

**日志输出管理详细设计文档**

**V1.0**

文档修订记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 修订内容简述 | 修订人 | 修订日期 | 修订后版本号 |
| 1 | 创建 | 张建新 |  | 1.00 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1. 概述 3](#_Toc530383456)

[1.1 背景 3](#_Toc530383457)

[1.2 目标 3](#_Toc530383458)

[2. 详细功能设计 3](#_Toc530383459)

[2.1 总体说明 3](#_Toc530383460)

[2.2 OSA管理注册及输出接口 5](#_Toc530383461)

[2.3 OSA查询和配置接口 6](#_Toc530383462)

[2.4 OSA输出统计无锁设计 6](#_Toc530383463)

[3. 详细结构及接口定义 7](#_Toc530383464)

[3.1 结构定义 7](#_Toc530383465)

[3.2 接口设计 7](#_Toc530383466)

[3.2.1 OSA注册和注销接口 7](#_Toc530383467)

[3.2.2 OSA单个模块/库查询和管理接口 8](#_Toc530383468)

[3.2.2 获取OSA所管理所有模块/库信息 11](#_Toc530383469)

[3.2.3 OSA输出接口设计 12](#_Toc530383470)

[4. 移植说明 14](#_Toc530383471)

[4.1 移植指南 14](#_Toc530383472)

[4.1 移植自测 15](#_Toc530383473)

[5. 总结 16](#_Toc530383474)

[6. 参考文档 16](#_Toc530383475)

# 1. 概述

## 1.1 背景

后端设备系统及底层库日志输出管控较为欠缺，设备重要敏感信息对外进行展现，加大设备被破解的风险。另外日志不受控制的输出，导致部分日志出现刷屏现象，这一方面导致系统性能严重下降，另一方面导致重要日志信息被覆盖或者输出不全，加大排查难度，另外也不便于公司日志存储管理。基于这一背景，应产品线日志管理的需求，开发可分类、可管理的日志输出模块。

## 1.2 目标

设计可分类和可管理的日志输出模块，可对各个引用模块单独设置日志输出等级，实现对模块/库的输出灵活管理，达到重点监控日志的功能。

# 2. 详细功能设计

## 2.1 总体说明

本设计主要针对系统模块及库提供统一日志输出管理，其整体框架如2.1所示，其中OSA模块负责对公司开发的驱动模块进行日志输出管理，OSA库负责对上层系统库提供日志管理输出，其他业务库的输出管理通过infra库实现。OSA库即对依赖其的系统库提供统一的打印接口，且对业务模块提供其所管理的系统模块和库日志管理接口，业务管理接口可以对OSA库的日志管理进行控制和查询信息等，后期可考虑扩展至对内核的日志也进行相关的管控和信息获取。OSA所实现的基本打印格式为：

<COIOR>[Module-name@SVN-VERSION]:[FUNC|LINE] [MSG INFO]

其中COLOR为日志输出颜色；

[Module-name@SVN-VERSION] 分别为模块名称以及模块的SVN版本号；

[FUNC|LINE] 为日志输出所在函数及行号，不是所有打印接口都有；

[MSG INFO] 为真实日志内容。



图2.1 OSA模块整体框图

需要依赖OSA(包括osa.ko和libosa.a,后续统一称作OSA)进行开发的模块/库必须在编译脚本需要引入宏OSA\_MODULE\_NAME和SVN\_VERSION，OSA\_MODULE\_NAME定义模块名称，该名称也作为日志管理接收的名称，后续作为日志管理的KEY值。SVN\_VERSION为模块的SVN版本号，便于后续问题定位。例如./build/rules.mk添加如下：

SVN\_VERSION := $(shell sed -n "11p" $(rootdir)/.svn/entries)

OSA\_MODULE\_NAME = XXX@(SVN\_VERSION) ;xxx为模块名

CFLAGS += -DOSA\_MODULE\_NAME='"$(OSA\_MODULE\_NAME)"'

EXTRA\_CFLAGS += -D'OSA\_MODULE\_NAME=\"$(OSA\_MODULE\_NAME)\"'

