

iLOG3 日志函数库 参考手册

（如打开本文档后文档结构图混乱，可在打开文档时连续按 ESC 键）

版本修订

版本	日期	修订人	内容
1.0.0	2014-01-12	calvin	创建
1.0.1	2014-02-03	calvin	新增 日志句柄集合相关章节
1.0.2	2014-02-09	calvin	新增 配置文件接口层相关章节

目录索引

1	宏	6
1.1	错误码宏	6
1.2	日志输出类型宏	6
1.3	日志等级宏	6
1.4	行日志风格组合宏	7
1.5	日志选项组合宏	8
1.6	日志转档模式宏	8
2	日志句柄函数	9
2.1	管理日志句柄	9
2.1.1	创建日志句柄	9
2.1.2	销毁日志句柄	9
2.2	句柄环境设置	9
2.2.1	设置日志输出	9
2.2.2	设置当前日志过滤等级	10
2.2.3	设置行日志风格方案	10
2.3	高级句柄环境设置	11
2.3.1	设置日志句柄选项	11
2.3.2	设置自定义标签	11
2.3.3	设置日志转档模式	11
2.3.4	设置日志文件转档大小	12
2.3.5	设置日志文件转档大小的压迫系数	12
2.3.6	设置行日志缓冲区大小	12
2.3.7	设置十六进制块日志缓冲区大小	13
2.4	行日志输出	13
2.4.1	带日志等级的写行日志	13
2.4.2	写调试行日志	14
2.4.3	写普通信息行日志	14
2.4.4	写警告行日志	14
2.4.5	写错误行日志	15

2.4.6	写严重错误行日志.....	15
2.5	十六进制块日志输出.....	16
2.5.1	带日志等级的写十六进制块日志.....	16
2.5.2	写十六进制块调试日志.....	16
2.5.3	写十六进制普通信息块日志.....	17
2.5.4	写十六进制块警告日志.....	17
2.5.5	写十六进制块错误日志.....	18
2.5.6	写十六进制块严重错误日志.....	18
3	日志句柄集合函数.....	19
3.1	管理日志句柄集合.....	19
3.1.1	创建日志句柄集合.....	19
3.1.2	销毁日志句柄集合.....	19
3.2	管理日志句柄集合中的日志句柄.....	19
3.2.1	压入一个日志句柄到日志句柄集合中.....	19
3.2.2	从一个日志句柄集合中弹出一个指定标识的日志句柄.....	20
3.2.3	从一个日志句柄集合中查询一个指定标识的日志句柄.....	20
3.2.4	遍历一个日志句柄集合中所有日志句柄.....	20
4	配置文件接口层函数.....	21
4.1	日志句柄.....	21
4.1.1	从配置文件构建日志句柄.....	21
4.2	日志句柄集合.....	21
4.2.1	从配置文件构建日志句柄集合.....	21
5	配置辅助函数.....	22
5.1	属性转换函数.....	22
5.1.1	日志输出类型（字符串转换为整型）.....	22
5.1.2	日志等级类型（字符串转换为整型）.....	22
5.1.3	日志等级类型（整型转换为字符串）.....	22
5.1.4	行日志风格（字符串转换为整型）.....	23
5.1.5	日志选项（字符串转换为整型）.....	23
5.1.6	日志转档模式（字符串转换为整型）.....	23

5.1.7	日志缓冲区（字符串转换为整型）	24
6	简单配置文件属性列表.....	24

1 宏

1.1 错误码宏

宏名	宏说明
LOG_RETURN_ERROR_ALLOC	申请内存失败
LOG_RETURN_ERROR_INTERNAL	内部错误；不是你的错！
LOG_RETURN_ERROR_PARAMETER	参数错误
LOG_RETURN_ERROR_NOTSUPPORT	sorry，暂不支持
LOG_RETURN_ERROR_CREATEFILE	创建文件失败
LOG_RETURN_ERROR_OPENFILE	打开文件失败

