DRM应用编程流程

深海带鲤鱼 LV.2
2022年09月20日 22:45 · 阅读 40

@[toc] 这一章主要对于drm应用编程的一般流程做一个简单的总结,请勿根据此流程进行实际操作

1.打开设备

DRM驱动会在/dev/dri下创建3个设备节点,用户空间可以使用 `libdrm.so` 提供的库函数来进行drm的应用开发

```
card0
controlD64
renderD128
```

使用:

```
    int fd = open("/dev/dri/card0", O_RDWR | O_CLOEXEC);
    if (fd < 0) {
        ret = -errno;
        fprintf(stderr, "cannot open '%s': %m\n", node);
        return ret;
    }
```

2.检查drm的能力

drm能力可以通过drmGetCap接口获取，用drm_get_cap结构体描述为:

```
/** DRM_IOCTL_GET_CAP ioctl argument type */
struct drm_get_cap {
    __u64 capability;
    __u64 value;
};

int drmGetCap(int fd, uint64_t capability, uint64_t *value)
{
    struct drm_get_cap cap;
    int ret;

    memclear(cap);
    cap.capability = capability;

    ret = drmIoctl(fd, DRM_IOCTL_GET_CAP, &cap);
    if (ret)
        return ret;

    *value = cap.value;
    return 0;
}
```

使用:

```
int drm_check_capability(int fd)
{
        uint64_t flag;

        if (drmGetCap(fd, DRM_CAP_DUMB_BUFFER, &flag) < 0 || !flag) {

  perror("this drm device does not support dumb buffer");
  return -1;

 }

        return 0;

}
```

3.检查Resource

用到的函数为

```
drmModeResPtr drmModeGetResources(int fd)
```

结构为:

```
typedef struct _drmModeRes {

    int count_fbs;
    uint32_t *fbs;

    int count_crtcs;
    uint32_t *crtcs;

    int count_connectors;
    uint32_t *connectors;

    int count_encoders;
    uint32_t *encoders;

    uint32_t min_width, max_width;
    uint32_t min_height, max_height;
} drmModeRes, *drmModeResPtr;
```

实例:

```
res = drmModeGetResources(fd);
if (!res) {
        perror("cannot get drm resources ,err ");
        return -errno;
}
```

4.获取Connector

结构描述:

```
typedef struct _drmModeConnector {
    uint32_t connector_id;
    uint32_t encoder_id; /**< Encoder currently connected to */
    uint32_t connector_type;
    uint32_t connector_type_id;
    drmModeConnection connection;
    uint32_t mmWidth, mmHeight; /**< HxW in millimeters */
    drmModeSubPixel subpixel;

    int count_modes;
    drmModeModeInfoPtr modes;

    int count_props;
    uint32_t *props; /**< List of property ids */
    uint64_t *prop_values; /**< List of property values */

    int count_encoders;
    uint32_t *encoders; /**< List of encoder ids */
} drmModeConnector, *drmModeConnectorPtr;
```

示例:

```
drmModeConnector *conn = drmModeGetConnector(fd, res->connectors[i]);
        if (!conn) {
            fprintf(stderr, "cannot retrieve DRM connector %u:%u (%d): %m\n",
                    i, res->connectors[i], errno);
             s continue;
        }
```

5.获取Enconder

深海带鲤鱼 LV.2
无 @ 无
关注 私信
获得点赞 10
文章被阅读 18,089

4月更文挑战来袭
投稿时间:04/10-05/12
稀土开发者大会
限时领掘金会员
立即前往 >

相关文章
1.MIPI基础-D-PHY层简单介绍
0点赞 · 0评论
Linux设备模型-kobject
0点赞 · 0评论
13.Linux驱动基础-request_firmware升级
0点赞 · 0评论
触摸屏驱动-学习笔记(韦东山)
2点赞 · 0评论
QNX-IPC通信
3点赞 · 0评论

目录
1.打开设备
2.检查drm的能力
3.检查Resource
4.获取Connector
5.获取Enconder
6.创建Framebuffer
7.准备CRTC
8.画图

下一篇
Linux下DRM的基本概念

Enconder结构描述为:

```c
typedef struct _drmModeEncoder {
    uint32_t encoder_id;
    uint32_t encoder_type;
    uint32_t crtc_id;
    uint32_t possible_crtcs;
    uint32_t possible_clones;
} drmModeEncoder, *drmModeEncoderPtr;
```

