Log4CPlus 总结



Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev. #** | **Date of change** | **Name of person making change** | **Description of Change** |
| 1.0 | 29-06-2019 | 娄伟 | Initial creation |
|  |  |  |  |

[Table of Contents](https://www-ps.net.plm.eds.com/wiki-nx/index.php/General_Instructions%23Table_of_Contents)

[Appendix A: Glossary of Terms and Definitions 4](#_Toc12716176)

[Appendix B: Other Considerations 4](#_Toc12716177)

[Appendix C: Input for What's New Guide 4](#_Toc12716178)

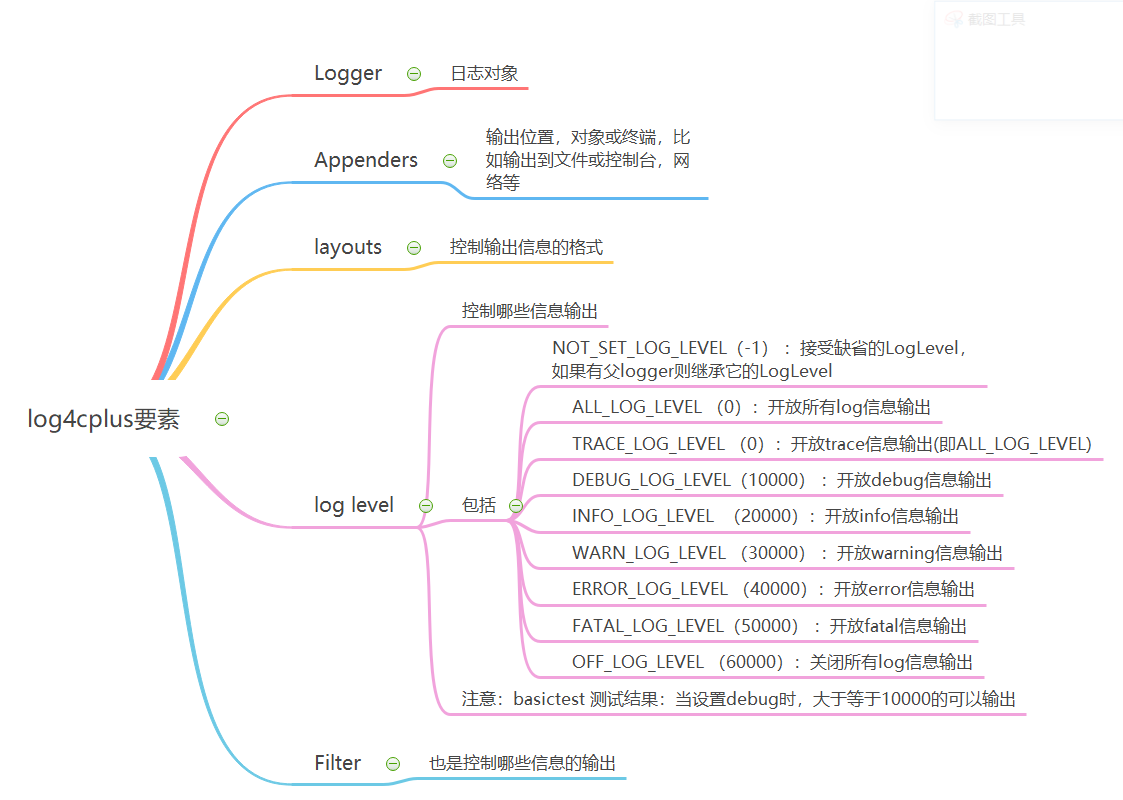
[Appendix D: Previous Dialogs 4](#_Toc12716179)

# 简介

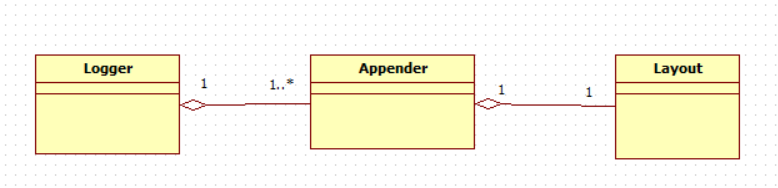
log4cplus is a simple to use C++ logging API providing **thread-safe**, flexible, and arbitrarily granular control over log management and configuration. It is modelled after the Java log4j API.

