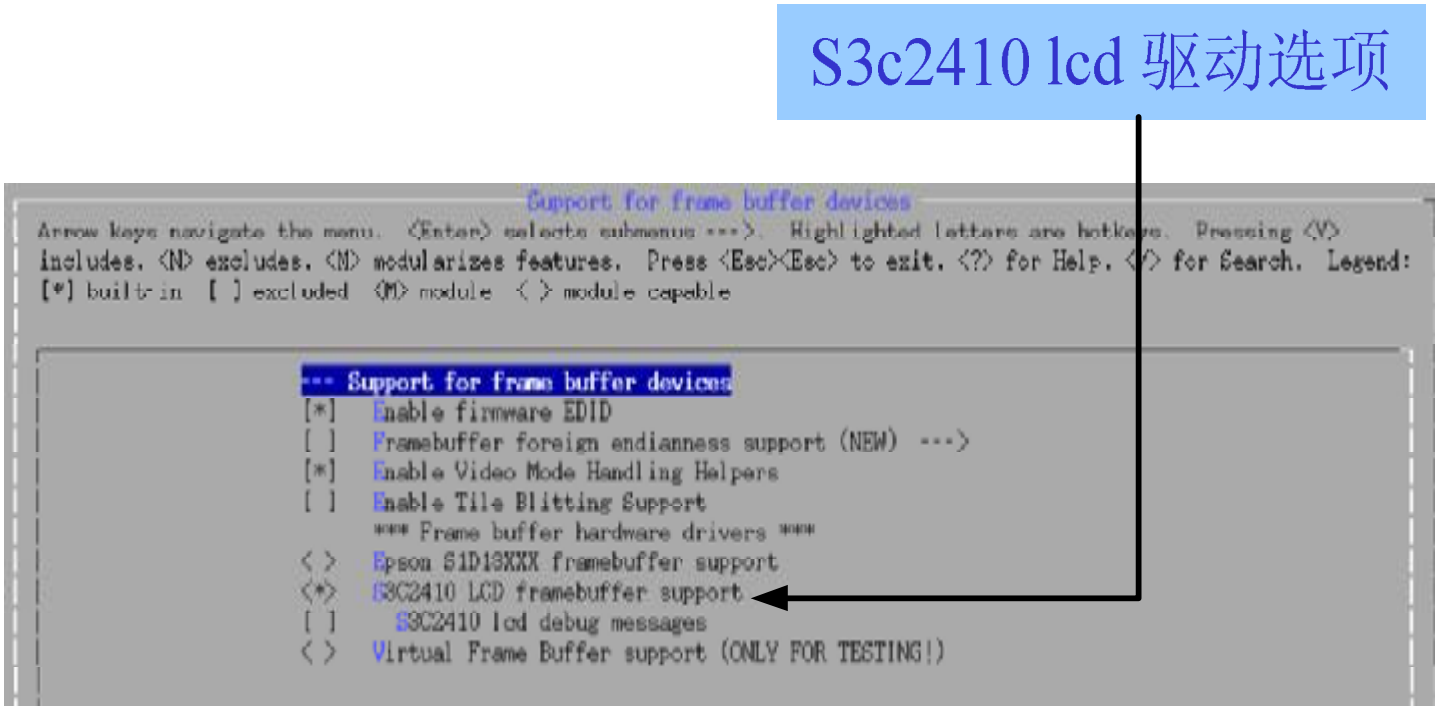
Framebuffer相关的内核配置选项

- make menuconfig
  - Device Drivers
    - Graphics support



Framebuffer相关的内核配置选项（续）

- Support for frame buffer devices



知识点

- ***帧缓冲(Framebuffer)显示技术概要***
- Framebuffer 原理简介
- 内核对 Framebuffer 的支持（编译配置选项）
- ***Framebuffer 的使用***
  - *使用 framebuffer 程序样例*
  - 基于 Framebuffer的 GUI简介

用户程序如何使用 framebuffer

- Framebuffer设备在系统中的体现

  0 = /dev/fb0 First frame buffer
  1 = /dev/fb1 Second frame buffer
  ...
  31 = /dev/fb31 32nd frame buffer

- Framebuffer 为应用程序提供了良好的接口
  - ioctl read write 等等
    - 比如cp /dev/fb0 screenshot，即是把fb0的内容拷贝到截图文件中
  - mmap方式使用

应用中使用framebuffer的基本步骤

- 在应用程序中，操作帧缓冲设备（/dev/fb0）的一般步骤如下：
  - 打开/dev/fb0设备文件(open)
  - 用ioctl操作取得当前显示屏幕的参数
  - 将屏幕缓冲区映射到用户空间 (mmap)
  - 映射后就可以直接读写屏幕缓冲区，进行绘图和图片显示了

