变更记录

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **变更序号** | **变更原因** | **变更页码** | **变更前版本号** | **变更后版本号** | **更改人** | **批准人** | **生效日期** | **备注** |
| **1** | **新建文档** |  |  | **1.0** | **王智** | **王智** | **17/5/25** |  |
|  |  |  |  |  |  |  |  |  |

## ilog简介

**驱动力**

电信行业中后台业务缺少易用的日志管理系统，很多现有的日志工具功能简单，尤其是在性能分析方面的空缺，本日志工具针对这些痛点为目标，解决后台系统在日志方面的问题，而部署上只需要简单的配置即可使用，目前支持linux版本；  
  
**软件介绍：**

支持多线程日志、多进程日志管理；

支持普通文件系统、SDFS日志类型；

支持的日志类为：通用日志、运行日志、告警日志、错误日志、统计日志、性能日志；

其中统计日志会根据配置的时间自动生成；

**软件优势：**灵活配置需要生成的日志类型；

统计日志按时间自动生成；

可灵活的自定义性能日志的分析范围；

支持多种文件系统，重启后自动切换，只需修改日志路径即可；

**第三方工具说明：**使用ifle工具类，用于读写等基本文件操作；

**目前使用情况**山西提醒平台使用；

由于该工具没有使用限制，以及后续c++开发中会持续进行使用

## ilog设计图

### 2.1程序结构设计



*2.1程序结构图*

### 2.2类图设计



*2.2类图*

### 2.3流程图设计



*2.3初始化流程图*

**

*2.2调用流程图*

## ilog接口管理

*接口设计在3.1、3.5章节*

### 3.1日志配置解析：

初始化日志配置：

**注：当程序所在主机安装有sdfs时，日志优先选择sdfs进行备份；**

**当所选日志路径在sdfs不存在时，会自动选择普通文件系统进行备份；**

**若均不存在，则init报错；**

需要初始化ilog配置类，进行日志系统初始化，具体代码如下：

ilog\_cfgserver \*log1 = new ilog\_cfgserver();

strcpy(log1->log\_path,"/ocsapp/work/wangzhia/ibf/ilog\_all/demo");

//strcpy(log1->log\_path,"/sdfslog/CLUSTER\_Z00/OFRF");

strcpy(log1->log\_name,"demo\_sdfs");

log1->log\_file\_num = 3;

log1->file\_type = 1;

log1->log\_thread\_num = 0;

log1->logfile\_infos[0].log\_type = 1004;

log1->logfile\_infos[0].log\_procmode = 2;

log1->logfile\_infos[0].log\_backup\_type = 3;

log1->logfile\_infos[0].log\_interval = 10;

log1->logfile\_infos[1].log\_type = 1009;

log1->logfile\_infos[1].log\_procmode = 2;

log1->logfile\_infos[1].log\_backup\_type = 1;

//log1->logfile\_infos[1].log\_interval = 10;

log1->logfile\_infos[2].log\_type = 1003;

log1->logfile\_infos[2].log\_procmode = 2;

log1->logfile\_infos[2].log\_backup\_type = 1;

//log1->logfile\_infos[2].log\_interval = 10;

*建议与icfg配合使用，对配置信息进行加载，配置模板如下：*

[LOG]

#日志服务器路径,根据日志路径选择是否sdfs

log\_path=/bossbgapp/work/wangzhia/log

log\_name=CRON\_BroadBand

file\_type=1 (0,sdfs 1,普通文件 默认0)

#log\_thread\_num=0时,为单线程或多线程写入同一个日志

log\_thread\_num=0

#日志服务器包含日志文件数目

log\_file\_num=3

#1000:通用日志，1001:运行日志，1002:告警日志，1003:错误日志，1004:统计日志，1009:性能日志

log\_type\*1=1004

#日志处理模式（1：同步方式，2：异步方式）

log\_procmode\*1=2

#日志备份类型（1：日期，2：文件大小，3：时间间隔）

log\_backup\_type\*1=3

#时间周期（单位秒）

log\_interval\*1=10

log\_type\*2=1009

log\_procmode\*2=2

log\_backup\_type\*2=1

log\_type\*3=1003

log\_procmode\*3=2

log\_backup\_type\*3=1

### 3.2全局属性：

//日志单例

typedef GDF\_Singleton<ilog\_manager> App\_LogManager;