这里之所将SVN版本号和并到模块名中，是因为库的头文件兼容问题，必须注意是OSA日志管理模块中只将@符前面字符串作为KEY值。且@符为必须的，OSA统计时根据@符提出去调用OSA\_INFO函数模块名。通过编译引入模块名称和库的SVN版本号，防止依赖模块/库未定义相关的信息。例如：

SVN\_NUM := 1111

OSA\_MODULE\_NAME:= OSA\_TESTU@$(SVN\_NUM)

CFLAGS += -DOSA\_MODULE\_NAME='"$(OSA\_MODULE\_NAME)"'

OSA管理库中实际使用的模块名为”OSA\_TESTU”。

## 2.2 OSA管理注册及输出接口

OSA使用统一的管理结构OSA\_LogManager对象对依赖其的模块/库进行统一管理，因此在依赖其的模块和库初始化阶段必须注册一个OSA\_LogManager对象到OSA。因而OSA整体调用流程如图2.2所示。每个模块/库调用注册函数后,OSA将它的结构对象链入到全局管理链表OSA\_LogManager\_g中，由该链表进行统一管理，考虑到避免因引入模块输出管理导致性能消耗，建议使用无锁设计，可考虑增加引用计数，即对调用注销函数不再使用的模块/库将OSA\_LogManager对象中的引用计数设置为0，而不对其结构对象的内存进行释放。当该模块再次调用注册函数开始使用时，复用前面已分配的对象内存。OSA提供类似Linux内核的分等级打印接口，例如OSA\_INFO/OSA\_ERR等。为了减轻代码整改的难度，OSA设计兼容已有的打印接口，依赖OSA的库无需对其打印代码进行整改，由接口实现进行向后兼容处理。即接口中对未能找到匹配OSA\_LogManager对象的模块/库，OSA采用默认输出的方式进行兼容。模块和库的OSA\_LogManager对象进行独立分开管理。为了实现管理结构对象内存复用以及减少锁的开销，管理结构的注册、注销、查找操作均是无锁操作。对于外部注册管理结构，OSA日志管理模块中为其分配管理对象副本，并将该新分配的副本注册到全局链表中，后续该模块所有管理操作也是对副本进行相应操作。调用者自行注销和释放其注册结构对象。



图2.2模块/库注册和输出流程图

但必须注意，模块或者库中必须根据输出信息重要性对所调用的接口进行整改更新，例如系统库对GLIBC接口/系统调用的错误，必须使用OSA\_ERR接口进行输出，考虑到部分libdvr接口编码不规范，导致对接口调用出错时的errno进行处理，导致后期排查造成极大的困扰，因此建议增加OSA\_ERRNO接口的输出部分，该接口实现中增加strerror(errno)部分的信息输出，即对于调用OSA\_ERRNO进行的输出格式为：

<COIOR>[Module-name@SVN-VERSION]:[FUNC|LINE] [MSG INFO] [ERROR INFO ]

方便后期问题的定位。且增加模块输出级别管控后，更应该规范使用OSA\_XXX系列接口的调用，真正做到可按打印级别输出。

若库的开发和维护者不想通过额外添加注册接口进行注册，可考虑将注册动作集成到OSA\_INFO/OSA\_ERR等输出接口当中，在输出接口中根据模块名没有找到匹配项时，主动注册一个默认模块管理结构对象。

## 2.3 OSA查询和配置接口

OSA的查询和配置接口统一对依赖的模块和库进行管理，在OSA模块中提供设备节点供OSA进行查询和配置，查询和配置流程如图2.3所示。



图2.3 模块/库配置和查询流程图

OSA查询和配置接口首先在OSA库中全局管理列表中查询匹配项，若找到则直接进行相关的配置和信息返回，否则则通过IOCTL命令进入到OSA模块进行同样的处理。

### 2.4 OSA输出统计无锁设计

OSA驱动模块打印量统计计数可用原子变量定义，而无需使用加锁操作，但是OSA库中对打印量统计技术的管理似乎需要加锁保护，GCC有内建原子操作函数，但目前主要用于自加自减，而且也是通过指令组合保证原子性。针对用户态的打印量统计，先暂时考虑加锁，后续考虑无锁设计。前期加锁设计需要评审对性能的影响情况。