Framebuffer 测试程序

```
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>
#include <linux/fb.h>
#include <sys/mman.h>
int main(int argc,  char *argv [])
{
   int fbfd = 0;
   struct fb_var_screeninfo vinfo;
   struct fb_fix_screeninfo finfo;
   long int screensize = 0;
   char *fbp = 0;
   int x = 0, y = 0;
   long int location = 0;
   /* 以读写方式打开显示设备节点文件 /dev/fb0 */
   fbfd = open("/dev/fb0", O_RDWR);
   if (fbfd <= 0) {
      printf("Error: cannot open framebuffer device.\n");
      return -1;
   }
   printf("The framebuffer device was opened successfully.\n");
   /* 得到framebuffer设备的固定屏幕信息 */
   if (ioctl(fbfd, FBIOGET_FSCREENINFO, &finfo)) {
      printf("Error reading fixed information.\n");
      return -2;
   }
   /* 得到可变屏幕信息 */
   if (ioctl(fbfd, FBIOGET_VSCREENINFO, &vinfo)) {
```

testfb.c

Framebuffer 测试程序（续1）

```
    /* 将映射的显存的大小 */
    screensize = vinfo.xres * vinfo.yres * vinfo.bits_per_pixel / 8;
    /* 将显存映射到用户空间 */
    fbp = (char *)mmap(0, screensize, PROT_READ | PROT_WRITE, MAP_SHARED, fbfd, 0);
    if ((int)fbp == -1) {
       printf("Error: failed to map framebuffer device to memory.\n");
       return -4;
    }
    printf("The framebuffer device was mapped to memory successfully.\n");
    x = 100; y = 100;       // Where we are going to put the pixel
    /* 在显存上彩色画渐变矩形条*/
    for (y = 100; y < 200; y++) {
       for (x = 100; x < 300; x++)  {
          location = (x+vinfo.xoffset) * (vinfo.bits_per_pixel/8) + (y+vinfo.yoffset) * finfo.line_length;
          /* 简单处理，假定是32位或者是16位 565模式*/
          if (vinfo.bits_per_pixel == 32)  {
             *(fbp + location) = 100;        // Some blue
             *(fbp + location + 1) = 15+(x-100)/2;     // A little green
             *(fbp + location + 2) = 200-(y-100)/5;    // A lot of red
             *(fbp + location + 3) = 0;      // No transparency
          }      else  { //assume 16bpp
             unsigned short b = 10;
             unsigned short g = (x-100)/6;     // A little green
             unsigned short r = 31-(y-100)/16;    // A lot of red
             unsigned short t = r<<11 | g << 5 | b;
             *((unsigned short *)(fbp + location)) = t;
          }
       }
    }
```

Framebuffer 测试程序（续2）

```
    munmap(fbp, screensize);
    printf("The framebuffer device was munmapped to memory successfully.\n");
    close(fbfd);
    printf("The framebuffer device was closed successfully.\n");
    return 0;
}
```

Ubuntu下开启终端 framebuffer

- 在引导是修改参数
- 修改 /boot/grub/menu.lst

vga=0x317

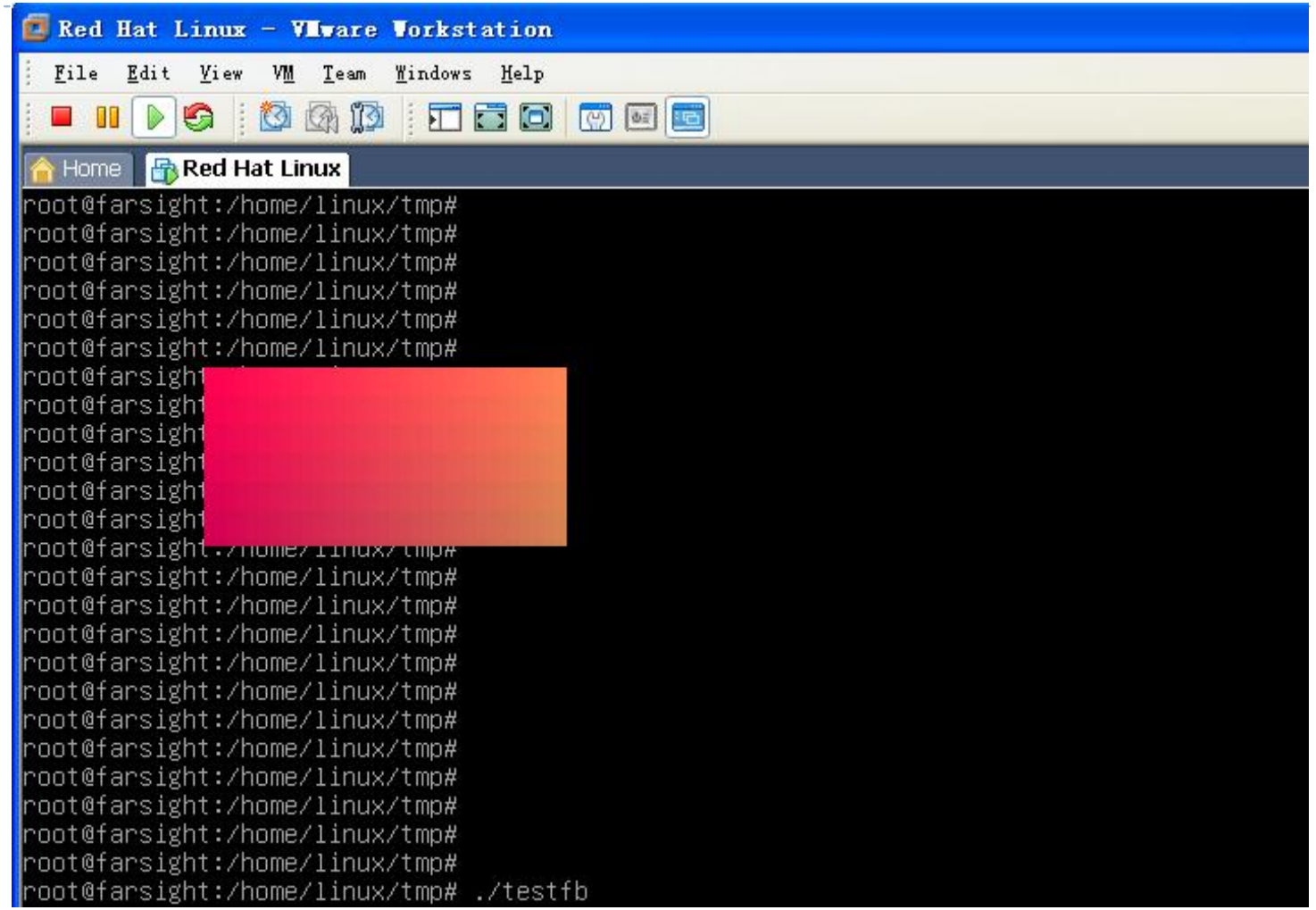
```
125
126 title       Ubuntu, kernel 2.6.20-15-generic
127 root        (hd0,0)
128 kernel      /boot/vmlinuz-2.6.20-15-generic root=UUID=321d44e8-c2de-49cb-a6d1-5aa8a22424da ro quiet splash vga=0x317
129 initrd      /boot/initrd.img-2.6.20-15-generic
130 quiet
131 savedefault
132
133 title       Ubuntu, kernel 2.6.20-15-generic (recovery mode)
134 root        (hd0,0)
135 kernel      /boot/vmlinuz-2.6.20-15-generic root=UUID=321d44e8-c2de-49cb-a6d1-5aa8a22424da ro single
136 initrd      /boot/initrd.img-2.6.20-15-generic
137
138 title       Ubuntu, memtest86+
139 root        (hd0,0)
140 kernel      /boot/memtest86+.bin
141 quiet
```