1.2 日志输出类型宏

宏名	宏说明
LOG_OUTPUT_STDOUT	标准输出
LOG_OUTPUT_STDERR	标准错误输出
LOG_OUTPUT_SYSLOG	UNIX&Linux 的 syslog 或 Windows 的 WINDOWS EVENT
LOG_OUTPUT_FILE	文件
LOG_OUTPUT_CALLBACK	自定义日志输出回调函数

1.3 日志等级宏

宏名	宏说明
LOG_LEVEL_DEBUG	调试等级

LOG_LEVEL_INFO	普通信息等级
LOG_LEVEL_WARN	警告等级
LOG_LEVEL_ERROR	错误等级
LOG_LEVEL_FATAL	严重错误等级 往往由系统问题造成的不能继续工作的错误，该结束进程了，免得破坏数据
LOG_LEVEL_NOLOG	不需要输出日志

1.4 行日志风格组合宏

宏名	宏说明
LOG_STYLE_DATE	日期"YYYY-MM-DD"
LOG_STYLE_DATETIME	日期时间"YYYY-MM-DD hh:mm:ss"
LOG_STYLE_DATETIMEMS	日期时间毫秒"YYYY-MM-DD hh:mm:ss.6ms" (日期时间类宏互斥，优先自动选用信息量最多的)
LOG_STYLE_LOGLEVEL	日志等级
LOG_STYLE_PID	进程 id
LOG_STYLE_TID	线程 id
LOG_STYLE_SOURCE	"源代码文件名:源代码行号"
LOG_STYLE_FORMAT	应用日志段
LOG_STYLE_NEWLINE	换行
LOG_STYLE_CUSTLABEL1	自定义标签 1
LOG_STYLE_CUSTLABEL2	自定义标签 2
LOG_STYLE_CUSTLABEL3	自定义标签 3
LOG_STYLE_CALLBACK	(使用自定义行日志风格回调函数)
LOG_STYLE_...	(等着你来扩展)

1.5 日志选项组合宏

宏名	宏说明
LOG_OPTION_OPEN_AND_CLOSE	每次都打开日志、写日志、关闭日志
LOG_OPTION_CHANGE_TEST	侦测文件变动
LOG_OPTION_OPEN_ONCE	日志打开一次不关闭（以上三项不能组合）
LOG_OPTION_SET_OUTPUT_BY_FILENAME	自动根据文件名覆盖输出类型 如果文件名前面存在"#stdout#"前缀，去掉前缀并覆盖输出类型为 LOG_OUTPUT_STDOUT 如果文件名前面存在"#stderr#"前缀，去掉前缀并覆盖输出类型为 LOG_OUTPUT_STDERR 如果文件名前面存在"#syslog#"前缀，去掉前缀并覆盖输出类型为 LOG_OUTPUT_SYSLOG
LOG_OPTION_FILENAME_APPEND_DOT_LOG	日志输出文件名后自动加上".log"

1.6 日志转档模式宏

宏名	宏说明
LOG_ROTATEMODE_NONE	不转档
LOG_ROTATEMODE_SIZE	按日志文件大小转档，和函数 SetLogRotateSize 配合使用 转档文件名格式"原日志文件名.序号"
LOG_ROTATEMODE_PER_DAY	按每天转档 转档文件名格式"原日志文件名.年年年年月月日日"
LOG_ROTATEMODE_PER_HOUR	按小时转档 转档文件名格式"原日志文件名.年年年年月月日日_小时"

2 日志句柄函数

2.1 管理日志句柄

2.1.1 创建日志句柄

函数原型: LOG *CreateLogHandle();

参数说明: (无)

返回值: 成功, 返回新创建的日志句柄
失败, 返回 NULL, 一般为申请内存失败

备注:

2.1.2 销毁日志句柄

函数原型: void DestroyLogHandle(LOG *g);

参数说明: 输入 LOG *g 日志句柄

返回值: (无)

备注:

2.2 句柄环境设置

2.2.1 设置日志输出

函数原型: int SetLogOutput (LOG *g , int output , char *log_pathfilename , funcOpenLog
*pfuncOpenLogFirst , funcOpenLog *pfuncOpenLog , funcWriteLog *pfuncWriteLog ,
funcChangeTest *pfuncChangeTest , funcCloseLog *pfuncCloseLog , funcCloseLog
*pfuncCloseLogFinally);

参数说明: 输入 LOG *g 日志句柄

输入 int output 日志输出类型, 见日志输出宏

输入 `char *log_pathfilename` 日志文件名，允许内嵌“\$....\$”以表示环境变量

输入 `LOG_OPTION_OPEN_ONCE` 或 `LOG_OPTION_CHANGE_TEST` 模式下，初始化时调用 `pfuncOpenLogFirst`，然后不停的 `pfuncWriteLog`，最后调用 `pfuncCloseLogFinally`

`LOG_OPTION_OPEN_AND_CLOSE` 模式下，写日志过程顺序调用 `pfuncOpenLog`、`pfuncWriteLog`、`pfuncCloseLog`

返回值： 成功，返回 0

失败，返回非 0，见错误码宏

备注：

2.2.2 设置当前日志过滤等级

函数原型： `int SetLogLevel(LOG *g , int log_level);`

参数说明： 输入 `LOG *g` 日志句柄

输入 `int log_level` 日志过滤等级，见日志等级宏

返回值： 返回 0

备注：

2.2.3 设置行日志风格方案

函数原型： `int SetLogStyles(LOG *g , long log_styles , funcLogStyle *pfuncLogStyle);`

参数说明： 输入 `LOG *g` 日志句柄

输入 `long log_styles` 行日志风格宏组合，见行日志风格组合宏

输入 `funcLogStyle *pfuncLogStyle` 作为填充整行的回调函数

返回值： 返回 0

备注：

2.3 高级句柄环境设置

2.3.1 设置日志句柄选项

函数原型: `int SetLogOptions(LOG *g , int log_options);`

参数说明: 输入 `LOG *g` 日志句柄

输入 `int log_options` 选项, 见日志选项组合宏

返回值: 成功, 返回 0

失败, 返回非 0, 见错误码宏

备注:

2.3.2 设置自定义标签

函数原型: `int SetLogCustLabel(LOG *g , int index , char *cust_label);`

参数说明: 输入 `LOG *g` 日志句柄

输入 `int index` 标签索引, 从 1 到 `LOG_MAXCNT_CUST_LABEL`

输入 `char *cust_label` 标签值, 最长 `LOG_MAXLEN_CUST_LABEL` 个字符

返回值: 成功, 返回 0

失败, 返回非 0, 见错误码宏

备注:

2.3.3 设置日志转档模式

函数原型: `int SetLogRotateMode(LOG *g , int rotate_mode);`

参数说明: 输入 `LOG *g` 日志句柄

输入 `int rotate_mode` 转档模式, 见日志转档模式宏

返回值: 成功, 返回 0

失败, 返回非 0, 见错误码宏

备注:

2.3.4 设置日志文件转档大小

函数原型: `int SetLogRotateSize(LOG *g , long log_rotate_size);`

参数说明: 输入 `LOG *g` 日志句柄

输入 `long log_rotate_size` 当按日志文件大小转档时, 设置临界大小

返回值: 成功, 返回 0

失败, 返回非 0, 见错误码宏

备注:

2.3.5 设置日志文件转档大小的压迫系数

函数原型: `int SetLogRotatePressureFactor(LOG *g , long pressure_factor);`

参数说明: 输入 `LOG *g` 日志句柄

输入 `long pressure_factor` 建议值为"写该日志文件的进程数*线程数

*2", 值越大提高转档文件大小精确度但影响日志输出性能

返回值: 成功, 返回 0

失败, 返回非 0, 见错误码宏

备注:

2.3.6 设置行日志缓冲区大小

函数原型: `int SetLogBufferSize(LOG *g , long log_bufsize , long max_log_bufsize);`

参数说明: 输入 `LOG *g` 日志句柄

输入 `long log_bufsize` 行日志缓冲区大小, 当为-1 时不设置

输入 `long max_log_bufsize` 最大行日志缓冲区大小, 当为-1 时不设置

返回值: 成功, 返回 0

失败, 返回非 0, 见错误码宏

备注:

2.3.7 设置十六进制块日志缓冲区大小

函数原型: `int SetHexLogBufferSize(LOG *g , long hexlog_bufsize , long max_log_bufsize);`

参数说明: 输入 `LOG *g` 日志句柄

输入 `long hexlog_bufsize` 块日志缓冲区大小, 当为-1 时不设置

输入 `long max_log_bufsize` 最大行日志缓冲区大小, 当为-1 时不设置

返回值: 成功, 返回 0

失败, 返回非 0, 见错误码宏

备注:

2.4 行日志输出

2.4.1 带日志等级的写行日志

函数原型: `int WriteLog(LOG *g , char *c_filename , long c_fileline , int log_level , char *format , ...);`

参数说明: 输入 `LOG *g` 日志句柄

输入 `char *c_filename` 当前源代码文件名 `__FILE__`

输入 `long c_fileline` 当前源代码行号 `__LINE__`

输入 `int log_level` 当前日志等级, 见日志等级宏

输入 `char *format , ...` 应用日志段, 可变参数

返回值: 成功, 返回 0

失败, 返回非 0, 见错误码宏

备注:

2.4.2 写调试行日志

函数原型: `int DebugLog(LOG *g , char *c_filename , long c_fileline , char *format , ...);`

参数说明: 输入 `LOG *g` 日志句柄
输入 `char *c_filename` 当前源代码文件名__FILE__
输入 `long c_fileline` 当前源代码行号__LINE__
输入 `char *format , ...` 应用日志段, 可变参数

返回值: 成功, 返回 0
失败, 返回非 0, 见错误码宏

备注:

2.4.3 写普通信息行日志

函数原型: `int InfoLog(LOG *g , char *c_filename , long c_fileline , char *format , ...);`

参数说明: 输入 `LOG *g` 日志句柄
输入 `char *c_filename` 当前源代码文件名__FILE__
输入 `long c_fileline` 当前源代码行号__LINE__
输入 `char *format , ...` 应用日志段, 可变参数

返回值: 成功, 返回 0
失败, 返回非 0, 见错误码宏

备注:

2.4.4 写警告行日志

函数原型: `int WarnLog(LOG *g , char *c_filename , long c_fileline , char *format , ...);`

参数说明: 输入 `LOG *g` 日志句柄
输入 `char *c_filename` 当前源代码文件名__FILE__
输入 `long c_fileline` 当前源代码行号__LINE__
输入 `char *format , ...` 应用日志段, 可变参数

返回值： 成功，返回 0
失败，返回非 0，见错误码宏

备注：

2.4.5 写错误行日志

函数原型： `int ErrorLog(LOG *g , char *c_filename , long c_fileline , char *format , ...);`

参数说明： 输入 `LOG *g` 日志句柄
输入 `char *c_filename` 当前源代码文件名 `__FILE__`
输入 `long c_fileline` 当前源代码行号 `__LINE__`
输入 `char *format , ...` 应用日志段，可变参数

返回值： 成功，返回 0
失败，返回非 0，见错误码宏

备注：

2.4.6 写严重错误行日志

函数原型： `int FatalLog(LOG *g , char *c_filename , long c_fileline , char *format , ...);`

参数说明： 输入 `LOG *g` 日志句柄
输入 `char *c_filename` 当前源代码文件名 `__FILE__`
输入 `long c_fileline` 当前源代码行号 `__LINE__`
输入 `char *format , ...` 应用日志段，可变参数