# 3. 详细结构及接口定义

## 3.1 结构定义

OSA通过OSA\_LogManager结构对象来管理各个依赖模块及库，为了便于管理抽象出内核态和用户态OSA打印管理模块通用结构化信息定义，当调用者尝试获取OSA管理模块信息时，OSA打印管理返回其所管理所有模块/库的OSA\_LogManager\_Common\_t,结构信息。

typedef struct OSA\_LogManagerCommon\_t {

/\* 模块的名称，取模块OSA\_MODULE\_NAME定义@字符前部分\*/

Char osa\_Name[OSA\_LOGENTITY\_NAMELEN];

Uint8 osa\_LogLevel; /\* 模块内部打印等级 \*/

Uint8 osa\_PrintEnable; /\* 模块打印使能，可设置关闭整个模块打印 \*/

Uint8 osa\_Stat; /\* 模块打印量统计功能开关 \*/

Int8 osa\_Refcnt; /\* 模块日志管理引用计数 \*/

}OSA\_LogManagerCommon;

typedef struct OSA\_LogManager\_t {

OSA\_LogManagerCommon info;

OSA\_ListHead node;

#ifdef \_\_KERNEL\_\_

atomic\_t osa\_PrintBytes; /\* 内核模块打印统计计数 \*/

#else

Uint64 osa\_PrintBytes; /\* 用户态库打印统计计数 \*/

OSA\_MutexHandle hMutex;

#endif

Uint8 reserve[OSA\_LOGMANAGER\_EXTEND]; /\* 后期扩展 \*/

}OSA\_LogManager;

OSA库和OSA驱动模块使用独立的全局变量OSA\_LogManager\_g管理链表。

## 3.2 接口设计

### 3.2.1 OSA注册和注销接口

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : OSA\_logManagerRegister

\* 描 述 : 该函数向OSA注册模块日志管理结构，通常在模块/库在初始化时调用。

\* 对未向OSA注册之前，所有模块打印默认关闭。

\* 当模块启动调用本接口向OSA注册管理结构后，

\* 所有的日志打印都通过日志管理模块的接口进行管理。

\*

\* 输 入 : OSA\_LogManager: 待注册的日志管理结构对象

\*

\* 输 出 : 无。

\* 返回值 : OSA\_SOK: 成功

\* OSA\_EFAIL: 失败

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Int32 OSA\_logManagerRegister(OSA\_LogManager \*pLogManager)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : OSA\_logManagerDeRegister

\* 描 述 : 该函数向OSA注销模块日志管理结构，只会在模块注销时调用。

\* 当模块被注销时，不对管理结构内存进行释放，只是对引用计数进行处理

\* 重新加载模块后，可按同名匹配后复用原来分配的内存对象。

\*

\* 输 入 : OSA\_LogManager: 待注册的日志管理结构对象

\*

\* 输 出 : 无。

\* 返回值 : OSA\_SOK: 成功

\* OSA\_EFAIL: 失败

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Int32 OSA\_logManagerDeRegister(OSA\_LogManager \*pLogManager)

注册动作可集成到打印输出接口中自动完成注册。

### 3.2.2 OSA单个模块/库查询和管理接口

配置请求结构定义

#define OSA\_REQ\_MAXLEN 32

typedef enum {

OSA\_LOGMANAGER\_CMD\_SETENABLE, /\*使能或者关闭模块日志输出\*/

OSA\_LOGMANAGER\_CMD\_GETENABLE, /\*获取模块/库日志输出使能\*/

OSA\_LOGMANAGER\_CMD\_SETLEVEL, /\*设置模块/库内部打印等级\*/

OSA\_LOGMANAGER\_CMD\_GETLEVEL, /\*获取模块/库内部打印等级\*/

OSA\_LOGMANAGER\_CMD\_GETPRINTS, /\*获取模块/库打印输出计数\*/

OSA\_LOGMANAGER\_CMD\_MAX

} OSA\_LogManagerCmd;