VESA 模式

Documentation/fb/vesafb.txt

| | 640x480 | 800x600 | 1024x768 | 1280x1024
---+---
256 | 0x301 | 0x303 | 0x305 | 0x307
32k | 0x310 | 0x313 | 0x316 | 0x319
64k | 0x311 | 0x314 | 0x317 | 0x31A
16M | 0x312 | 0x315 | 0x318 | 0x31B

Framebuffer 测试程序显示效果



知识点

- ***帧缓冲(Framebuffer)显示技术概要***
- Framebuffer 原理简介
- 内核对 Framebuffer 的支持（编译配置选项）
- ***Framebuffer 的使用***
  - *操作 framebuffer 程序样例*
  - 基于 Framebuffer的 GUI简介

基于 Framebuffer 的图形系统

- Framebuffer 只是一个提供显示内存和显示芯片寄存器从物理内存映射到进程地址空间中的设备
- 对于应用程序而言，如果希望在 FrameBuffer 之上进行图形编程，还需要自己动手完成其他许多工作
- 有大量的图形系统可供选择和使用
  - 商业
  - 开源

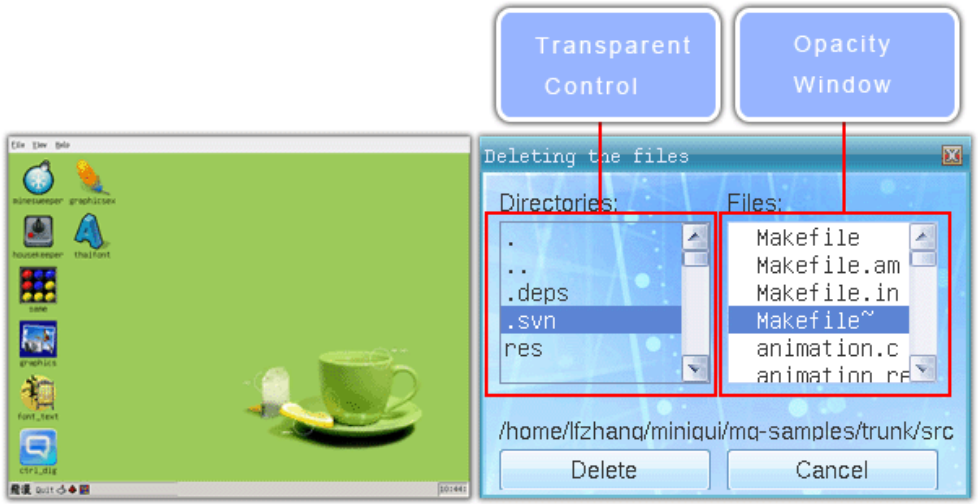
- **Qtopia and Qt/Embedded**
  - **大名鼎鼎，由挪威奇趣科技（*Trolltech*）公司开发，目前被诺基亚收购**
  - **Qtopia PDA，Qtopia Phone**
  - 基于**Qt/Embedded**，包括窗口系统和应用程序套件，包括 **PIM** 应用程序，**Internet**客户端，娱乐和游戏程序，**Qtopia/PDA**可使用商业版权或者开源协议，但 **Qtopia Phone Edition** 只能使用商业版权

Qtopia 控件截图

商业嵌入式 Linux 图形系统软件

- **MiniGUI**
  - 个人认为是国内做得最好的GUI系统
  - 一个双重协议的(GPL或者商业协议)的项目,建立一个小型的窗口系统,支持嵌入式系统和设备.它提供相应Win32API的窗口管理功能.

