C语言插件机制(下)

作者 正文

发表时间：2010-08-25

- abruzzi
- 等级: 💎💎

-
- 性别: ♂
- 文章: 340
- 积分: 1270
- 来自: 西安
- 我现在离线

C

前言

上一篇文章简单介绍了*NIX下的动态库的使用，我们在这篇文章中实现一个计算器，计算器程序calc本身不做运算，只是将操作数传递给具体的插件(adder, suber, muler, diver)来完成实际运算。首先，计算器根据插件配置文件plugin.xml来确定插件的位置，名称，入口符号的定义，然后依次调用各个插件完成计算。

插件列表

文中涉及到的插件定义在plugin.xml中，文档结构如下：

Xml代码

锁定老帖子 主题：C语言插件机制(下)
精华帖 (7) :: 良好帖 (0) :: 新手帖 (0) :: 隐藏帖 (0)

作者 正文

```
<plugins>
  <plugin name="adder">
    <library path="/home/juntao/.libs/adder.so">
    </library>
    <entry name="add">
    </entry>
  </plugin>
  <plugin name="suber">
    <library path="/home/juntao/.libs/suber.so">
    </library>
    <entry name="sub">
    </entry>
  </plugin>
  <plugin name="muler">
    <library path="/home/juntao/.libs/muler.so">
    </library>
    <entry name="mul">
    </entry>
  </plugin>
  <plugin name="diver">
    <library path="/home/juntao/.libs/diver.so">
    </library>
    <entry name="div">
    </entry>
  </plugin>
</plugins>
```

每个插件为一个plugin标签，plugin标签中包含library, entry两个字标签，分别定义动态库文件的路径及名称和插件函数的入口。为了简便，我们不重复设计list及xml解析，这里使用libxml2作为xml的分析器，GLIB中的GSList(单链表)来作为插件列表的链表对象。

每一个插件在C语言中的定义如下，非常简单(plugin.h)

C代码

```
#ifndef _PLUGIN_H_
#define _PLUGIN_H_

typedef struct{
    char name[64];
    char path[256];
    char entry[128];
    int version;
```

作者 正文

```
 9. }Plugin;
10.
11. #endif
```

这里为了行文方便，Plugin结构中的字符串为静态尺寸。

计算器

计算器调用parseconf模块中的load_plugins将plugin.xml中定义的Plugin加载进一个GSList，以备后用：

C代码

```
1. #include "plugin.h"
2.
3. extern int load_plugins(char *config, GSList **list);
```

插件中的函数原型应该符合接口定义：

C代码

```
1. //pointer to function, which return a double, and get 2 double as input
2. double (*pfunc)(double a, double b);
```

计算器的主要代码如下：

C代码

```
1. int calc_test(double a, double b){
2.     GSList *list = NULL, *it = NULL;
3.     Plugin *pl = NULL;
4.
5.     //insert a null node into list at first
6.     list = g_slist_append(list, NULL);
7.
8.     int code = 0;
9.     double result;
```

锁定老帖子 主题：C语言插件机制(下)
精华帖 (7) :: 良好帖 (0) :: 新手帖 (0) :: 隐藏帖 (0)

作者 正文

```
10.
11.  //load plugin defined in plugin.xml into list
12.  load_plugins("plugin.xml", &list);
13.
14.  for(it = list; it != NULL; it = it->next){
15.      pl = (Plugin *)it->data;
16.      if(pl == NULL){
17.          continue;
18.      }else{
19.          //open the library
20.          flib = dlopen(pl->path, RTLD_LAZY);
21.          dlError = dlerror();
22.
23.          if(dlError){
24.              fprintf(stderr, "open %s failed\n", pl->name);
25.              g_slist_free(list);
26.              return -1;
27.          }
28.
29.          //get the entry
30.          *(void **)(&pfunc) = dlsym(flib, pl->entry);
31.          dlError = dlerror();
32.          if(dlError){
33.              fprintf(stderr, "find symbol %s failed\n", pl->entry);
34.              g_slist_free(list);
35.              return -1;
36.          }
37.
38.          //call the function
39.          result = (*pfunc)(a, b);
40.          printf("%s(%f, %f) = %f\n", pl->entry, a, b, result);
41.
42.          //then close it
43.          code = dlclose(flib);
44.          dlError = dlerror();
45.
46.          if(code){
47.              fprintf(stderr, "close lib error\n");
48.              g_slist_free(list);
49.              return -1;
50.          }
51.      }
52.  }
53.
54.  g_slist_free(list);
55.  return 0;
56. }
```

作者 正文

首先，定义一个GSList，然后将其传递给load_plugins，load_plugins解析plugin.xml，然后填充list返回，calc_test遍历插件列表，并调用每一个插件定义的entry.

