## 日志组件简介

基于[log4cplus](http://sourceforge.net/projects/log4cplus/files/log4cplus-stable/)修改。（log4cplus是C++编写的开源的日志系统，前身是java编写的log4j系统。）

修改内容：

1） 只支持Windows和Linux平台，去掉其他平台的支持（vs2010和gcc4.4测试通过）。

2） 去除unicode的支持

3） 只保留consoleAppender、fileAppender；并添加customAppender，以方便我们通过回调函数调用本地的方法（如windows平台的OutputDebugString）。

4） 精简宏，只保留目前会使用到的宏。

5） 修改目录结构，修改命名空间，修改代码写法，修改为我熟悉的方式。

6） 去掉配置文件看门狗线程。

7） 去掉原有锁，替换简单的锁

8） 添加log4cplus封装类，封装类头文件不依赖于log4cplus，这样便于使用其他的日志组件（如glog、log4cpp等）。

## 使用限制

1. **必须使用配置文件。**修改的目的就是为了使用配置文件，让使用的人员不知道细节。
2. **必须自己实现封装类。**实现log4cplus封装类目的是屏蔽对log4cplus的依赖，这样便于项目替换其他的日志组件（比如glog，或者我们公司的日志组件），svn流上已经提供了一个log4cplus封装类，开发人员可以基于该类修改。
3. **不支持unicode。**如果需要写入unicode字符流，请自行转换。

## Log4cplus封装类的编写

Log4cplus默认支持日志输入到控制台和文件。如果需要设置输出到其它地方需要设置customerAppender的回调函数

如果程序中需要支持osp，可以编写如下回调函数：

#ifdef WIN32

void \_\_stdcall ospAppendFunc(const char\* sz)

{

OutputDebugString(sz);

}

#else

void ospAppendFunc(const char\* sz)

{

}

#endif

并在StartLogSystem中添加一行：

log4cplus::CustomAppender::setCustomFunc(ospAppendFunc);

当然你也需要包含osp的头文件。

## 配置文件说明

### 3.1 举例说明

典型的配置文件：

# Define logger.logFile1

log4cplus.logger.logFile1=TRACE, customAppender, consoleAppender, fileAppender1

# Define logger.logFile2

log4cplus.logger.logFile2=TRACE, customAppender, consoleAppender, fileAppender2

# Define a file appender named "consoleAppender"

log4cplus.appender.consoleAppender=ConsoleAppender

log4cplus.appender.consoleAppender.layout=SimpleLayout

# log4cplus.appender.consoleAppender.filters.1=LogLevelMatchFilter

# log4cplus.appender.consoleAppender.filters.1.LogLevelToMatch=INFO

# log4cplus.appender.consoleAppender.filters.2=DenyAllFilter

# Define a file appender named "customAppender"

log4cplus.appender.customAppender=CustomAppender

log4cplus.appender.customAppender.layout=SimpleLayout

log4cplus.appender.customAppender.filters.1=LogLevelMatchFilter

log4cplus.appender.customAppender.filters.1.LogLevelToMatch=INFO

log4cplus.appender.customAppender.filters.2=DenyAllFilter

# Define a file appender named "fileAppender1"

log4cplus.appender.fileAppender1=RollingFileAppender

log4cplus.appender.fileAppender1.MaxFileSize=200KB

log4cplus.appender.fileAppender1.File=./fileAppender1/fileAppender1.log

log4cplus.appender.fileAppender1.CreateDirs=true

log4cplus.appender.fileAppender1.ImmediateFlush=true

log4cplus.appender.fileAppender1.MaxBackupIndex=3

log4cplus.appender.fileAppender1.layout=SimpleLayout

log4cplus.appender.fileAppender1.filters.1=LogLevelRangeFilter

log4cplus.appender.fileAppender1.filters.1.LogLevelMin=INFO

log4cplus.appender.fileAppender1.filters.1.LogLevelMax=ERROR

log4cplus.appender.fileAppender1.filters.1.AcceptOnMatch=true

log4cplus.appender.fileAppender1.filters.2=DenyAllFilter

# Define a file appender named "fileAppender2"

log4cplus.appender.fileAppender2=DailyRollingFileAppender

log4cplus.appender.fileAppender2.File=fileAppender2.log

log4cplus.appender.fileAppender2.Schedule=DAILY

log4cplus.appender.fileAppender2.DatePattern = yyyy-MM-dd

log4cplus.appender.fileAppender2.MaxFileSize=10MB

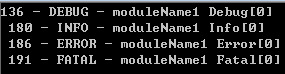
log4cplus.appender.fileAppender2.MaxBackupIndex=2

log4cplus.appender.fileAppender2.layout=PatternLayout

log4cplus.appender.fileAppender2.layout.ConversionPattern=%d - %p - %m %n

说明：

1. #为注释，起说明作用。
2. 一共有两个模块：logFile1和logFile2；logFile1可以输出到控制台、自定义的输出（比如osp）和fileAppender1.log；logFile2可以输出到控制台、自定义的输出（比如osp）和fileAppender2.log。其中有两个输出文件，分别为当前目录的fileAppender2.log和在当前目录子目录fileAppender1的fileAppender1.log
3. 输出控制台的配置为consoleAppender，可以输出所有等级的日志信息，输出格式为简单格式，如：