# 术语介绍

## Log4CPlus要素



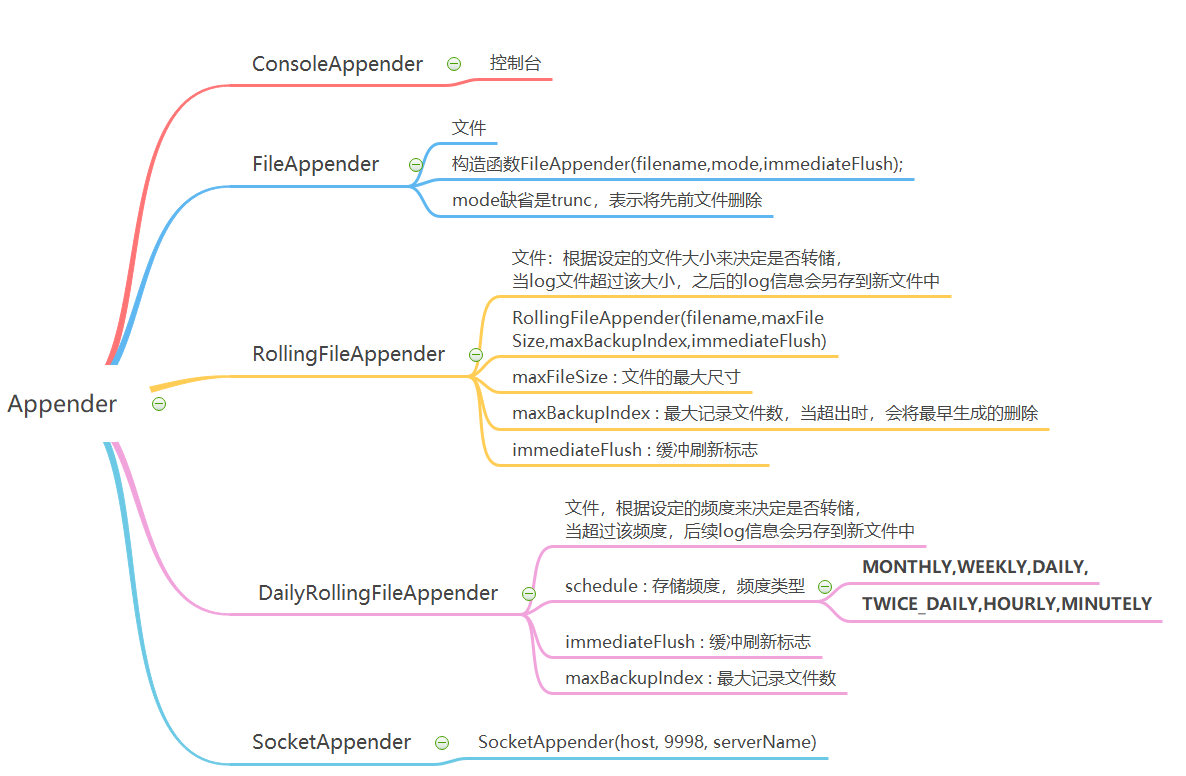
## 各个要素之间的关系



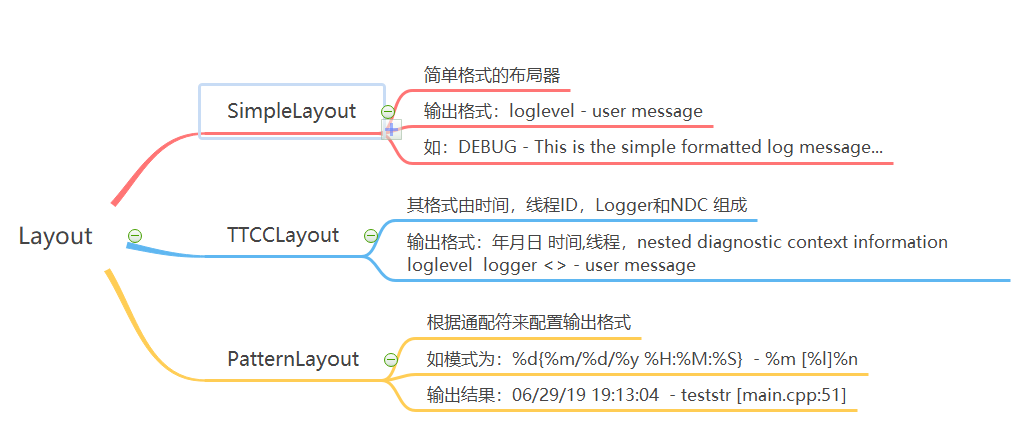
## Logger



## appender



## layout



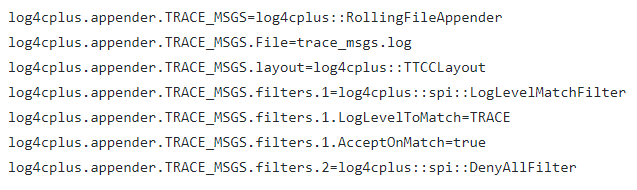
## filter



配置文件配置filter是很好写的，可以参考例子<https://github.com/log4cplus/log4cplus/blob/master/tests/filter_test/log4cplus.properties.in>

如下设置先过滤掉所有的输出，然后设置只trace输出.

最后只有trace信息可以输出.



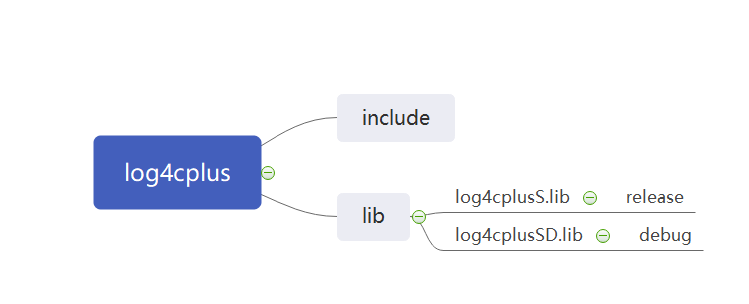
## Nested Diagnostic Context

N/A

# 应用

在该文档同目录下有个工程log4cplus，用vs编写各种测试代码.

Log4cplus已经编成库，其文件结构如下，以下测试代码都是用该库。



## 基本的测试

请参考项目：BasicTest

* 该测试代码探讨了 log4cplus 使用步骤
  + 定义appender
  + 定义layout
  + 定义log
  + Loglevel 控制
  + 打印log
* 静态库参加工程目录下文件夹ThirdParty下

#include <log4cplus/logger.h>

#include <log4cplus/consoleappender.h>

#include <log4cplus/layout.h>

#include <log4cplus/ndc.h>

#include <log4cplus/mdc.h>

#include <log4cplus/helpers/loglog.h>

#include <log4cplus/thread/threads.h>

#include <log4cplus/loggingmacros.h>

#include <log4cplus/fileappender.h>

#include <iostream>

#include <string>

#include<thread>

using namespace *std*;

using namespace log4cplus;

using namespace log4cplus::helpers;

int main()

{

// define appender

SharedAppenderPtr pConsoleAppender(new ConsoleAppender());

SharedAppenderPtr pFileAppender(new FileAppender("..\Basic\_testfile.log"));

SharedAppenderPtr pFileAppenderWithPattern(new FileAppender("..\Basic\_patter.log"));

// define layout

*auto\_ptr*<Layout> pSimpleLayout(new SimpleLayout());

pConsoleAppender->setLayout(pSimpleLayout);

*auto\_ptr*<Layout> pTTCLayout(new TTCCLayout());

pFileAppender->setLayout(pTTCLayout);

*auto\_ptr*<Layout> pPatternLayout(new PatternLayout("%d{%m/%d/%y %H:%M:%S,%Q} [%t] %-5p %c{2} %%%x%% - %X{key} - %m [%l]%n"));

pFileAppenderWithPattern->setLayout(pPatternLayout);

// define Logger

Logger rootLogger = Logger::getInstance("root");

// Note：childLogger is child of rootLogger and will inherit all appenders.