返回值： 成功，返回 0
失败，返回非 0，见错误码宏

备注：

2.5 十六进制块日志输出

2.5.1 带日志等级的写十六进制块日志

函数原型: `int WriteHexLog(LOG *g , char *c_filename , long c_fileline , int log_level , char *buffer , long buflen , char *format , ...);`

参数说明:

输入	LOG *g	日志句柄
输入	char *c_filename	当前源代码文件名__FILE__
输入	long c_fileline	当前源代码行号__LINE__
输入	int log_level	当前日志等级，见日志等级宏
输入	char *buffer	块日志缓冲区
输入	long buflen	块日志缓冲区长度
输入	char *format , ...	应用日志段，可变参数

返回值: 成功，返回 0
失败，返回非 0，见错误码宏

备注:

2.5.2 写十六进制块调试日志

函数原型: `int DebugHexLog(LOG *g , char *c_filename , long c_fileline , char *buffer , long buflen , char *format , ...);`

参数说明:

输入	LOG *g	日志句柄
输入	char *c_filename	当前源代码文件名__FILE__
输入	long c_fileline	当前源代码行号__LINE__
输入	char *buffer	块日志缓冲区
输入	long buflen	块日志缓冲区长度
输入	char *format , ...	应用日志段，可变参数

返回值: 成功，返回 0
失败，返回非 0，见错误码宏

备注:

2.5.3 写十六进制普通信息块日志

函数原型: `int InfoHexLog(LOG *g , char *c_filename , long c_fileline , char *buffer , long buflen , char *format , ...);`

参数说明:

输入	LOG *g	日志句柄
输入	char *c_filename	当前源代码文件名__FILE__
输入	long c_fileline	当前源代码行号__LINE__
输入	char *buffer	块日志缓冲区
输入	long buflen	块日志缓冲区长度
输入	char *format , ...	应用日志段, 可变参数

返回值:

成功, 返回 0

失败, 返回非 0, 见错误码宏

备注:

2.5.4 写十六进制块警告日志

函数原型: `int WarnHexLog(LOG *g , char *c_filename , long c_fileline , char *buffer , long buflen , char *format , ...);`

参数说明:

输入	LOG *g	日志句柄
输入	char *c_filename	当前源代码文件名__FILE__
输入	long c_fileline	当前源代码行号__LINE__
输入	char *buffer	块日志缓冲区
输入	long buflen	块日志缓冲区长度
输入	char *format , ...	应用日志段, 可变参数

返回值:

成功, 返回 0

失败, 返回非 0, 见错误码宏

备注:

2.5.5 写十六进制块错误日志

函数原型: `int ErrorHexLog(LOG *g , char *c_filename , long c_fileline , char *buffer , long buflen , char *format , ...);`

参数说明:

输入	LOG *g	日志句柄
输入	char *c_filename	当前源代码文件名__FILE__
输入	long c_fileline	当前源代码行号__LINE__
输入	char *buffer	块日志缓冲区
输入	long buflen	块日志缓冲区长度
输入	char *format , ...	应用日志段, 可变参数

返回值: 成功, 返回 0
 失败, 返回非 0, 见错误码宏

备注:

2.5.6 写十六进制块严重错误日志

函数原型: `int FatalHexLog(LOG *g , char *c_filename , long c_fileline , char *buffer , long buflen , char *format , ...);`

参数说明:

输入	LOG *g	日志句柄
输入	char *c_filename	当前源代码文件名__FILE__
输入	long c_fileline	当前源代码行号__LINE__
输入	char *buffer	块日志缓冲区
输入	long buflen	块日志缓冲区长度
输入	char *format , ...	应用日志段, 可变参数

返回值: 成功, 返回 0
 失败, 返回非 0, 见错误码宏

备注:

3 日志句柄集合函数

3.1 管理日志句柄集合

3.1.1 创建日志句柄集合

函数原型: LOGS *CreateLogsHandle();

参数说明: (无)

返回值: 成功, 返回新创建的日志句柄集合
失败, 返回 NULL, 一般为申请内存失败

备注:

3.1.2 销毁日志句柄集合

函数原型: void DestroyLogsHandle(LOGS *g);

参数说明: 输入 LOGS *gs 日志句柄集合

返回值: (无)

备注:

3.2 管理日志句柄集合中的日志句柄

3.2.1 压入一个日志句柄到日志句柄集合中

函数原型: int AddLogToLogs(LOGS *gs, char *g_id, LOG *g);

参数说明: 输入 LOGS *gs 日志句柄集合

输入 char *g_id 要压入的日志句柄标识, 用于后面查询和弹出

输入 LOG *g 要压入的日志句柄

返回值: 成功, 返回 0
失败, 返回非 0, 见错误码宏

备注:

3.2.2 从一个日志句柄集合中弹出一个指定标识的日志句柄

函数原型: LOG *RemoveOutLogFromLogs(LOGS *gs , char *g_id);

参数说明: 输入 LOGS *gs 日志句柄集合
 输入 char *g_id 要压入的日志句柄标识, 用于后面查询和弹出
 输入 LOG *g 要压入的日志句柄

返回值: 成功, 返回弹出的日志句柄
 失败, 返回 NULL

备注: 弹出日志句柄只是解除与日志句柄集合的关系, 并没有销毁之, 需用户自己销毁掉。或者销毁日志句柄集合 DestroyLogsHandle 连带销毁集合中所有日志句柄。

3.2.3 从一个日志句柄集合中查询一个指定标识的日志句柄

函数原型: LOG *GetLogFromLogs(LOGS *gs , char *g_id);

参数说明: (同上)

返回值: (同上)

备注:

3.2.4 遍历一个日志句柄集合中所有日志句柄

函数原型: int TravelLogFromLogs(LOGS *gs , long *p_index , char **pp_g_id , LOG **pp_g);

参数说明: 输入 LOGS *gs 日志句柄集合
 输入 long *p_index 第一次声明一个 long 型变量初始化为
LOG_TRAVELLOG_INDEX_INIT 传进去, 作为遍历跟踪序号

 输出 char **pp_g_id 每次遍历成功后设置的日志句柄标识字符串指针

 输出 LOG **pp_g 每次遍历成功后设置的日志句柄指针

返回值： 成功，返回 0
遍历结束，返回 LOGS_RETURN_INFO_NOTFOUND
失败，返回其它

备注：

4 配置文件接口层函数

4.1 日志句柄

4.1.1 从配置文件构建日志句柄

函数原型： LOG *CreateLogHandleFromConfig(char *config_filename , char *postfix);

参数说明： 输入 最终的配置文件名可以是 config_filename，也可以由 config_filename 和 postfix 拼接而成

返回值： 成功，返回新创建的日志句柄
失败，返回 NULL，一般为申请内存失败

备注：

4.2 日志句柄集合

4.2.1 从配置文件构建日志句柄集合

函数原型： LOGS *CreateLogsHandleFromConfig(char *config_filename , char *postfix);

参数说明： 输入 最终的配置文件名可以是 config_filename，也可以由 config_filename 和 postfix 拼接而成

返回值： 成功，返回新创建的日志句柄集合
失败，返回 NULL，一般为申请内存失败

备注：

5 配置辅助函数

5.1 属性转换函数

5.1.1 日志输出类型（字符串转换为整型）

函数原型: `int ConvertLogOutput_atoi(char *output_desc , int *p_log_output);`

参数说明: 输入 `char *output_desc` 日志输出类型字符串

输出 `int *p_log_output` 实际使用的日志输出类型值

返回值: 成功, 返回 0

失败, 返回非 0, 一般都是输入参数无效造成的

备注:

5.1.2 日志等级类型（字符串转换为整型）

函数原型: `int ConvertLogLevel_atoi(char *log_level_desc , int *p_log_level);`

参数说明: 输入 `char *log_level_desc` 日志等级类型字符串

输出 `int *p_log_level` 实际使用的日志等级值

返回值: 成功, 返回 0

失败, 返回非 0, 一般都是输入参数无效造成的

备注:

5.1.3 日志等级类型（整型转换为字符串）

函数原型: `int ConvertLogLevel_itoa(int log_level , char **log_level_desc);`

参数说明: 输入 `int log_level_desc` 实际使用的日志等级值

输出 `char **log_level_desc` 日志等级类型字符串指针

返回值： 成功，返回 0
失败，返回非 0，一般都是输入参数无效造成的

备注：

5.1.4 行日志风格（字符串转换为整型）

函数原型： `int ConvertLogStyle_atol(char *line_style_desc , long *p_line_style);`

参数说明： 输入 `char *line_style_desc` 行日志风格字符串
输出 `long *p_line_style` 实际使用的行日志风格值

返回值： 成功，返回 0
失败，返回非 0，一般都是输入参数无效造成的

备注： 此函数只转换单个风格值

5.1.5 日志选项（字符串转换为整型）

函数原型： `int ConvertLogOption_atol(char *log_option_desc , long *p_log_option);`

参数说明： 输入 `char *log_option_desc` 日志选项字符串
输出 `long *p_log_option` 实际使用的日志选项值

返回值： 成功，返回 0
失败，返回非 0，一般都是输入参数无效造成的

备注： 此函数只转换单个选项值

5.1.6 日志转档模式（字符串转换为整型）

函数原型： `int ConvertLogRotateMode_atoi(char *rotate_mode_desc , int *p_rotate_mode);`

参数说明： 输入 `char *rotate_mode_desc` 日志转档模式字符串
输出 `int *p_rotate_mode` 实际使用的日志转档模式值

返回值： 成功，返回 0

失败，返回非 0，一般都是输入参数无效造成的

备注：

5.1.7 日志缓冲区（字符串转换为整型）

函数原型： `int ConvertBufferSize_atol(char *bufsize_desc , long *p_bufsize);`

参数说明： 输入 `char *bufsize_desc` 日志缓冲区大小字符串

输出 `long *p_bufsize` 实际使用的日志缓冲区大小值

返回值： 成功，返回 0

失败，返回非 0，一般都是输入参数无效造成的

备注： 日志缓冲区大小字符串支持后缀“B”、“KB”、“MB”、“GB”

6 简单配置文件属性列表

属性语法	说明 (基本与对应宏同名，去掉左前缀)
output 日志输出类型 [日志文件名]	
level 日志过滤等级	
custlabel1~custlabel3 自定义标签值	自定义标签 1~标签 3
styles 行日志风格组合	用' '组合，前后用双引号引起来
options 日志选项组合	用' '组合，前后用双引号引起来；不设置缺省为 CHANGE_TEST
rotate_mode 日志转档方案	
rotate_size 日志转档大小	当日志转档方案为按大小转档时起作用；支持单位后缀，如“GB”、“MB”、“KB”
rotate_pressure_factor 按大小转档压迫系数	
log_buffersize 行日志缓冲区初始大小 [行日志缓冲区最大大小]	支持单位后缀

hexlog_buffersize 十六进制块日志缓冲区初始 大小 [十六进制块日志缓冲区最大大小]	支持单位后缀
--	--------

示例

```
id          hello
output      FILE test_logconf.log
level       INFO
styles      DATETIME|LOGLEVEL|PID|TID|SOURCE|FORMAT|NEWLINE
options     CHANGE_TEST
rotate_mode SIZE
rotate_size 10MB
log_bufsize 1MB 5MB
```

或

```
id          hello
output      FILE test_logconf.log
level       INFO
styles      DATETIME|LOGLEVEL|PID|TID|SOURCE|FORMAT|NEWLINE
options     CHANGE_TEST
rotate_mode SIZE
rotate_size 10MB
log_bufsize 1MB 5MB
hexlog_bufsize 5MB

id          stdout
output      STDOUT
level       INFO
styles      DATETIME|LOGLEVEL|PID|TID|SOURCE|FORMAT|NEWLINE
```