MiniGUI截屏图片



商业嵌入式Linux图形系统软件

- **Linux PEG**
  - 一个嵌入式GUI库和开发工具
  - 为嵌入式体系设计
- FancyPants
  - 商业图形框架,支持,**skinning,overlays,fancy** 特效。
  - 目标是中量级的消费电子和移动设备比如**POS(Point of Sales/Sevices)**系统,机顶盒,移动电视

开源嵌入式Linux图形系统软件

- X Windows系统 (XFree86或者Xorg)
  - 老牌的图形系统，客户/服务器结构
  - 软件丰富
  - 不适合嵌入式
    - X Window System : 5MB RAM, 16MB disk
    - GNOME : 14MB RAM, 95MB disk
    - KDE : 11MB RAM, 96MB disk
    - Mozilla : 12MB RAM, 26MB disk
- Tiny-X 系统
  - 一个为嵌入式系统设计的小型 **X** 系统。由 **XFree86** 核心开发小组的 **Keith Packard** 开发，**SuSE** 赞助**.**
  - 典型的基于 **Tiny-X** 的 **Xserver** 少于 **1M** 内存（**x86 CPU**）,它已经移植到几个项目上，可以移植到ARM和MIPS等体系结构上

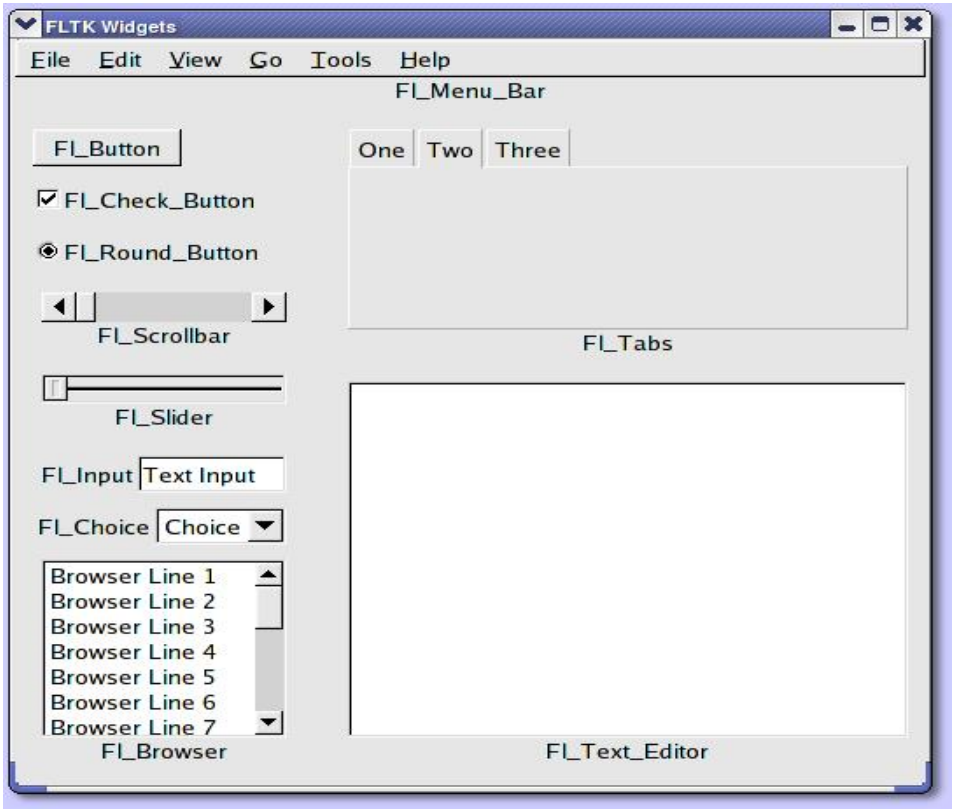
开源嵌入式Linux图形系统软件

- **GtkFB**
  - 从**2.0**开始,**GTK+** 开始发布基于**Linux Framebuffer** 版本, 可用于资源受到限制嵌入式系统，因为即可利用已有的丰富的基于**GTK**的应用程序，又可避免**X Windows**的系统消耗
  - 单任务
- **DirectFB**
  - DirectFB 是一个非常小，但功能强大的程序库，它提供给开发者以图形硬件加速支持,输入设备处理和抽象，集成窗口系统，支持半透明窗口和 **Linux Framebuffer**设备的多重显示层。它是一个完全的硬件抽象层和软件**fallback**为每个图形操作，不支持**underlying** 硬件。
  - 与GTK+一起构成嵌入式图形系统是一个不错的选择
    - broncho 基于Linux+DirectFB+Gtk+的手机系统