Logger childLogger = Logger::getInstance("root.child");

// attach appenders to logger

rootLogger.addAppender(pConsoleAppender);

rootLogger.addAppender(pFileAppender);

childLogger.addAppender(pFileAppenderWithPattern);

childLogger.setLogLevel(log4cplus::ALL\_LOG\_LEVEL);

// print syslog

LOG4CPLUS\_WARN(rootLogger, "rootLogger:this is a warning message...");

LOG4CPLUS\_TRACE(Logger::getRoot(), "Logger::getRoot:this is a TRACE message...");

LOG4CPLUS\_DEBUG(childLogger, "childLogger:this is a DEBUG message...");

LOG4CPLUS\_INFO(childLogger, "childLogger:this is a INFO message...");

LOG4CPLUS\_ERROR(childLogger, "childLogger:this is a ERROR message...");

LOG4CPLUS\_FATAL(childLogger, "childLogger:this is a FATAL message...");

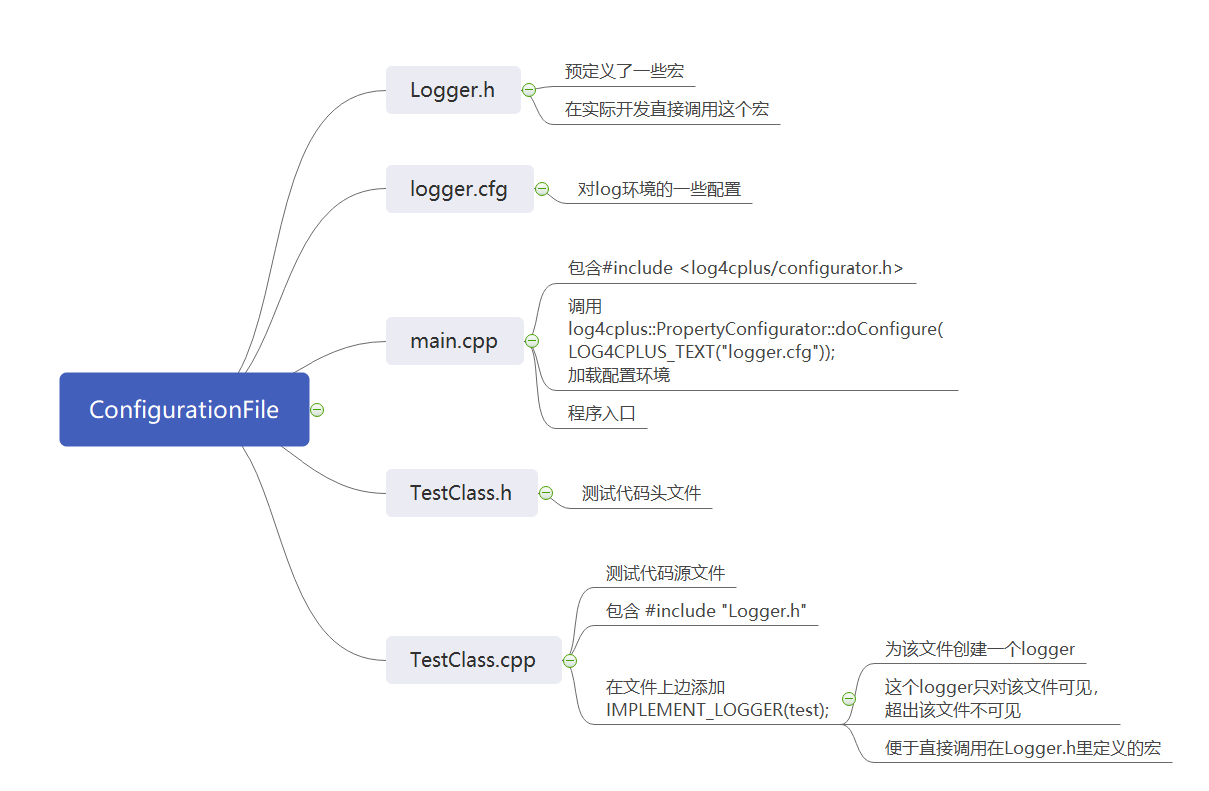
*getchar*();

return 0;

}

## 可用配置文件

除了3.1节用代码来实现log环境配置以外，log4cplus利用PropertyConfigurator实现了脚本配置.本小节使用的方法是目前MPR,FFRCT 和AI 小精灵使用的方法.可参见项目名称ConfigurationFile, 主要的文件结构如下：



该方法的应用的主要步骤如下：

* 把头文件Logger.h加入自己的代码中
* 配置logger的脚本文件logger.cfg
* 在应用程序加载时加载该配置文件
* 在要打日志的文件顶部，添加IMPLEMENT\_LOGGER(test);
* 可以直接调用Logger.h上的宏打印日志.

