**异步日志模块ALOG**

**设计文档**

第1.0版

**银联数据服务有限公司**

**2021年5月**

目录

[1. 前言 4](#_Toc72509956)

[2. 全局配置 5](#_Toc72509957)

[3. 自定义配置 6](#_Toc72509958)

[列1注册名 6](#_Toc72509959)

[列2日志级别 6](#_Toc72509960)

[列3 单个文件最大尺寸 6](#_Toc72509961)

[列4 打印格式 6](#_Toc72509962)

[列5 当前日志文件名模式 7](#_Toc72509963)

[列6 备份日志文件名模式 7](#_Toc72509964)

[列7 退出强制备份开关 7](#_Toc72509965)

[举例 7](#_Toc72509966)

[4. 接口和宏定义 8](#_Toc72509967)

[g\_alog\_ctx 8](#_Toc72509968)

[alog\_initContext 8](#_Toc72509969)

[alog\_close 8](#_Toc72509970)

[alog\_writelog\_t 8](#_Toc72509971)

[logtype 日志类型 9](#_Toc72509972)

[level 日志级别 9](#_Toc72509973)

[regname 注册名 9](#_Toc72509974)

[cstname 标识名 9](#_Toc72509975)

[modname 模块名 9](#_Toc72509976)

[file文件名 10](#_Toc72509977)

[lineno 文件行数 10](#_Toc72509978)

[buf 输入缓冲区 10](#_Toc72509979)

[len 输入缓冲区长度 10](#_Toc72509980)

[fmt,… 格式化输出模式及可变参数列表 10](#_Toc72509981)

[宏定义 10](#_Toc72509982)

[5. 编译和调用方法 12](#_Toc72509983)

[ALOG编译 12](#_Toc72509984)

[调用方编译 12](#_Toc72509985)

[共享内存操作 12](#_Toc72509986)

[6. 设计原理说明 13](#_Toc72509987)

[数据结构 13](#_Toc72509988)

[线程逻辑 15](#_Toc72509989)

**文档编写版本维护记录**

| 版本 | 日期 | 拟稿/修改部门 | 拟稿/修改人 | 审核人 | 说明 |
| --- | --- | --- | --- | --- | --- |
| V1.0 | 20210521 | 软开 | 王昊 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

注：编写文档时的版本维护信息

|  |
| --- |
| 本文档中的所有内容为银联数据服务有限公司专属所有。未经银联数据服务有限公司的明确书面许可，任何组织或个人不得以任何目的、任何形式及任何手段复制或传播本文档部分或全部内容。 |

# 1. 前言

为了更好地在Linux/X86环境提供高效安全的日志服务，alog的设计遵循了轻量化、线程安全、灵活、异步、高效的原则。通过在调用程序内存中构造循环链表日志缓冲区的方式实现了日志的异步打印，日志缓冲区以块为单位进行冲刷，有效减少了IO频率提高写入效率。另外，模块本身采用异步线程模型，通过mutex锁和cond条件变量相结合实现线程安全。功能上实现了配置文件实时动态刷新，当前日志和备份日志文件名的模式自定义，调用方可根据调用情况配置单个内存块的大小以及总内存用量大小。整个ALOG模块代码仅千行，可读性良好，除pthread线程库外无其他依赖。

# 2. 全局配置

全局配置可通过用户环境变量注入，启用ALOG之前需要手动加载放置在ALOG安装目录下的env/alog.env文件。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 环境变量名 | 说明 | 单位 | 可选性 | 缺省值 |
| ALOG\_HOME | ALOG模块HOME目录 | 无 | 必填 | 无 |
| ALOG\_SHMKEY | ALOG模块共享内存键值 | 无 | 可选 | 无 |
| ALOG\_SINGLEBLOCKSIZE | 持久化线程单次冲刷的日志块大小 | KB | 可选 | 4 |
| ALOG\_MAXMEMORYSIZE | 单个调用进程对使用的内存上限 | MB | 可选 | 4 |
| ALOG\_FLUSHINTERVAL | 持久化线程强制冲刷间隔 | 秒 | 可选 | 2 |
| ALOG\_CHECKINTERVAL | 配置刷新检查间隔 | 秒 | 可选 | 5 |