Broncho开发样机图片

开源嵌入式Linux图形系统软件

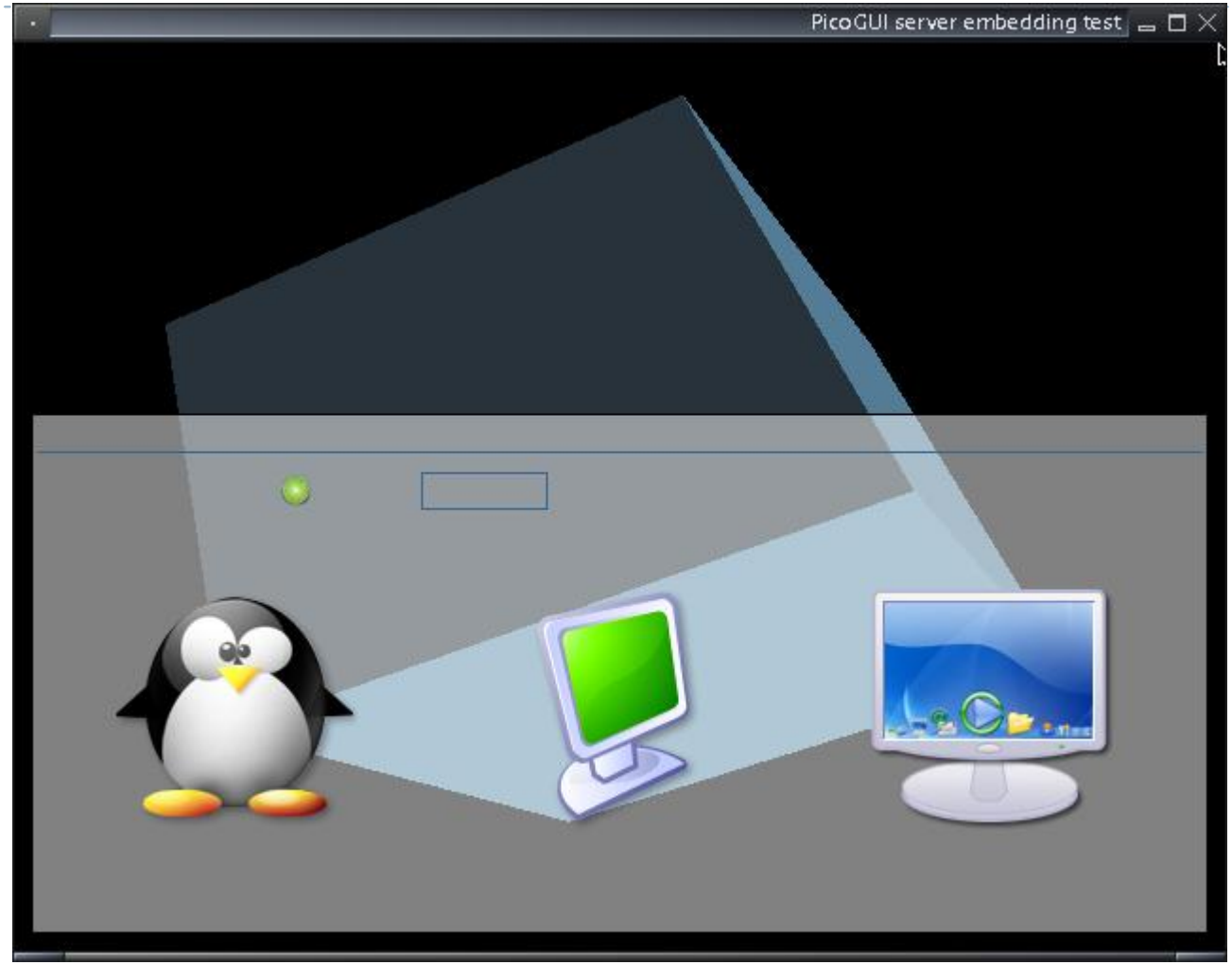
- FLTK
  - **Fast Light Toolkit**
  - **C++ GUI**
  - UNIX/Linux
  - Windows
  - MacOS



开源嵌入式Linux图形系统软件

- **SDL**
  - **Simple DirectMedia Layer** ,是一个非常棒的跨平台的多媒体库
  - **基于SDL的应用和库繁多**
- **PicoGUI**
  - 小巧，可移植的客户/服务器**GUI**设计
  - 工作在许多不同类型的硬件上，包括手持计算机
  - 弹性的客户服务端架构
    - 字体，位图，窗口等放在服务器端
  - 牺牲了灵活性，带来的是速度上的提升和体形的缩小
- **OpenGUI**
  - 一个快速的 **32** 位的高级**C/C++**图形库和窗口库**/GUI**，建立在一个快速的，低级的**x86**汇编语言图形内核之上。提供原始的 **2D** 绘图能力和事件驱动的窗口**API**
- **FBUI**
  - 非常小的**GUI,** 驻留在**linux**内核中
  - 在内核驱动中实现了一个图形系统
  - 包括内核模块和应用层程序库