## 线程安全

这个写了两个小测试程序，项目名称为MultipleThreadTest 和ThreadTestInLOG

## 抽象一个单例类

由于在3.2 小节中，在添加每一个文件时都需要IMPLEMENT\_LOGGER(test)，而且每一个文件都必须要有一个logger，所以试图创建一个单例类SysLogger，用户只需关注该类的调用.

# 主要链接

Github源码：<https://github.com/log4cplus/log4cplus>

DOC: <https://sourceforge.net/p/log4cplus/wiki/Home/#documentation>

下载：<https://sourceforge.net/projects/log4cplus/files/log4cplus-stable/>

Example code：<https://github.com/log4cplus/log4cplus/tree/master/tests>

Socketappender：<https://github.com/log4cplus/log4cplus/blob/master/tests/socket_test/main.cxx>

Filter Test：

<https://github.com/log4cplus/log4cplus/tree/master/tests/filter_test>

1. PatternLayout 通配符介绍

"%%"，转义为% 。

"%c"，输出logger名称，如test.subtest 。也可以控制logger名称的显示层次，比如"%c{1}"时输出"test"，其中数字表示层次。  
"%D"，显示本地时间，比如："2004-10-16 18:55:45"，%d显示标准时间。     可以通过%d{...}定义更详细的显示格式，比如%d{%H:%M:%s}表示要显示小时:分钟：秒。大括号中可显示的  
     预定义标识符如下：  
       
   %a -- 表示礼拜几，英文缩写形式，比如"Fri"  
   %A -- 表示礼拜几，比如"Friday"  
   %b -- 表示几月份，英文缩写形式，比如"Oct"  
   %B -- 表示几月份，"October"  
   %c -- 标准的日期＋时间格式，如 "Sat Oct 16 18:56:19 2004"  
   %d -- 表示今天是这个月的几号(1-31)"16"  
   %H -- 表示当前时刻是几时(0-23)，如 "18"  
   %I -- 表示当前时刻是几时(1-12)，如 "6"  
   %j -- 表示今天是哪一天(1-366)，如 "290"  
   %m -- 表示本月是哪一月(1-12)，如 "10"  
   %M -- 表示当前时刻是哪一分钟(0-59)，如 "59"  
   %p -- 表示现在是上午还是下午， AM or PM  
   %q -- 表示当前时刻中毫秒部分(0-999)，如 "237"  
   %Q -- 表示当前时刻中带小数的毫秒部分(0-999.999)，如 "430.732"  
   %S -- 表示当前时刻的多少秒(0-59)，如 "32"  
   %U -- 表示本周是今年的第几个礼拜，以周日为第一天开始计算(0-53)，如 "41"  
   %w -- 表示礼拜几，(0-6, 礼拜天为0)，如 "6"  
   %W -- 表示本周是今年的第几个礼拜，以周一为第一天开始计算(0-53)，如 "41"  
   %x -- 标准的日期格式，如 "10/16/04"  
   %   X -- 标准的时间格式，如 "19:02:34"  
   %y -- 两位数的年份(0-99)，如 "04"  
   %Y -- 四位数的年份，如 "2004"  
   %Z -- 时区名，比如 "GMT"

"%F"，输出当前记录器所在的文件名称，比如"main.cpp"  
"%L"，输出当前记录器所在的文件行号，比如"51"  
"%l"，输出当前记录器所在的文件名称和行号，比如"main.cpp:51"  
"%m"，输出原始信息。  
"%n"，换行符。

"%p"，输出LogLevel，比如"DEBUG"  
"%t"，输出记录器所在的线程ID，比如 "1075298944"  
"%x"，嵌套诊断上下文NDC (nested diagnostic context) 输出，从堆栈中弹出上下文信息，NDC可以用对不同源的log信息（同时地）交叉输出进行区分。  
格式对齐，比如"%-10m"时表示左对齐，宽度是10，当然其它的控制字符也可以相同的方式来使用，比如"%-12d"，"%-5p"等等。

1. [Other Considerations](https://www-ps.net.plm.eds.com/wiki-nx/index.php/Functional_Spec%23Other_Considerations)

N/A

1. [Input for What's New Guide](https://www-ps.net.plm.eds.com/wiki-nx/index.php/Functional_Spec%23Input_for_What.27s_New_Guide)

***N/A***

1. [Previous Dialogs](https://www-ps.net.plm.eds.com/wiki-nx/index.php/Functional_Spec%23Previous_Dialogs)

N/A