wdt3385的专栏

keepalived配置文件解析系列之(三)配置文件解析过程

分类： keepalived 2013-06-27 15:19 148人阅读 评论(0) 收藏 举报

如(一)所言， keepalived在配置文件解析方面拥有非常灵活的方式， 采用关键字分层（每层的关键字数量不限，且关键字的层次也不限制）的方法进行组织一个配置文件， 且支持平行或者嵌套地include多个其它配置文件语句和正则表达式记法的配置文件名。文(二)中介绍了keepalived关键字的存储和相关的操作，下面将具体分析怎么样解析配置文件。

一、入口点及全局流程

位于parser.c文件中的init_data(char *conf_file, vector (*init_keywords) (void))函数是解析配置文件的入口点， 该函数第一个参数为需要解析的配置文件名(可以由正则表达式构成，如*.conf)， 第二个参数为函数指针，指向关键字组织结构（实际取值为check_init_keywords 或者vrrp_init_keywords）。

解析配置文件过程中用到的几个全局变量包括current_keywords（当前关键字列表）、keywords(层次为0的关键字列表， 整个关键字组织的最顶层)、current_stream(当前打开的文件的流， 即文件指针， 指向当前打开的文件，因为keepalived支持嵌套的文件，所以必须设置这样的变量)、current_conf_file(当前解析的文件的名字， 同上， 是为支持嵌套的配置文件而设立的)、kw_level(当前关键字列表的层次， 处理下一层关键字列表之前递增， 回退到上一层关键字列表时递减)。

全局流程如下: 在init_data()中，调用read_conf_file(conf_file)， 把要解析的配置文件的名字传递给read_conf_file()函数；read_conf_file()函数先利用glob库取得所有的conf_file（尤其是带正则表达式符号的名字），然后依次根据 当前关键字列表current_keywords， 调用process_stream(current_keywords)进行配置文件的解析。解析 的方法是依次读取配置文件中的一行， 把该行转化为一个字符串数组， 再根据该字符串数组的第一个元素（即该行的第一个字符串）进行以下四种情况的处理: 1)若该字符串是井号或者感叹号(#or!) 表明这是一行注释， 不做处理， 继续读取下一行； 2)若该字符串为右大括号（"}"），且不是层次0的关键字列表, 意味着当前层次的关键字解析结束， 直接退出process_stream()函数； 3)若该字符串为"include"， 说明包含嵌套配置文件， 则在保存了当前的文件流和配置文件名字后，递归调用read_conf_file(conf_file)处理include进行来的配置文件， 处理结束后， 再恢复之前保存的文件流和配置文件名字， 继续原配置文件的处理； 4)若该字符串不属于上述三种情况， 那它预期是一个关键字， 则遍历当前关键字列表current_keywords， 并逐一和该字符串进行配置， 若找到了确切的关键字， 则调用 该关键字的handler()进行处理，若该关键字还关联着下一层次的关键字列表， 则先递增关键字列表层次变量kw_level， 再递归调用process_stream(keyword->sub)处理下一层次的关键字列表， 处理结束后再递减kw_level。 若该字符串不属于于当前关键字列表中的关键字， 则忽略该行。配置文件的解析结束发生在读取到EOF， 而 一个合法的配置文件，一般都 是从层次0开始处理关键字， 中间会依次处理下一层次的关键字列表， 而最终又回归到层次0的关键字列表中。

二、函数的具体分析

根据上面的总体流程， 在整个解析 配置文件的过程中有5个关键的函数: init_data(char *conf_file, vector (*init_keywords) (void))、 read_conf_file(char *conf_file)、 process_stream(vector keywords_vec)、 read_line(char *buf, int size) 和check_include(char *buf)。下面分别对这个5个函数 进行分析。

init_data(char *conf_file, vector (*init_keywords) (void))该函数首先调用vector_alloc()分配一个关键字列表， 并令全局变量keywords指向该列表， 然后设置当前的关键字列表current_keywords为keywords， 表示从层次0的关键字列表开始解析配置文件， 接着调用read_conf_file(conf_file)进行具体的配置文件解析， 结束后，释放一开始分配的关键字列表keywords。

read_conf_file(char *conf_file)该函数首先调用glob(conf_file, 0, NULL, &globbuf)取得配置文件名字conf_file对应的所有配置文件， 然后依次取出每个配置文件进行处理。每个配置文件具体是这样处理的：先用fopen（）打开该配置文件， 并保存文件流到当前文件流变量current_stream中， 并保存当前配置文件名字到current_conf_file中， 同时为了在被递归调用后可以回退到当前的配置文件继续处理， 它还保存了当前的路径到prev_path中， 然后调用process_stream(current_keywords)根据当前的关键字层次解析配置文件。直到由glob()取得的所有的配置文件解析完毕才退出read_conf_file()函数。

process_stream(vector keywords_vec)该函数以当前关键字列表为参数， 并从当前的文件流current_stream中取得配置文件中的内容， 进行相应的处理。 具体地， 先保存当前的关键字列表到prev_keywords中（同样地， 为了后面的递归调用结束后可以回退到当前的处理）， 然后循环调用read_line(buf, MAXBUF)从当前 文件流中读取一行， 若读取到的是EOF， 则退出循环；否则调用alloc_strvec(buf)把读取的该行数据转化为字符串列表， 并取得该列表的第一个元素（即行首的字符串）到str中，若str是大括号("}"), 且当前 的关键字列表层次变量kw_level大于0， 则退出循环， 结束当前

