GL4A 日志库设计思想

作者：欧阳天华　　邮箱：kyzy_duck@163.com　　QQ:2798352856

1 目的：

用于工业设备问题追踪和分析。

用于在运行和调试过程中，记录崩溃所在的大致位置。

用于在日常运行过程中，监控必要的数据。

2 特点

记录时间小于 1ms；

实时记录，可用于崩溃追踪；

缓冲记录，可用于记录日常运行数据；

记录文件名和行号，并支持格式化输出；

支持运行过程中实时重加载配置；

可记录在控制台中，自由选择功能开关；

可记录在文件中，自由选择功能开关；

可自由添加组件/模块，分模块记录，模块可自由开关；

可自由配置日志等级。

3 组成

3.1 主要概念

组件、日志记录器、等级

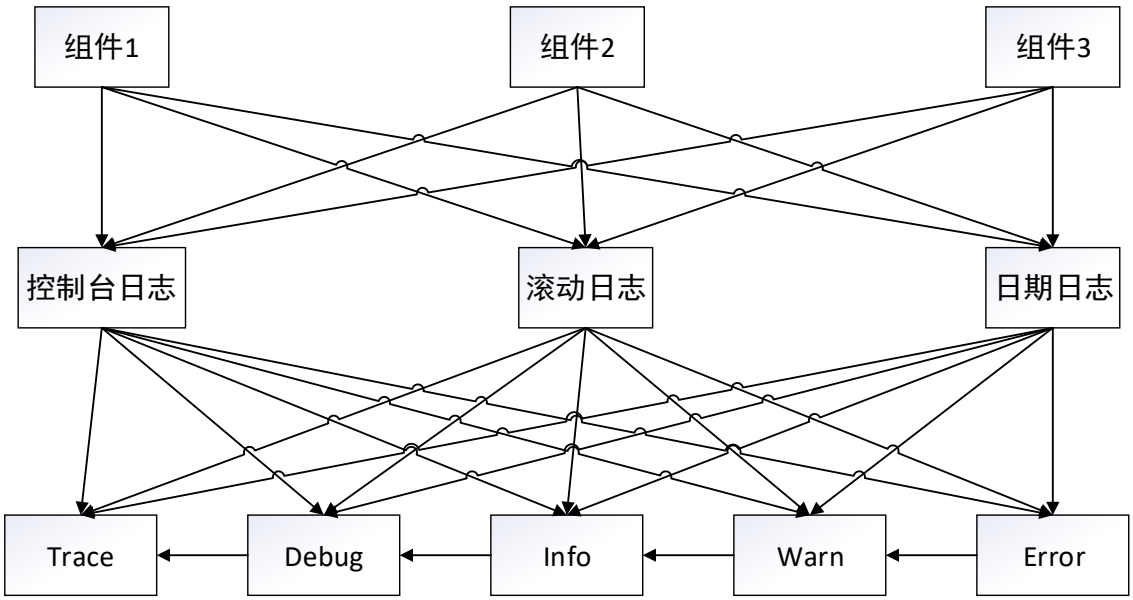
组件可由用户增删改，组件可自由开关，如果某个组件关闭，那么所有其挂载的日志记录器均不会对其进行日志记录。

日志记录器分为 console 控制台日志记录、rolling 滚动型日志记录、daily 按天日志记录。三种日志记录器均可以自由开关，如果某种日志记录器被关闭，那么所有此类型的日志均不会被记录。

等级分为 trace 和 error，其中，error 等级用于追踪崩溃，trace 用于日常数据记录。所有 error 日志均会在 trace 中备份记录一份，主要用于保持 trace 数据上下文完整性。

所有配置项均存在 config 文件夹中的配置文件 GL4A_config.cfg 中。

3.2 结构图



4 使用

4.1 基本用法

```
{
    // 在程序入口处，根据配置文件路径，初始化日志库
    eGL4A_init(GL4A_DEFAULT_CONFIG_DIR);

    // 在 M0 组件函数入口处，记录 ERROR，并标识为 IN（用于定位大致崩溃位置）
    GL4A_ERROR_LOG("M0", __F__, __L__, "XXX 函数，IN！\n");
    // 在中间必要时，记录 TRACE，用于日常数据记录
    GL4A_TRACE_LOG("M0", __F__, __L__, "记录数据！\n");
    // 在 M0 组件函数入口处，记录 ERROR，并标识为 OUT（用于定位大致崩溃位置）
    GL4A_ERROR_LOG("M0", __F__, __L__, "XXX 函数，OUT！\n");

    // 在程序出口处，关闭日志库
    GL4A_close();
}
```

注：当程序发生崩溃时，相应函数入口 ERROR IN 会被记录，但 ERROR OUT 不会被记录，那么，根据记录的文件名和行号，即可定位崩溃发生的位置。

在记录 ERROR 时，同时会在 TRACE 中备份记录一份，用于保持 TRACE 记录上下文的完整性。

4.2 基本函数

主要包括两个部分：

1. 日志库初始化/关闭/重加载，日常输出

GL4A_init——用于初始化日志库

GL4A_reloadConfig——用于实时重加载日志配置文件

GL4A_close——用于关闭日志库

GL4A_TRACE_LOG——用于输出 TRACE 日志

GL4A_ERROR_LOG——用于输出 ERROR 日志

2. 日志库配置文件修改

loadCfgFileToStruct——用于获取配置文件结构体

saveCfgFileFromStruct——用于将配置结构体保存到配置文件

genConfigDefault——重新按照默认的配置结构体生成配置文件

TotalCfgInfos——配置结构类，其中包含对配置项的基本操作

注：如果对 json 格式比较了解，可直接对配置文件进行操作

注：具体参见 GL4A_user.h 中 GL4A 命名空间，有非常详尽的注释