PicoGUI

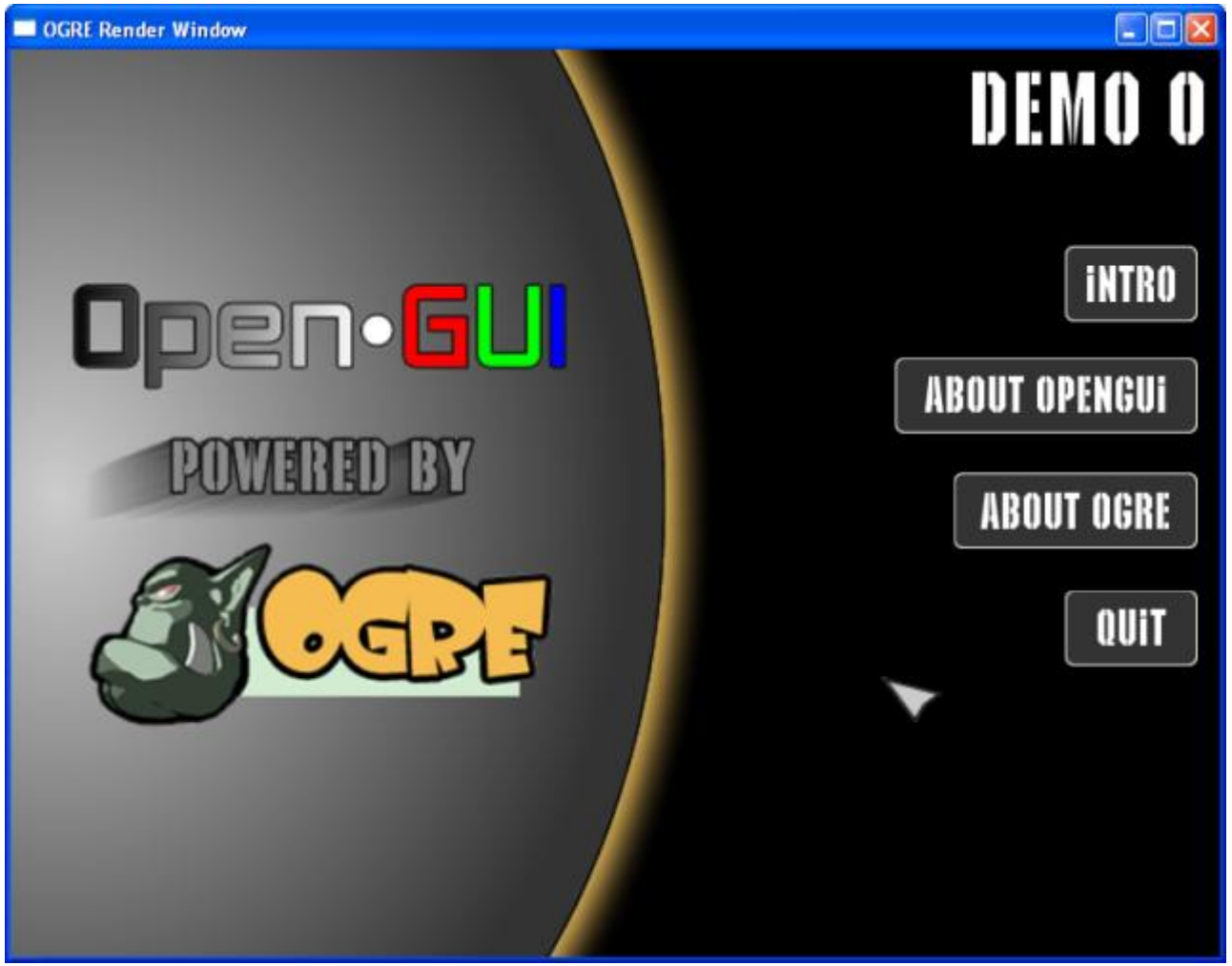


PicoGUI Handinfo hi970

PicoGUI


JetEngine
jetstream
Introduction
Single-player Missions
Multiplayer
About
Mods
JetEngine - JetStream technology demo
10
100