### 3.3公共接口：

#### 3.3.1 日志初始化及日志加载

调用该函数，初始化日志管理线程；支持云化；若程序是多进程启动或云化进程，请使用函数2，第二个参数传入多进程标示或集群名称即可

//日志初始化（单进程）

bool ilog\_Init (ilog\_cfgserver \*pCfgInfo);

//日志初始化（多进程）

bool ilog\_Init (ilog\_cfgserver \*pCfgInfo,char \*processNum);

bool ilog\_Init (ilog\_cfgserver \*pCfgInfo,int processNum);

//请在加载日志完毕后，调用该函数使得日志进程启动

void ilog\_Run ();

请在需要日志进程结束的时候调用该函数

bool ilog\_Close (void);

判断日志进程是否运行

extern bool ilog\_IsRun ();

#### 3.3.2 日志标签

//设置时间戳，以此处作为程序性能计算的开始

extern void ilog\_Begin(string log\_name);//单线程

extern void ilog\_Begin(string log\_name,UnInt32 threadNum);//多线程

//设置时间戳，计算性能点的其中一个节点，性能日志会根据各个mark节点以及begin、end节点计算相邻节点之前的耗时，并打印到性能日志当中

extern void ilog\_Mark(string log\_name);//单线程

extern void ilog\_Mark(string log\_name,UnInt32 threadNum); //多线程

//设置时间戳，以此处作为程序性能计算的结束点，统计日志会自动分析begin到end之前代码的性能，包含：总处理次数，总耗时，平均耗时，最大耗时，最小耗时

extern void ilog\_End(string log\_name);//单线程

extern void ilog\_End(string log\_name,UnInt32 threadNum); //多线程

#### 3.3.3 写日志

调用WriteLog函数来写日志，其中：

错误日志和告警日志会在每行日志前自动添加当前时间；

统计日志内容根据3.3.2章节函数生成，为自动生成，无需调用WriteLog；

性能日志内容根据3.3.2章节函数生成，需要调用WriteLog；

extern void WriteLog (string log\_name,UnInt32 nLogType, const char \*fmt, ...); //单线程

extern void WriteLog (string log\_name,UnInt32 nLogType, UnInt32 threadNum, const char \*fmt, ...); //多线程

nLogType 取值类型：

LOG\_SYS\_NORMAL = 1000, //通用日志

LOG\_SYS\_RUNNING = 1001, //运行日志

LOG\_SYS\_WARNING = 1002, //告警日志

LOG\_SYS\_ERROR = 1003, //错误日志

LOG\_APP\_STATINFO = 1004, //统计日志

LOG\_APP\_PERFORMANCE = 1009 //性能分析

统计日志：总数量|总耗时|平均耗时|最大耗时|最小耗时

实例：10|5721|572|1021|435

性能日志：[end到begin之间的耗时]|mark1耗时|mark2耗时|…|markn耗时|end:end到最后一次mark的耗时

注：[end到begin之间的耗时]= mark1耗时+mark2耗时+…+markn耗时+end到最后一次mark的耗时

实例：[2741]14|1940|end:787

## ilog环境搭建及DEMO

### 4.1环境搭建：

**1．需要配置ifile环境变量（若主机已经搭建了ifle则忽略该步骤）：**

export IBF\_PATH=安装目录

export IFILE\_HOME=$IBF\_PATH/ifile\_all/ifile

export LD\_LIBRARY\_PATH=$IBF\_PATH/ifile\_all/ifle/lib:$LD\_LIBRARY\_PATH

如果需要重新编译动态库：到ifile\_all/ifle/src目录下，执行sh make\_all

**2．需要配置ilog环境变量（如果使用静态库，无需配置环境变量）**

export LD\_LIBRARY\_PATH=$IBF\_PATH/ilog\_all/ilog/lib:$LD\_LIBRARY\_PATH

如果需要重新编译动态库：到ilog\_all/ilog/src目录下，执行sh make\_all

### 4.2 demo使用：

修改demo中的路径及文件名，编译后即可执行