为了保持与OSA库的打印等级兼容，保留OSA原打印等级的定义如下：

typedef enum

{

OSA\_LOG\_LV\_ERR = 0, /\*错误打印\*/

OSA\_LOG\_LV\_WARN, /\*警告打印\*/

OSA\_LOG\_LV\_INFO, /\*通告打印\*/

OSA\_LOG\_LV\_DBG, /\*调试打印\*/

OSA\_LOG\_LV\_MAX

} OSA\_LogLevel;

typedef enum {

OSA\_LOGMANAGER\_DISABLE, /\*关闭模块日志输出 \*/

OSA\_LOGMANAGER\_ENABLE, /\* 使能模块日志输出 \*/

OSA\_LOGMANAGER\_ENABLE\_MAX

} OSA\_LogManangerEnable;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : OSA\_LogManagerSetEntryPrintLevel

\* 描 述 : 该函数向OSA管理的对应模块/库设置其打印等级

\* 无锁设计，调用者需要保证保证配置安全

\*

\* 输 入 : mod\_name,: 待操作模块名称

\* 输 入 : level: 打印等级

\* 输 出 :

\* 返回值 : OSA\_SOK: 成功

\* OSA\_EFAIL: 失败

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Int32 OSA\_LogManagerSetEntryPrintLevel(const Char \*modName, Int32 level)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : OSA\_LogManagerGetEntryPrintLevel

\* 描 述 : 该函数获取OSA管理的对应模块/库的打印等级

\* 无锁设计，调用者需要保证保证配置安全

\*

\* 输 入 : mod\_name,: 待操作模块名称

\* 输 入 : level: 返回模块打印等级

\* 输 出 :

\* 返回值 : OSA\_SOK: 成功

\* OSA\_EFAIL: 失败

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Int32 OSA\_LogManagerGetEntryPrintLevel(const Char \*modName, Int32 \*level)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : OSA\_LogManagerSetEntryEnable

\* 描 述 : 该函数设置OSA管理的对应模块/库打印使能或关闭

\* 无锁设计，调用者需要保证保证配置安全

\*

\* 输 入 : mod\_name,: 待操作模块名称

\* 输 入 : status: 使能或关闭模块/库的打印输出

\* 输 出 :

\* 返回值 : OSA\_SOK: 成功

\* OSA\_EFAIL: 失败

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Int32 OSA\_LogManagerSetEntryEnable(const Char \*modName,

OSA\_LogManangerEnable status)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : OSA\_LogManagerGetEntryPrintStatics

\* 描 述 : 该函数获取对应模块/库相关打印统计量

\* 用户态统计计数加锁保护，内核态使用原子变量

\* 当前主要可获取模块/库的打印统计计数

\*

\* 输 入 : mod\_name,: 待操作模块名称

\* 输 出 : printedBytes: 返回模块/库打印计数

\* 返回值 : OSA\_SOK: 成功

\* OSA\_EFAIL: 失败

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Int32 OSA\_LogManagerGetEntryPrintStatics(const Char \*modName, Uint64 \*printedBytes)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : OSA\_LogManagerGetEntryInfo

\* 描 述 : 该函数获取OSA所管理的单个模块/库信息

\*

\* 输 入 : pentry 填入需要获取信息模块/库的名称

\* 输 出 : pentry 返回指定模块/库的信息,不包括该模块打印统计量

\* 返回值 : OSA\_SOK: 成功

\* OSA\_EFAIL: 失败

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Int32 OSA\_LogManagerGetEntryInfo(OSA\_LogManagerCommon \*pentry)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : OSA\_LogManagerSetEntryInfo

\* 描 述 : 该函数设置OSA所管理的单个模块/库管理功能

\*

\* 输 入 : pentry 设置OSA管理功能选项

\* 输 出 : 无

\* 返回值 : OSA\_SOK: 成功

\* OSA\_EFAIL: 失败

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Int32 OSA\_LogManagerSetEntryInfo(OSA\_LogManagerCommon \*pentry)

### 3.2.2 获取OSA所管理所有模块/库信息

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : OSA\_LogManagerGetAllEntryInfoSize

