Google日志系统

参考链接：

- 源码仓库：Google Logging
- 使用指南：google/glog: C++ implementation of the Google logging module (github.com)

1.简介

Google日志系统是Google提供的一个C++库，用于简化C++程序的日志记录。它提供了一个易于使用的日志记录API，支持灵活的日志级别控制、日志信息格式化、多线程安全等特性。

1. **灵活的日志级别控制**：支持不同的日志级别，包括 `INFO`、`WARNING`、`ERROR`、`FATAL` 等。可以根据需要动态设置日志级别，以便在不同的环境中灵活控制日志输出量。
2. **日志信息格式化**：支持类似于printf的格式化输出，可以将变量值、函数名等信息插入日志信息中，方便调试和排查问题。
3. **轻量级且高效**：Google Glog设计简洁，不依赖于其他库，可以轻松地集成到现有的C++项目中。同时，它的实现经过优化，保证了高效的日志记录性能。
4. **多线程安全**：Google Glog提供了多线程安全的日志记录机制，可以在多线程环境下安全地使用，避免了由于多线程竞争而导致的日志输出混乱或丢失的问题。
5. **支持日志文件分割**：可以设置日志文件的最大大小和数量，当日志文件大小达到设定的阈值时，自动进行日志文件切割，避免单个日志文件过大。
6. **跨平台支持**：Google Glog可以在多种操作系统上运行，包括Linux、Windows、macOS等，具有良好的跨平台性。

2.使用glog

使用glog时按严重性级别记录消息、从命令行控制日志记录行为、基于条件进行日志记录、在未满足预期条件时中止程序、引入自己的详细日志记录级别、自定义附加到日志消息的前缀等。

您可以指定以下严重性级别之一（按严重性递增顺序）：

```
1. `INFO`, //信息
2. `WARNING`, //警告
3. `ERROR` //错误
4. `FATAL`. //严重错误
```

INFO级别日志用于记录程序执行的一般性信息，例如进入某个函数、执行某个操作的结果等。这些日志通常用于跟踪程序的执行流程和输出一些关键状态信息，以便在需要时进行排查和调试。

WARNING级别日志用于记录程序中的潜在问题或不符合预期的情况，但并不会影响程序的正常执行和结果。这些日志通常用于指示一些警告性的条件或行为，提示开发人员可能需要关注和修复的地方，但不会导致程序的崩溃或错误。

ERROR级别日志用于记录程序中的错误情况，表示程序执行过程中发生了一些无法忽略的错误或异常。这些日志通常用于标识程序中的问题，需要开发人员及时处理和修复，以确保程序的正确运行和预期结果。

FATAL级别日志用于记录程序中的严重错误，表示发生了一个无法恢复的错误，导致程序无法继续执行。一旦触发FATAL日志，程序通常会立即终止运行。这些日志通常用于标识非常严重的错误或异常情况，需要开发人员立即处理和修复。

3.全志平台使用glog

参考示例程序: `sample_glog`

3.1 使用glog库

使用 Google 的日志库（glog）的基本步骤如下：

1. **初始化库**：在开始记录日志之前，必须先初始化 glog。这通常在程序的 `main` 函数中完成：

```
google::InitGoogleLogging(argv[0]); //
```

2. **设置日志文件保存目录**：在初始化库之前，设置日志文件的保存目录。这个目录必须已经存在，否则 glog 无法创建日志文件：

```
FLAGS_log_dir = "/tmp/log";
```

3. **定义日志级别**：glog 提供了四个日志级别：

```
enum SeverityLevel {
    google::INFO = 0,
    google::WARNING = 1,
    google::ERROR = 2,
    google::FATAL = 3,
};
```

4. **记录日志**：使用 `LOG` 宏来记录日志，例如：

```
LOG(INFO) << "info test"; // 记录一个 Info 级别的日志
LOG(WARNING) << "warning test"; // 记录一个 Warning 级别的日志
LOG(ERROR) << "error test"; // 记录一个 Error 级别的日志
LOG(FATAL) << "fatal test"; // 记录一个 Fatal 级别的日志，会中止程序
```

5. **条件输出**：可以根据条件来决定是否输出日志：

```
LOG_IF(INFO, num_cookies > 10) << "Got lots of cookies";
```

6. **关闭库**：在程序结束前，关闭 glog 来释放资源：

```
google::ShutdownGoogleLogging();
```

3.2 使用封装好的glog库

1.设置日志相关参数

```
    GLogConfig stGLogConfig =
    {
        .FLAGS_logtostderr = 0,
        .FLAGS_colorlogtostderr = 1,
        .FLAGS_stderrthreshold = _GLOG_INFO,
        .FLAGS_minloglevel = _GLOG_INFO,
        .FLAGS_logbuflevel = -1,
        .FLAGS_logbufsecs = 0,
        .FLAGS_max_log_size = 25,
        .FLAGS_stop_logging_if_full_disk = 1,
    };
    strcpy(stGLogConfig.LogDir, "/tmp/log");
    strcpy(stGLogConfig.InfoLogFileNameBase, "LOG-");
    strcpy(stGLogConfig.LogFileNameExtension, "SDV-");
```

2.初始化glog库

```
log_init(argv[0], &stGLogConfig);
```

3.记录日志：使用 函数来记录日志，例如：

```
alogv("v, hello, world!");
alogd("d, hello, world!");
alogw("w, hello, world!");
aloge("e, hello, world!");
```

4.退出log，停止保存log信息。

```
log_quit();
```