env/alog.env举例：

export ALOG\_HOME=${HOME}/alog

export PATH=${PATH}:${ALOG\_HOME}/bin

export ALOG\_SHMKEY=1234

export ALOG\_SINGLEBLOCKSIZE=10

export ALOG\_MAXMEMORYSIZE=16

export ALOG\_FLUSHINTERVAL=1

export ALOG\_CHECKINTERVAL=1

# 3. 自定义配置

自定义日志类型配置存放在${ALOG\_HOME}/cfg/alog.cfg文件中,每一行用于配置一种注册名的日志信息，共7列由中括号包围。

## 列1注册名

用于唯一定位一行配置，对应接口函数中的regname，最大20字节。

## 列2日志级别

一共支持7种级别日志

LOGNON - 无日志

LOGFAT - 致命

LOGERR - 错误

LOGWAN - 警告

LOGINF - 信息

LOGADT - 审计

LOGDBG - 调试

## 列3 单个文件最大尺寸

单位为MB，如当前打印日志文件超过此大小，则进行当前文件备份操作。

## 列4 打印格式

采用位图的方式定义一行日志的前缀组成，共七位。

对应位置位图填写为1时显示，0则不显示。

第一位:日期(YYYYMMDD)

第二位:时间(HH:MM:SS)

第三位:微妙

第四位:进程号+线程号

第五位:模块名

第六列:日志级别

第七列:文件名+行数

## 列5 注册名默认基础路径

接口调用未指定logfilepath时使用此默认基础路径，支持环境变量定义。

# ${} : environment variabale

## 列6 当前日志文件名模式

当前注册名对应日志文件名模式定义。

# ${} : environment variabale

# %R : regname

# %C ：cstname

# %Y ：yead(YYYY)

# %M ：month(MM)

# %D ：date(DD)

# %h ：hour(hh)

# %m ：minute(mm)

# %s ：second(ss)

# %P ：PID

## 列7 备份日志文件名模式

当前注册名对应的备份日志文件名模式定义，模式同当前文件路径。

## 列8 退出强制备份开关

设置成非0时调用alog\_close后将强制将当前日志文件进行备份。

## 举例

[TEST0][LOGINF][1][0111111][${ALOG\_HOME}/log][ %R.%C.log][%R.%C.log.%Y%M%D%h%m%s][1]

# 4. 接口和宏定义

模块对外接口函数和宏定义放在${ALOG\_HOME}/include/alog.h中。

## g\_alog\_ctx

**类型：**

alog\_context\_t\*

**用法：**

调用方无须手动取用，由ALOG其他接口函数使用，用于指向全局上下文结构，在alog\_initContext函数中初始化并申请相关空间，在alog\_close函数中销毁。

## alog\_initContext

**功能：**

alog上下文初始化。在调用其他接口函数之前需要先初始化日志上下文，调用后会对g\_alog\_ctx分配空间并初始化上下文结构体。

**返回值：**

见${ALOG\_HOME}/include/alogerror.h

参数列表：

无

## alog\_close

**功能**：

销毁alog上下文。程序退出之前应销毁日志上下文。调用Close操作会强制冲刷缓冲区中的剩余日志、通知所有的后台持久化线程退出并通知配置刷新线程退出，最终销毁所有申请的内存。

**返回值：**

见${ALOG\_HOME}/include/alogerror.h

**参数列表：**

无

## alog\_writelog\_t

**功能：**

日志打印接口

**返回值：**

见${ALOG\_HOME}/include/alogerror.h

**参数列表：**

**int logtype,**

**enum alog\_level level,**

**char \*regname,**

**char \*cstname,**

**char \*modname,**

**char \*file,**

**int lineno,**

**char \*logfilepath,**

**char \*buf,**

**int len,**

**char \*fmt , ...**

### logtype 日志类型

用于合并调用接口，取值范围：

ALOG\_TYPE\_ASC: ascii码格式

ALOG\_TYPE\_BIN: 二进制格式

ALOG\_TYPE\_HEX: 十六进制格式

### level 日志级别

当前日志的级别，取值范围：

LOGNON,LOGFAT,LOGERR,LOGWAN,LOGINF,LOGADT,LOGDBG

### regname 注册名

用于唯一定位一行自定义配置信息，作为查询alog.cfg第一列的索引，可通过%R注入到文件名中。

### cstname 标识名

用于在同一组regname下进行第二维度的区分，可通过%C注入到文件名中。

### modname 模块名

用于在具体的日志打印时区分不同模块，对应自定义配置中的第四列的位图第五域。

### file文件名

日志产生的源码文件名，取值应为编译宏\_\_FILE\_\_。

### lineno 文件行数

日志产生的源码文件行数，取值应为编译宏\_\_LINE\_\_。

### logfilepath 日志基础路径

日志文件的基础路径

### buf 输入缓冲区

使用二进制或者十六进制打印时，通过buf传入输入缓冲区。

### len 输入缓冲区长度

使用二进制或者十六进制打印时，取值为buf缓冲区的大小长度，也对应最终输出的长度。

### fmt,… 格式化输出模式及可变参数列表

用法同printf格式化输出

## 宏定义

可以通过宏定义的方式封装alog\_writelog\_t接口。

范例如下：

/\* ascii码打印宏 \*/

#define ALOG\_FATASC(regname,cstname,modname,fmt,...)\

alog\_writelog\_t( ALOG\_TYPE\_ASC ,LOGFAT , regname , cstname , modname , \_\_FILE\_\_ , \_\_LINE\_\_ , NULL , 0 , fmt , ##\_\_VA\_ARGS\_\_)