\* 描 述 : 该函数获取OSA所管理的所有模块/库管理结构占用内存空间

\* 目的在于便于调用者获取管理结构信息时，分配足够内存空间

\*

\* 输 入 : 无

\* 输 出 : 无

\* 返回值 : OSA所管理所有模块/库核心结构信息占用空间

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Int32 OSA\_LogManagerGetAllEntryInfoSize(void)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : OSA\_LogManagerGetAllEntryInfo

\* 描 述 : 该函数获取OSA所管理的所有模块/库当前所总输出打印量

\*

\*

\* 输 入 : size 为buf的大小

\* 输 出 : buf 用于返回OSA所管理模块/库的数据结构化信息，buf必须足够可容纳

: 否则将获取失败，返回获取信息所需要缓存大小

\* 返回值 : OSA\_SOK: 成功

\* OSA\_EFAIL: 失败

> 0的值表示参数buf所提供的空间不能满足返回结构化信息所需空间，此时

返回所需的空间

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Int32 OSA\_LogManagerGetAllEntryInfo(OSA\_LogManagerCommon \*pentry, size\_t size)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : OSA\_LogManagerDisableAllEntryStat

\* 描 述 : 该函数关闭/打开OSA所管理的所有模块/库输出打印量统计功能

\* 用户态打印统计功能基于锁的性能开销，增加该接口关闭统计功能

\*

\* 输 入 : 无

\* 返回值 : OSA\_SOK: 成功

\* OSA\_EFAIL: 失败

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Int32 OSA\_LogManagerDisableAllEntryStat(OSA\_LogManangerEnable status);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : OSA\_LogManagerUserInit

\* 描 述 : 该函数为OSA管理库的初始化接口

\*

\* 输 入 : 无

\* 输 出 : 无

\* 返回值 : OSA\_SOK: 成功

\* OSA\_EFAIL: 失败

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Int32 OSA\_LogManagerUserInit(void)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : OSA\_LogManagerUserExit

\* 描 述 : 该函数为OSA管理功能注销接口

\* 该接口将所有用户应用库所有模块注销

\*

\* 输 入 : 无

\* 输 出 : 无

\* 返回值 : OSA\_SOK: 成功

\* OSA\_EFAIL: 失败

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Int32 OSA\_LogManagerUserExit(void)

### 3.2.3 OSA输出接口设计

OSA打印输出接口完全与原接口定义完全兼容且一致，原则上调用者无需整改，但为了便于统一打印输出管理，尽量修改使用对应接口。若基于OSA模块开发的模块或者库，没有调用OSA\_logManagerRegister注册模块的管理接口，在以下打印接口中，会主动注册一个默认打印等级的管理结构。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : OSA\_ERROR

\* 描 述 : 该函数获取OSA所管理的所有模块/库当前所总输出打印量

\*

\* 输 入 :无

\* 输 出 :无

\* 返回值 : 返回所有模块/库当前总输出打印量

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void OSA\_ERROR(const char \*fmt, …)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : OSA\_WARN

\* 描 述 : 该函数获取OSA所管理的所有模块/库当前所总输出打印量

\*

\* 输 入 :无

\* 输 出 :无

\* 返回值 : 返回所有模块/库当前总输出打印量

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void OSA\_WARN(const char \*fmt, …)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : OSA\_ DGB

\* 描 述 : 该函数获取OSA所管理的所有模块/库当前所总输出打印量

\*

\* 输 入 :无

\* 输 出 :无

\* 返回值 : 返回所有模块/库当前总输出打印量

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void OSA\_DGB (const char \*fmt, …)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 函数名 : OSA\_INFO

\* 描 述 : 该函数获取OSA所管理的所有模块/库当前所总输出打印量

\*

\* 输 入 :无

\* 输 出 :无

\* 返回值 : 返回所有模块/库当前总输出打印量

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void OSA\_INFO(const char \*fmt, …)

输出接口伪代码：

OSA\_INFO

以'@'为分隔符从fmt中提取出调用结构模块名，若无”@”则默认直接输出

pbuf = strchr(pFmt, '@');