1. customAppender为自定义的输出。只能输出info等级的日志，输出格式也为内置的简单格式。
   1. fileAppender1输出到文件，在当前目录创建了一个子目录fileAppender1，并将日志输出到子目录中的fileAppender1.Log;
   2. 采用回滚的形式，最多有3个200KB的文件同时存在
   3. 所有日志没有使用缓存，来一行信息立刻写入文件中（跟ImmediateFlush相关）。
   4. 只输出INFO和ERROR的信息,其他等级的被过滤掉
   5. 输出形式使用了内置的SimpleLayout
   6. fileAppender2输出到文件fileAppender2.Log;
   7. 采用天记录的形式，每天最多有2个10MB的文件。
   8. 输出日志采用了自定义的格式，日期格式为：yyyy-MM-dd，删除格式为：

%d - %p - %m %n

### 3.2 配置

基本配置语法主要针对包括rootLogger和non-root logger。

**根Logger的配置**

语法：

log4cplus.rootLogger=[LogLevel], appenderName, appenderName, ...

**非根Logger的配置**

语法：

log4cplus.logger.logger\_name=[LogLevel|INHERITED], appenderName, appenderName, ...

说明：INHERITED表示继承父Logger的日志级别。

#### Appender配置

语法：

log4cplus.appender.appenderName=fully.qualified.name.of.appender.class

举例：

log4cplus.appender.consoleAppender=ConsoleAppender

#### Filter配置

Appender可以附加Filter组成的链表，如果Filter链中存在过滤器Filter， log4cplus在输出日志之前将调用链表中Filter的过滤方法decide(),根据该方法的返回值决定是否过滤该输出日志。

语法：

log4cplus.appender.appenderName.Filter.FilterNumber= Filter.class

log4cplus.appender.appenderName.Filter.FilterNumber.FilterCondition=value.of.FilterCondition

举例：

log4cplus.appender.consoleAppender.filters.1=LogLevelMatchFilter

log4cplus.appender.consoleAppender.filters.1.LogLevelToMatch=INFO

log4cplus.appender.consoleAppender.filters.2=DenyAllFilter

目前log4plus提供的过滤器包括DenyAllFilter 、LogLevelMatchFilter、LogLevelRangeFilter、和StringMatchFilter。

* LogLevelMatchFilter根据特定的日志级别进行过滤。

过滤条件包括LogLevelToMatch和AcceptOnMatch（true|false）， 只有当日志的LogLevel值与LogLevelToMatch相同，且AcceptOnMatch为true时才会匹配。

* LogLevelRangeFilter根据根据日志级别的范围进行过滤。

过滤条件包括LogLevelMin、LogLevelMax和AcceptOnMatch，只有当日志的LogLevel在LogLevelMin、LogLevelMax之间同时AcceptOnMatch为true时才会匹配。

* StringMatchFilter根据日志内容是否包含特定字符串进行过滤。

过滤条件包括StringToMatch和AcceptOnMatch，只有当日志包含StringToMatch字符串 且AcceptOnMatch为true时会匹配。

* DenyAllFilter则过滤掉所有消息。

过滤条件处理机制类似于Linux中IPTABLE的Responsibility chain机制，（即先deny、再allow）不过执行顺序刚好相反，后写的条件会被先执行，比如上面的例子，会首先执行filters.2的过滤条件，关闭所有过滤器，然后执行filters.1，仅匹配INFO信息。

#### Layout配置

可以选择impleLayout或PatternLayout，也可以选择不设置，如果不设置，会输出SimpleLayout格式的日志。

设置PatternLayout的语法：

log4cplus.appender.append\_1.layout=log4cplus::PatternLayout

举例：

log4cplus.appender.fileAppender2.layout=PatternLayout

log4cplus.appender.fileAppender2.layout.ConversionPattern=%d - %p - %m %n

### 3.3 Layout的转换标识符解释

PatterLayout支持的转换标识符主要包括：

（1）"%%"，转义为%, 即，std::string pattern = "%%" 时输出"%"

（2）"%c"，输出logger名称，比如std::string pattern ="%c" 时输出: "test\_logger.subtest"， 也可以控制logger名称的显示层次，比如"%c{1}"时输出"test\_logger"，其中数字表示层次。

（3）"%D"，显示本地时间，当std::string pattern ="%D" 时输出:"2004-10-16 18:55:45"，%d显示标准时间，所以当std::string pattern ="%d" 时输出 "2004-10-16 10:55:45" （因为北京时间位于东8区，差8个小时）。

（7）"%m"，输出原始信息，比如std::string pattern ="%m" 时输出: "teststr"，即上述代码中LOG4CPLUS\_DEBUG的第二个参数，这种实现机制可以确保原始信息被嵌入到带格式的信息中。

（8）"%n"，换行符，没什么好解释的。

（9）"%p"，输出LogLevel，比如std::string pattern ="%p" 时输出: "DEBUG"。

（12）格式对齐，比如std::string pattern ="%-10m"时表示左对齐，宽度是10，此时会输出"teststr "，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。

### 3.4 假如没有配置文件

上面说到该日志组件需要使用配置文件，但是如果没有配置文件，或者路径不对呢？也不用担心，因为log4cplus中做个默认处理。

当没有配置文件时，所有模块的日志都会保存到root\_default.log，每个文件最大为200K，最多有3个文件。

具体信息：

Appender：RollingFileAppender

Filename：root\_default.log

maxFileSize：200\*1024  
 maxBackupIndex：3