示例:

```c
if (conn->encoder_id)
{
    drmModeEncoder *enc = drmModeGetEncoder(fd, conn->encoder_id);
}

drmModeEncoderPtr drmModeGetEncoder(int fd, uint32_t encoder_id)
{
    struct drm_mode_get_encoder enc;
    drmModeEncoderPtr r = NULL;

    memclear(enc);
    enc.encoder_id = encoder_id;

    if (drmIoctl(fd, DRM_IOCTL_MODE_GETENCODER, &enc))
        return 0;

    if (!(r = drmMalloc(sizeof(*r))))
        return 0;

    r->encoder_id = enc.encoder_id;
    r->crtc_id = enc.crtc_id;
    r->encoder_type = enc.encoder_type;
    r->possible_crtcs = enc.possible_crtcs;
    r->possible_clones = enc.possible_clones;

    return r;
}
```

6.创建Framebuffer

需要用到的结构体表示为:

```c
/* create a dumb scanout buffer */
struct drm_mode_create_dumb {
 __u32 height;
 __u32 width;
 __u32 bpp;
 __u32 flags;
        /* handle, pitch, size will be returned */
 __u32 handle;
 __u32 pitch;
 __u64 size;
};

/* set up for mmap of a dumb scanout buffer */
struct drm_mode_map_dumb {
        /** Handle for the object being mapped. */
 __u32 handle;
 __u32 pad;
        /**
  * Fake offset to use for subsequent mmap call
  *
  * This is a fixed-size type for 32/64 compatibility.
  */
 __u64 offset;
};

struct drm_mode_destroy_dumb {
 __u32 handle;
};

int drmModeAddFB(int fd, uint32_t width, uint32_t height, uint8_t depth,
   uint8_t bpp, uint32_t pitch, uint32_t bo_handle,
   uint32_t *buf_id)
{
        struct drm_mode_fb_cmd f;
        int ret;

 memclear(f);
 f.width  = width;
 f.height = height;
 f.pitch  = pitch;
 f.bpp    = bpp;
 f.depth  = depth;
 f.handle = bo_handle;

        if ((ret = DRM_IOCTL(fd, DRM_IOCTL_MODE_ADDFB, &f)))
  return ret;

 *buf_id = f.fb_id;
        return 0;
}
```

创建DUMB Buffer

```c
struct drm_mode_create_dumb creq;
uint32_t *fb_id;
rval = drmIoctl(fd, DRM_IOCTL_MODE_CREATE_DUMB, &creq);
        if (rval < 0) {
  printf("%s-->create dumb buffer error:%m\n", __func__);
  return -errno;
```

创建framebuffer对象

```c
/* create framebuffer object for the dumb-buffer */
rval = drmModeAddFB(fd, crep.width,  crep.height, 24, crep.bpp,
      crep.width * bpp/8, crep.fb_handle, &fb_id);
        if (rval < 0) {
  printf("%s-->create framebuffer error: %m\n", __func__);
  return rval;
 }
```

进行mmap

```c
struct drm_mode_map_dumb mreq;
/* 准备内存映射: 获取dumb缓冲区的mmap偏移量 */
 rval = drmIoctl(fd, DRM_IOCTL_MODE_MAP_DUMB, &mreq);
        if (rval) {
  printf("%s-->MODE_MAP_DUMBr error:%m\n", __func__);
  return -errno;
 }
```

进行内存映射:

```c
        /* 内存映射 */
 kms->fmt->vaddr = mmap(0, kms->fmt->size, PROT_READ | PROT_WRITE,
          MAP_SHARED, fd, mreq.offset);

        if (MAP_FAILED == kms->fmt->vaddr) {
  printf("%s-->cannot mmap dumb buffer error:%m\n", __func__);
  return -errno;
 }

        memset(kms->fmt->vaddr, 0, kms->fmt->size);
```

7.准备CRTC

drmModeGetCrtc,drmModeSetCrtc函数

```c
drmModeCrtcPtr drmModeGetCrtc(int fd, uint32_t crtcId)
{
    struct drm_mode_crtc crtc;
    drmModeCrtcPtr r;

    memclear(crtc);
    crtc.crtc_id = crtcId;
```

```c
    if (drmIoctl(fd, DRM_IOCTL_MODE_GETCRTC, &crtc))
        return 0;

    /*
     * return
     */

    if (!(r = drmMalloc(sizeof(*r))))
        return 0;

    r->crtc_id        = crtc.crtc_id;
    r->x              = crtc.x;
    r->y              = crtc.y;
    r->mode_valid     = crtc.mode_valid;
    if (r->mode_valid) {
        memcpy(&r->mode, &crtc.mode, sizeof(struct drm_mode_modeinfo));
        r->width = crtc.mode.hdisplay;
        r->height = crtc.mode.vdisplay;
    }
    r->buffer_id      = crtc.fb_id;
    r->gamma_size     = crtc.gamma_size;
    return r;
}

int drmModeSetCrtc(int fd, uint32_t crtcId, uint32_t bufferId,
           uint32_t x, uint32_t y, uint32_t *connectors, int count,
           drmModeModeInfoPtr mode)
{
    struct drm_mode_crtc crtc;

    memclear(crtc);
    crtc.x            = x;
    crtc.y            = y;
    crtc.crtc_id      = crtcId;
    crtc.fb_id        = bufferId;
    crtc.set_connectors_ptr = VOID2U64(connectors);
    crtc.count_connectors = count;
    if (mode) {
      memcpy(&crtc.mode, mode, sizeof(struct drm_mode_modeinfo));
      crtc.mode_valid = 1;
    }

    return DRM_IOCTL(fd, DRM_IOCTL_MODE_SETCRTC, &crtc);
}
```