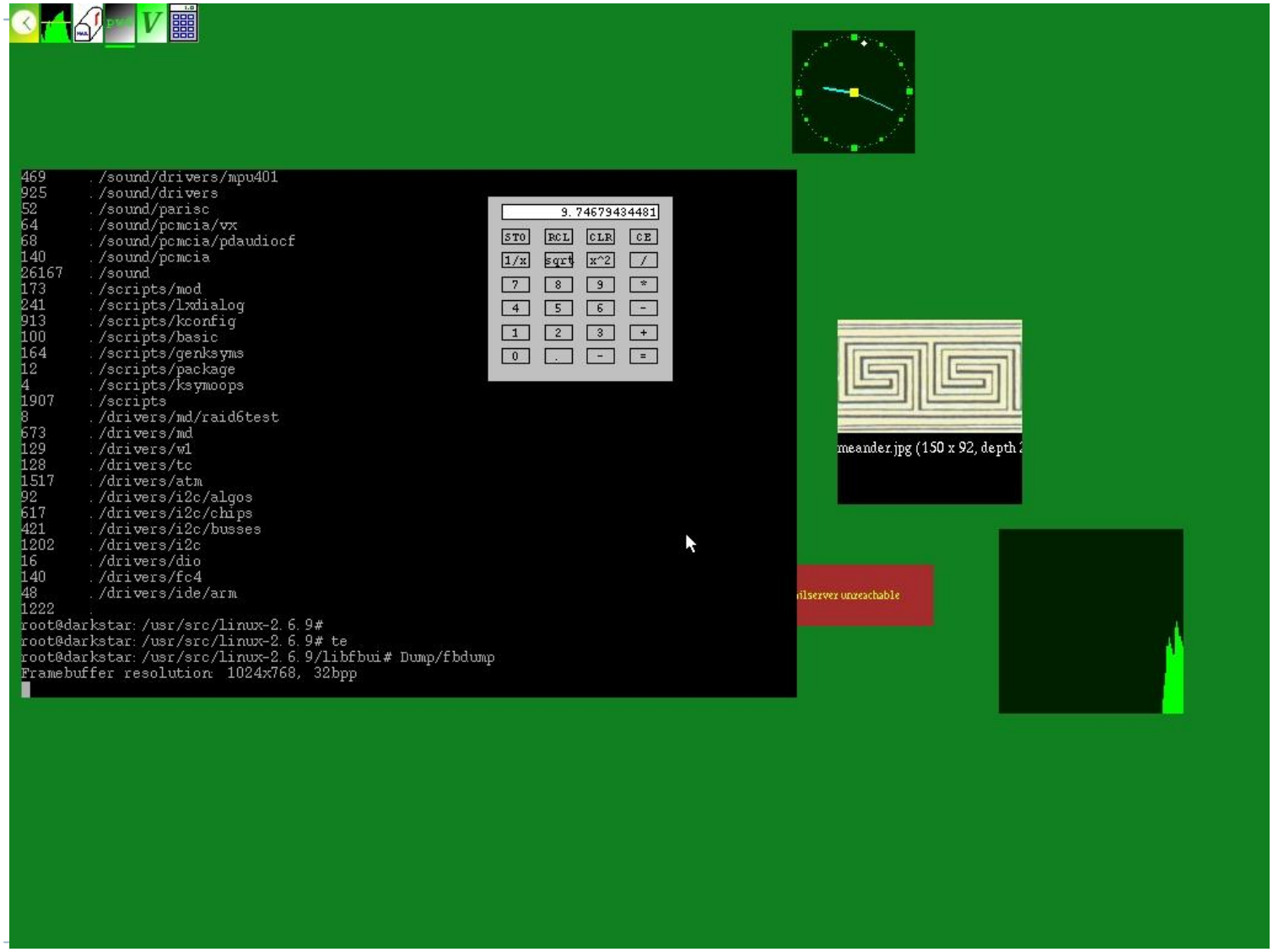
OpenGUI


OGRE Render Window
DEMO 0
Open·GUI
POWERED BY
OGRE
INTRO
ABOUT OPENGUI
ABOUT OGRE
QUIT

fbui


fbview (Viewer/497_1031_3.tif)
To do: 5 items
1. Rehydrate
2. Be tightwad
3. GregPalast.org
4. VoteCobb.org
5. VotersUnite.o
①fbview
②fbscribble
③fbcalc
④fbterm
497_1031_3.tif (689 x 801, depth 24)
Start

fbui

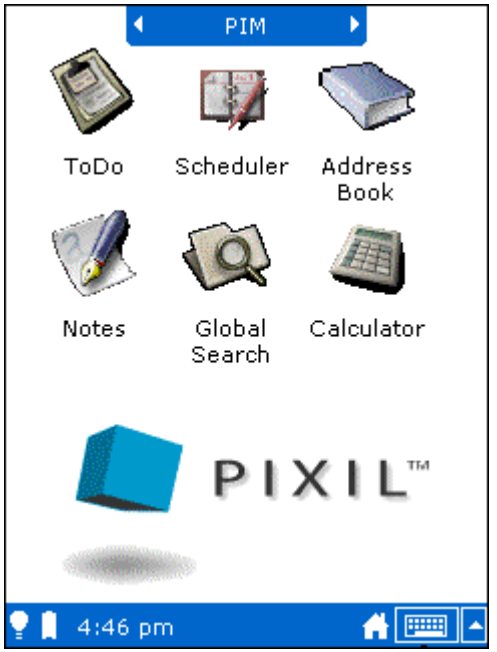

469 ./sound/drivers/mpu401
925 ./sound/drivers
52 ./sound/parisc
64 ./sound/pcmcia/vx
68 ./sound/pcmcia/pdaudiocf
140 ./sound/pcmcia
26167 ./sound
173 ./scripts/mod
241 ./scripts/lxdialog
913 ./scripts/kconfig
100 ./scripts/basic
164 ./scripts/genksyms
12 ./scripts/package
4 ./scripts/ksymoops
1907 ./scripts
8 ./drivers/md/raid6test
673 ./drivers/md
129 ./drivers/w1
128 ./drivers/tc
1517 ./drivers/atm
92 ./drivers/i2c/algos
617 ./drivers/i2c/chips
421 ./drivers/i2c/busses
1202 ./drivers/i2c
16 ./drivers/dio
140 ./drivers/fc4
48 ./drivers/ide/arm
1222 .
root@darkstar: /usr/src/linux-2.6.9#
root@darkstar: /usr/src/linux-2.6.9# te
root@darkstar: /usr/src/linux-2.6.9/libfbui# Dump/fbdump
Framebuffer resolution: 1024x768, 32bpp
9.74679434481
STO RCL CLR CE
1/x sqrt x^2 /
7 8 9 *
4 5 6 -
1 2 3 +
0 . - =
meander.jpg (150 x 92, depth