//根据抽取出来的模块名查找对应模块管理对象

OSA\_LogManager\_p osa\_entry = NULL;

osa\_entry = OSA\_LogManagerGetbyName(mod\_name)

//若没有找到管理对象，则直接输出，若找到管理对象，则根据管理对象的

//配置决定是否输出

if (!osa\_entry || (osa\_entry && osa\_entry->info.osa\_refcnt > 0 &&

osa\_entry->info.osa\_PrintEnable &&

osa\_entry->info.osa\_LogLevel >= level)) {

va\_start(args, fmt);

length = gLogOps.OpWrite(level, color, pFmt, args);

//若满足打印要求，且使能统计功能，则针对不同场景进行统计累加

if (osa\_entry &&

(OSA\_LOGMANAGER\_ENABLE == osa\_entry->info.osa\_stat)) {

#if defined(\_\_KERNEL\_\_)

atomic\_add(length, &osa\_entry->osa\_PrintBytes);

#else

OSA\_mutexLock(osa\_entry->hMutex);

osa\_entry->osa\_PrintBytes += length;

OSA\_mutexUnlock(osa\_entry->hMutex);

#endif

}

# 4. 移植说明

## 4.1 移植指南

相关库或者驱动模块需要支持打印管理，需要按以下要求进行整改：

1、在库的编译脚本中定义符合2.1章所要求的OSA\_MODULE\_NAME宏，并添加到编译选项中：

CFLAGS += -DOSA\_MODULE\_NAME='"$(OSA\_MODULE\_NAME)"'

可自行调用OSA\_logManagerRegister注册打印管理对象，不自行注册的模块，在首次调用OSA提供打印接口时，OSA打印管理功能块自动为库注册默认模块管理对象，注册的默认打印为OSA\_LOG\_LV\_INFO，默认打印统计功能为关闭状态。

2、设备集成OSA库到如下SVN路径及版本

svn 路径：<http://10.6.5.2/svn/SysBaseline/Libs/StorageOSA/Trunk>

r35147 | 27855 | 2018-11-13 14:15:08 +0800 (Tue, 13 Nov 2018) | 7 lines

【需求(问题)描述】提交测试用例删除不必要文件

【设计(修改)方法】上传基于Hisi3559A的测试示例

【影响范围】 无

【测试重点】 OSA Log管理接口

【对应TD】 无

【修改人】 27855

【评审人】 蔡龙之、周阳2

使用OSA头文件接口的库需要固化升级到OSA库的55158版本，即固化如下库及版本

<http://10.6.5.2/svn/Camera_api/IPCSysHeader/Trunk/common@55158> common

3、按照程序逻辑规范调用OSA\_XXX打印接口，避免不合理接口调用。

4、需要调用OSA库的管理接口时需先调用初始化和注销接口：

Int32 OSA\_LogManagerUserInit(void)

Int32 OSA\_LogManagerUserExit(void)

在正常设备中只需在设备启动和结束中调用一次。

## 4.1 移植自测

整改后的库需验证的库可参考OSA库中上传的测试用例(测试用例在库的Trunk/test/osalog\_test目录下)，自行集成相关用例测试，自测验证项可包括以下几类：

1. 打印是否符合预定格式，通常预期格式如下：

[OSA\_TESTU@1111] OSA\_logManagerDeRegister Finish；

1. 验证指定的打印等级下，对应低等级OSA\_XXX接口是否有输出打印；
2. 调用接口使能打印统计功能，验证统计量是否满足需求；
3. 调用相关接口确认禁止模块打印输出后，是否仍有输出；

其他验证可参考OSA打印自测用例中的Usage说明：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

usage:

./osaLogCtl GLevel modName

./osaLogCtl SLevel modName newLevel

./osaLogCtl GEnable modName

./osaLogCtl SEnable modName newEnable

./osaLogCtl GStat modName

./osaLogCtl SStat modName newStat

./osaLogCtl GetAll

./osaLogCtl RoundTest rounds modName.

/osaLogCtl Register round modName

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 5. 总结

# 6. 参考文档