#define ALOG\_ERRASC(regname,cstname,modname,fmt,...)\

alog\_writelog\_t( ALOG\_TYPE\_ASC ,LOGERR , regname , cstname , modname , \_\_FILE\_\_ , \_\_LINE\_\_ , NULL , 0 , fmt , ##\_\_VA\_ARGS\_\_)

#define ALOG\_WANASC(regname,cstname,modname,fmt,...)\

alog\_writelog\_t( ALOG\_TYPE\_ASC ,LOGWAN , regname , cstname , modname , \_\_FILE\_\_ , \_\_LINE\_\_ , NULL , 0 , fmt , ##\_\_VA\_ARGS\_\_)

#define ALOG\_INFASC(regname,cstname,modname,fmt,...)\

alog\_writelog\_t( ALOG\_TYPE\_ASC ,LOGINF , regname , cstname , modname , \_\_FILE\_\_ , \_\_LINE\_\_ , NULL , 0 , fmt , ##\_\_VA\_ARGS\_\_)

#define ALOG\_ADTASC(regname,cstname,modname,fmt,...)\

alog\_writelog\_t( ALOG\_TYPE\_ASC ,LOGADT , regname , cstname , modname , \_\_FILE\_\_ , \_\_LINE\_\_ , NULL , 0 , fmt , ##\_\_VA\_ARGS\_\_)

#define ALOG\_DBGASC(regname,cstname,modname,fmt,...)\

alog\_writelog\_t( ALOG\_TYPE\_ASC ,LOGDBG , regname , cstname , modname , \_\_FILE\_\_ , \_\_LINE\_\_ , NULL , 0 , fmt , ##\_\_VA\_ARGS\_\_)

# 5. 编译和调用方法

## ALOG编译

1. 投产模式：**cd ${ALOG\_HOME}/src & make clean all**

2. 调试模式：**cd ${ALOG\_HOME}/src & make clean debug all**

3. 编译产出

**${ALOG\_HOME}/bin:**

alogcmd alog控制台程序

alogtest alog单元测试程序

**${ALOG\_HOME}/lib:**

libalog.a alog静态库文件

## 调用方编译

1. 修改alog.env中ALOG\_HOME为当前alog所在目录
2. 加载环境变量**cd alog/env && . alog.env**
3. 代码中加入头文件引用：**#include “alog.h”**
4. 调用 **alog\_initContext**初始化上下文
5. 调用 **alog\_writelog\_t**进行日志输出
6. 调用 **alog\_close**销毁上下文
7. 编译时加上： **-I ${ALOG\_HOME}/include**
8. 链接时加上： **-L ${ALOG\_HOME}/lib -lalog**

## 共享内存操作

1. 加载环境变量：

**. ./alog.env**

2. 初始化共享内存

**alogcmd init**

3. 重载配置文件和环境变量

**alogcmd reload**

4. 显示当前共享内存信息

**alogcmd print**

5. 关闭销毁共享内存

**alogcmd close**

## 无状态模式

当不配置ALOG\_SHMKEY环境变量时，无需通过alogcmd启动共享内存，调用进程也不连接共享内存，转而直接从alog.cfg文件读入配置，后台update线程采用定时刷新方式获取alog.cfg的更新，直接修改配置文件即可更新。

# 6. 设计原理说明

## 数据结构

**线程上下文结构alog\_context\_t**

全局指针g\_alog\_ctx记录上下文结构的地址，在alog\_initContext中申请空间并赋值。

**全局配置结构alog\_shm\_t**

对应共享内存结构，用于存放从环境变量读入的全局变量值以及自定义配置的所有条项。上下文结构中的g\_shm指向挂载到程序空间的共享内存地址，l\_shm指向共享内存在程序内存的本地副本，通过后台更新线程每隔一段时间（由全局配置checkInterval定义）进行同步。持久化线程默认访问本地副本l\_shm。