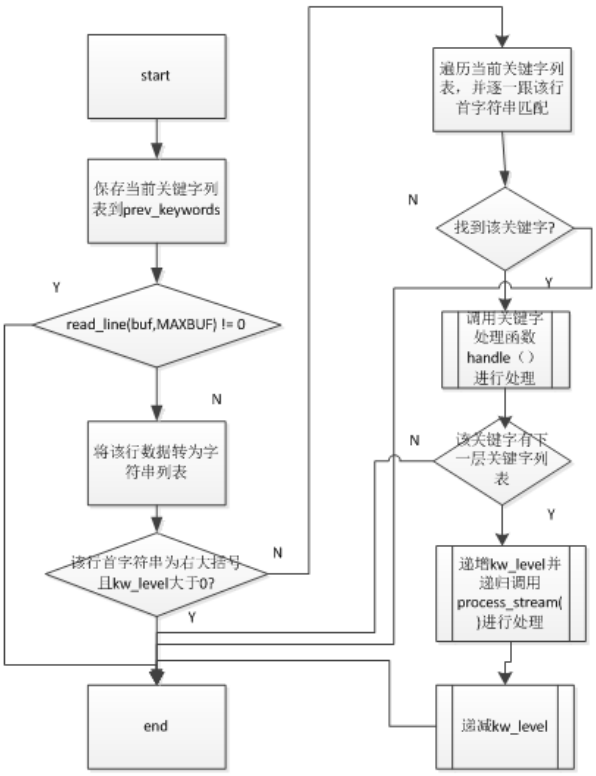
个人资料

wdt3385

访问： 338971次
积分： 5988分
排名： 第1052名

原创： 78篇 转载： 1184篇
译文： 0篇 评论： 21条

层次的关键字处理； 否则依次遍历当前关键字列表， 若逐一匹配str， 若找到与str相同的关键字， 则调用该关键字的handler()处理该行数据，若该关键字存在下一层次关键字列表， 则递增关键字列表变量kw_level， 再递归调用process_stream(keyword->sub)处理下一层次的关键字列表， 处理结束后，再递减关键字列表变量kw_level。实际上，一个process_stream()对应着一层关键字列表处理（这里要注意一点， 即同一层次的关键字列表可以有多个， 只要它们的上一层关键字不同即可）。（可参考下面的流程图）



process_stream()函数流程

read_line(char *buf, int size)该函数用来从当前文件流中读取一行数据，并调用check_include(检测是否以"include"开头，若是的话，进入"include"的情况处理(下面的check_include()函数详细介绍)，且不返回该行数据给调用者， 继续读取下一行数据。若读到非"include"语句则返回该行数据(通过参数), 若遇到EOF， 则返回0(return 0)。

check_include(char *buf)该函数检查该行数据是否包含"include"， 若不含"include"直接返回0， 否则会进一步解析include进来的文件。 具体地， 调用alloc_strvec(buf) 把该行数据转化为字符串列表， 然后判断行首的字符串是否为"include"， 是的话， 先保存当前的文件流current_stream到prev_stream， 当前的配置文件current_conf_file到prev_conf_file，当前的路径到prev_path中， 然后递归调用read_conf_file(conf_file)函数解析include进来的配置文件，解析结束后， 还原current_stream, current_conf_file为之前保存的值。

三、小结

keepalived在解析配置文件时充分地利用了递归， 如process_stream()函数中，会根据当前的关键字是否有下一层的关键字列表递归调用process_stream()；在check_include()函数中， 会递归调用read_conf_file()去解析新的配置文件。正是这些递归的设计， 使它在解析配置文件方面显得很强大。

然而，事情并不总是完美的。 keepalived在配置文件解析方面也存在一些问题， 如没有限制配置文件的嵌套层次（考虑这样一个场景include * 或者自包含），没有对关键字进行严格检查（相应的出错提示可能不够详细）等等， 这些问题其实也可以在现有的代码框架里面进行修改增强的。

上一篇 keepalived配置文件解析系列之(二)keyword存储的设计与实现
下一篇 keepalived 添加pop3_check模块(一)

管理 (1)
QT (46)
Eric4 (4)
S (0)
性能 (4)
代码走查 (1)
WAF (1)
邮件 (1)
ARP (2)
ACM (5)
网络 (0)
log (0)
logstash (3)
防篡改 (1)
PyRTF (1)
Tengine (1)
IP地址库 (3)
防盗链 (1)

文章存档

2014年06月 (1)
2014年05月 (3)
2014年04月 (4)
2014年03月 (3)
2014年02月 (6)

展开

阅读排行

一个完整的安装程序实例 (3608)
一个完整的安装程序实例 (3451)
一个完整的安装程序实例 (3164)
关于Installshield里一些 (3157)
一个完整的安装程序实例 (3109)
一个完整的安装程序实例 (2933)
如何为工程添加一个Insta (2448)
PKI/CA (1896)
Linux进程间通信之共享 (1642)
Bolt引擎内置的元对象介 (1609)

评论排行

Bolt引擎内置的元对象介 (3)
Thinkphp的RBAC，基于 (2)
(转)什么是套接字(Socke (2)
Mongoose笔记——mair (2)
PyQt4 精彩实例分析* 实 (2)
金山开源安全卫士全套代 (1)
套接字编程中,select:bad (1)
关于Installshield里一些 (1)
头文件.h和源文件.cpp的 (1)
linux修改系统时间 (1)

推荐文章

最新评论

套接字编程中,select:bad file des
xiaoyan19891227: 貌似没有解决我的问题啊

Thinkphp的RBAC，基于角色的
yellowxiaotian: 3Q

查看评论

暂无评论

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker
OpenStack VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC
WAP jQuery BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML
LBS Unity Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra
CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App
SpringSide Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP
HBase Pure Solr Angular Cloud Foundry Redis Scala Django Bootstrap