开源嵌入式Linux图形系统软件

- **Microwindows**
  - 一个开源项目， 适用于于嵌入式系统
  - 跨平台，因为 **MicroSoftWindows** 的缘故 **MicroWindows** 改名为 **Nano-X**
- **OpenGL ES**
  - OpenGL for Embedded Systems 是 OpenGL 三维图形API的子集，针对手机、PDA和游戏主机等嵌入式设备而设计。该API由Khronos集团定义推广，Khronos是一个图形软硬件行业协会，该协会主要关注图形和多媒体方面的开放标准。
  - 是从 OpenGL 裁剪定制而来的
- PIXIL
  - **基于NanoX, fltk**
  - 提供嵌入式在高级因特网中应用程序的应用。它虽然是为商业化准备的，但它提供**GPL**协议下的版本，区别是没有技术支持。

Nano-X

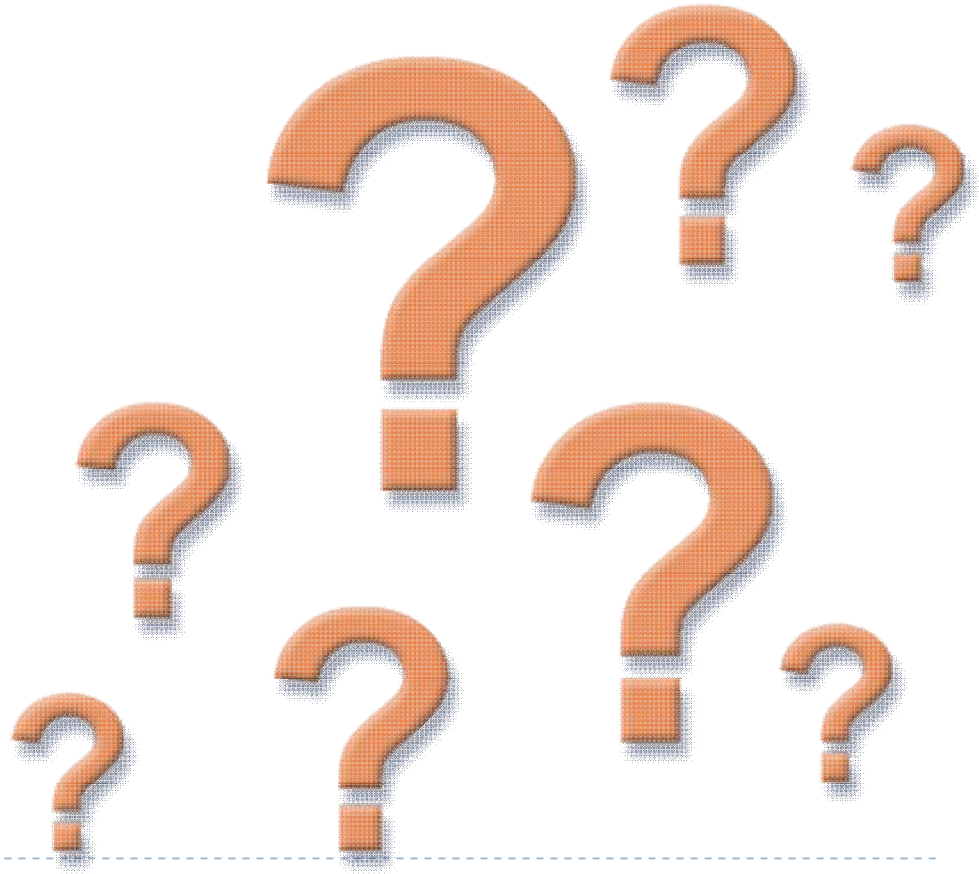

Microwindows
File Edit View Search Go Bookmarks Tasks Help
http://www.netscape.com/
Search
Home My Netscape
Netscape
WebMail Download My Netscape AIM/Chat
Netscape Search
Search
Classifieds DirectoryJob Search Movie ListingsTravel Deals
Decision GuidesGames Maps/DirectionsWhite Pages Yellow Pages
Saturday, October 14, 2000
'Sex' Finale
Sarah Jessica Parker and the girls face roosters, transsexuals and infidelity as the third season of "Sex and the City" comes to a close. Fall TV Special Report
TV Guide
Poll: Who is your favorite 'Sex and the City' character?
Carrie Charlotte Samantha Miranda
Vote
SmartBargains:
50% off FUBU
backpack.
Top Picks
New: Netscape 6
Book Your Flight
Build a Web Site
Consumer Guides
Latest Software
Love@Netscape
Music Search
Online Games
Departments
Autos
Computing
Health
LATEST NEWS
CNN.com Fresh shadows hang over Mideast summit
Movies 'Lost Souls,'
Document: Done
Business Tech Fun Interact


PIM
ToDo
Scheduler
Address Book
Notes
Global Search
Calculator
PIXIL™
4:46 pm

有什么问题吗？

华清远见 FARSIGHT 嵌入式培训专家

Q&A



谢谢！