**自定义配置结构alog\_regCfg\_t**

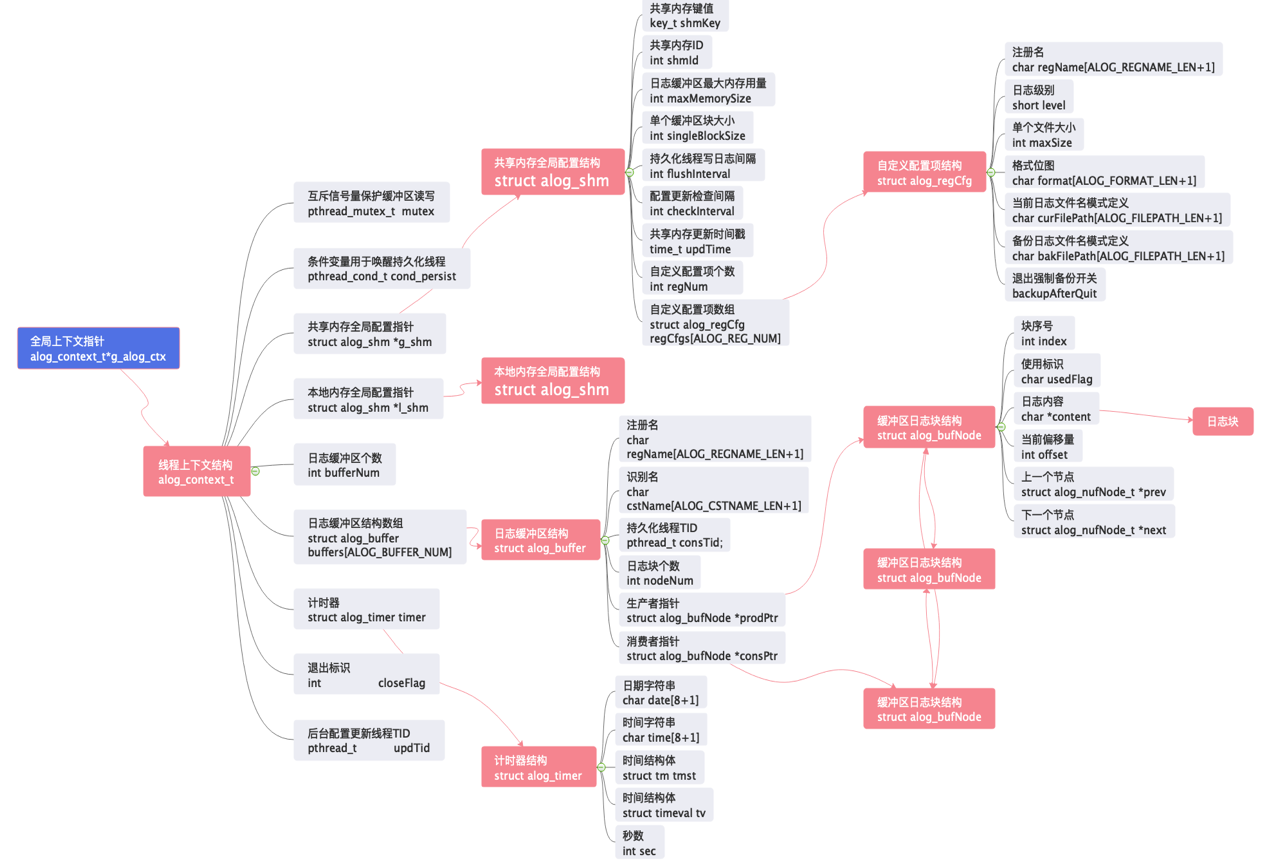
对应alog.cfg中的自定义日志配置，通过regname进行检索。

**计时器结构alog\_timer\_t**

用于记录当前进程环境的日期、时间信息。产生日志时，通过比较日志产生时的秒数与timer的秒数进行判断，如信息一致则直接使用timer中的日期、时间字符串进行输出，如当前日志时间与timer不一致，则更新timer。这样一秒内产生的日志无需重复获取日期和时间信息。

**缓冲区结构alog\_buffer\_t/日志块结构alog\_bufNode\_t**

每一个日志缓冲区由regname+cstname进行检索，对应一个后台持久化线程。其中，产生日志的线程通过prodPtr写入对应的日志块结构，持久化线程则通过consPtr读取日志块内容然后写入具体的物理文件。日志块之间通过指针形成循环双链表，初始化时仅申请一个日志块，如当前日志块出现空间不足时则新增一个日志块连接进链表中，单个日志块的大小(content)由全局配置singleBlockSize定义，缓冲区总内存用量由全局配置maxMemoryUse定义。



## 线程逻辑