**日志系统设计文档**

版本 1.0

版权所有 © 2006-2011 杭州淘淘搜科技有限公司

公司网址：<http://www.taotaosou.com>

修订历史

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 2011/9/30 | 1.0 | 初稿 | Focus |

**目录**

[修订历史 2](#_Toc305158899)

[1引言 4](#_Toc305158900)

[1.1编写目的 4](#_Toc305158901)

[1.2背景 4](#_Toc305158902)

[1.3定义 4](#_Toc305158903)

[1.4参考资料 4](#_Toc305158904)

[2总体设计 4](#_Toc305158905)

[2.1需求规定 4](#_Toc305158906)

[2.2运行环境 5](#_Toc305158907)

[2.3基本设计概念和处理流程 5](#_Toc305158908)

[2.4结构 5](#_Toc305158909)

[3接口设计 5](#_Toc305158910)

[3.1用户接口 5](#_Toc305158911)

[3.2外部接口 6](#_Toc305158912)

[3.3内部接口 6](#_Toc305158913)

[4使用说明 6](#_Toc305158914)

[4.1初始化 6](#_Toc305158915)

[4.2使用 6](#_Toc305158916)

[4.3释放 7](#_Toc305158917)

# 1引言

## 1.1编写目的

便于软件开发人员使用和扩展该日志系统。

## 1.2背景

目前引擎及图像算法没有一个统一的日志系统，而日志系统在整个系统中是非常重要。比如在系统运行期间是很难做单步调试的，所以只能根据系统运行的轨迹来推断出错的位置；另外还可以用来作为信息采集，运营维护。

## 1.3定义

无。

## 1.4参考资料

1.Jdk日志系统；

2.log4cplus日志系统；

3.设计模式

# 2总体设计

## 2.1需求规定

1. 使用方便，包括开发和信息检索；

2. 组织合理，日志内容被有效的组织起来。有助于基于日志内容的分析；

3. 过程重现，这个目标其实对开发者来说是最重要的。基于日志的分析，最重要的是重现当时运行的过程；

4.线程安全；

5.按日期自动生成新日志文件；

6.统一输出格式，便于合并、查找；

7.日志分级；

8.对性能无影响；

## 2.2运行环境

该日志系统主要用于图像算法和引擎服务；

**软件环境**：linux/c++；

**硬件环境**：无特殊要求；

## 2.3基本设计概念和处理流程

该日志系统根据2.1需求规定所提出的需求来进行设计，使用非常方便，可以根据使用者需求，输出统一格式的信息，对日志信息进行过滤，可扩展的特性。

## 2.4结构



# 3接口设计

## 3.1用户接口

主要以两种方式提供：

1. 类的方式；
2. 宏定义方式；

## 3.2外部接口

类接口：

void CLogger::Log(CLogRecord record)

宏定义：

LOG\_ERROR(logger,logRecord)

LOG\_WARN(logger,logRecord)

LOG\_INFO(logger,logRecord)

LOG\_DEBUG(logger,logRecord)

LOG\_TRACE(logger,logRecord)

## 3.3内部接口

参见源代码。

# 4使用说明

## 4.1初始化

//获取日志管理器：

CLogManager\* pManager=CLogManager::Instance();

//创建一个过滤器，并设置过滤等级：

SharedFilterPtr filter\_ptr(new CLevelFilter());

static\_cast<CLevelFilter\*>(filter\_ptr.Get())->SetLevel(CLevel(TRACE\_LOG\_LEVEL));

//创建格式化：

SharedFormatterPtr console\_formatter(new ConsoleFormatter());

//创建日志处理句柄

SharedHandlerPtr console\_ptr(new ConsoleHandler());

//设置过滤器

console\_ptr->SetFilter(filter\_ptr);

//设置格式化

console\_ptr->SetFormatter(console\_formatter);

//添加到日志管理器

pManager->AddHandler(console\_ptr);

## 4.2使用

方法1:

获取日志记录器：

CLogger logger=pManager->GetLogger();

通过宏进行日志记录：

LOG\_DEBUG(logger,"this is a deubg"<<endl);

LOG\_ERROR(logger,"this is a error"<<endl);

LOG\_FATAL(logger,"this is fatal"<<endl);

LOG\_INFO(logger,"this is info"<<endl);

LOG\_TRACE(logger,"this is trace"<<endl);

LOG\_WARN(logger,"this is warn"<<endl);

LOG\_INFO(logger,"this is a bool:"<<true<<endl);

LOG\_INFO(logger,"this is a char:"<<'x'<<endl);

LOG\_INFO(logger,"this is a string:"<<string("string")<<endl);

LOG\_INFO(logger,"this is a short:"<<(short)-100<<endl);

LOG\_INFO(logger,"this is a int:"<<(int)1000<<endl);

LOG\_INFO(logger,"this is a unsigned int:"<<(unsigned int)1000<<endl);

LOG\_INFO(logger,"this is a long(hex):0x"<<hex<<(long)1000000<<endl);

LOG\_INFO(logger,"this is a unsigned long:"<<(unsigned long)100000000<<endl);

LOG\_INFO(logger,"this is a float:"<<(float)1.2345<<endl);

方法2:

LOG\_DEBUG(CLogManager::Instance()->GetLogger(),"this is a deubg"<<endl);

## 4.3释放

释放资源：

方法1:CLogManager::DestroyInstance();

方法2：pManager->DestroyInstance();

具体详情可参考源代码示例。