示例:

```c
kms.old_crtc = drmModeGetCrtc(fd, kms.crtc_id);

 err = drmModeSetCrtc(fd, kms.crtc_id, kms.fb_id, 0, 0,
     &kms.conn_id, 1, &kms.mode);
        if (err < 0) {

  printf("cannot set CRTC for connector \n");

  return err;
 }
```

8.画图

```c
void drm_draw_background(void)
{
        int i, k;

        int color[] = {0xff0000, 0xffff00, 0x00ff00}; //红,黄,绿

        for (k = 0; k < 3; k++) {

  for (i = 0; i < (kms.fb_size / 4); i++)
   kms.fb_map[i] = color[k];

  sleep(2);
 }

}
```

分类: 后端 标签: 后端

文章被收录于专栏:

Display
显示屏基础知识

订阅专

安装掘金浏览器插件
多内容聚合浏览、多引擎快捷搜索、多工具便捷提效、多模式随心畅享，你想要的，这里都有！

前往安装

相关小册

Netty 网络编程之道
肖恩Sean 小程序 416购买
¥19.95 ¥39.9 首单券后价

程序员的必修课
奔波儿灞取... 小程序 1233购买
¥24.95 ¥49.9 首单券后价

评论

看完啦，登录 分享一下感受吧～

相关推荐

AntBlack | 24分钟前 | 后端
ElasticSearch：FST 数据结构
37 点赞 评论

Pokeya | 1天前 | 后端 Go
一起 Go 编程 | Go 的面向对象编程
761 1 评论

小靳x | 1天前 | 后端 Spring Cloud Spring Boot
Spring Cloud Stream：打造强大的微服务事件驱动架构
793 2 评论

码下客 | 1天前 | 后端
分治思想之ForkJoin
708 点赞 评论

一无是处的研究僧 | 1天前 | 后端 Python Linux
深入理解python虚拟机：调试器实现原理与源码分析
447 2 评论

沐华 | 7天前 | 前端 后端 程序员
我是怎么在掘金找到老婆的
2.6w 1007 970

优弧 | 23小时前 | 前端 后端 人工智能
关于规范 AIGC 能力进行辅助创作的公告
914 9 11

趣开源 | 1天前 | 后端 面试 Java

springboot中nacos-client是怎么获取配置的？

342 3 1

转转技术团队 | 1天前 | 后端 监控

支付通道监控系统的搭建

261 2 评论

京东云开发者 | 1天前 | Java JVM 后端

从原理聊JVM(三)：详解现代垃圾回收器Shenandoah和ZGC | 京东云技术团队

262 2 评论

匹多莫德的传说 | 6小时前 | 后端

Spring 定义声明式 HTTP 服务 初探

188 1 评论

Anoxia1 | 5天前 | 后端 Spring

07 实现bean初始化、摧毁方法的注入

193 点赞 评论

Bug终结者_ | 7小时前 | 前端 后端 JavaScript

Spring Boot 整合Dubbo + Zookeeper 实现分布式 消费者与服务者的业务调用

188 1 评论

Anoxia1 | 6天前 | Spring Boot 后端

1、SpringBoot-Start组件开发之记录接口日志信息

195 点赞 评论

Neoooo | 12小时前 | 后端

一、Java反射机制&自定义注解

92 1 2

Anoxia1 | 5天前 | 后端 Spring

08 实现Aware接口，我们能用他干点什么！

188 点赞 评论

Samson_bu | 7小时前 | 后端 Spring Spring Boot

Spring Boot「48」扩展：深入分析 IoC 容器

190 1 评论

wayne214 | 1天前 | 后端 IntelliJ IDEA

推荐一款IDEA神级插件【Bito】而且免费！

2354 30 17

一只爱捣猫的程序猿 | 21小时前 | 后端

什么是策略模式？

85 1 评论

老王随聊 | 15小时前 | 后端 Java

如何快速构建一个SpringBoot项目

140 点赞 评论

稀土掘金浏览器插件——你的一站式工作台

多内容聚合浏览、多引擎快捷搜索、多工具便捷提效、多模式随心畅享，你想要的，这里都有。

安装浏览器插件