除法器

我们来看一个具体的插件：做除法的模块

C代码

```
1. #include <stdio.h>
2.
3. double div(double a, double b){
4.     if(b == 0){
5.         fprintf(stderr, "div zero error\n");
6.         return -1;
7.     }else{
8.         return a / b;
9.     }
10. }
```

diver.c在编译之后，生成diver.so，将其置于plugin.xml定义的位置处即可。

运行结果如下图所示：

锁定老帖子 主题：C语言插件机制(下)
精华帖 (7) :: 良好帖 (0) :: 新手帖 (0) :: 隐藏帖 (0)

作者 正文

```
[juntao@juntao calc]$ ./calc
please input 2 numbers
[juntao@juntao calc]$ ./calc 135.9 23.3
input = 135.900000, 23.300000
add(135.900000, 23.300000) = 159.200000
sub(135.900000, 23.300000) = 112.600000
mul(135.900000, 23.300000) = 3166.470000
div(135.900000, 23.300000) = 5.832618
[juntao@juntao calc]$ ./calc 29 0
input = 29.000000, 0.000000
add(29.000000, 0.000000) = 29.000000
sub(29.000000, 0.000000) = 29.000000
mul(29.000000, 0.000000) = 0.000000
div zero error
div(29.000000, 0.000000) = -1.000000
[juntao@juntao calc]$ ./calc 100 25
input = 100.000000, 25.000000
add(100.000000, 25.000000) = 125.000000
sub(100.000000, 25.000000) = 75.000000
mul(100.000000, 25.000000) = 2500.000000
div(100.000000, 25.000000) = 4.000000
[juntao@juntao calc]$
```

其他代码如xml的解析，GSList的使用等与插件机制关系不大，感兴趣的朋友可以在附件中查看。

- calc.tar.gz (12.4 KB)
- 下载次数: 182

- 查看图片附件

推荐链接

返回顶楼 - - - - -

- smzd
- 等级: 初级会员
-
- 性别:
- 文章: 61
- 积分: 0
- 来自: 无
- 我现在离线

发表时间：2010-09-02
放眼望去皆动态啊！

锁定老帖子 主题：C语言插件机制(下)
精华帖 (7) :: 良好帖 (0) :: 新手帖 (0) :: 隐藏帖 (0)

作者 正文

返回顶楼 回帖地址
1 0 请登录后投票

- raojl
- 等级: 初级会员
- 
- 性别:
- 文章: 530
- 积分: 90
- 来自: 北京
- 我现在离线

发表时间：2010-09-02
思路挺好的。

返回顶楼 回帖地址
1 0 请登录后投票

- abruzzi
- 等级:
- 
- 性别:
- 文章: 340
- 积分: 1270
- 来自: 西安
- 我现在离线

发表时间：2010-09-02
smzd 写道
放眼望去皆动态啊！

这个主要看如何权衡了，动态的好处是松耦合，大家面向接口写程序，效率较静态的肯定会低一些。这个例子当然完全可以用静态的实现，但是如果应用较复杂，而对效率要求不是很高的话，可以考虑向动态转。不过话又说回来了，现在代码的执行效率，要求貌似已经不那么严格了，看看现在的Java应用有多少，呵呵。

返回顶楼 回帖地址
0 0 请登录后投票

- mxswl
- 等级: 初级会员

发表时间：2010-09-03
和vm比较的话，感觉
差别:静态编译
共同点:主动的动态链接.

锁定老帖子 主题：C语言插件机制(下)
精华帖 (7) :: 良好帖 (0) :: 新手帖 (0) :: 隐藏帖 (0)

作者 正文

- 
- 性别: 💣
- 文章: 97
- 积分: 30
- 来自: 深圳
- 我现在离线

返回顶楼 回帖地址
0 0 请登录后投票

- mallon
- 等级: 初级会员
- 
- 发表时间：2010-09-03
- 但是很多场合下函数原型不可能固定的啊...
- 性别: 💣
- 文章: 50
- 积分: 70
- 来自: 南京
- 我现在离线

返回顶楼 回帖地址
0 0 请登录后投票

- sunday1207
- 等级: 初级会员
- 
- 发表时间：2010-09-04
- 这个列子里动态库的参数只能有两个，是否有必要把参数也配置话呢
- 文章: 20
- 积分: 50
- 来自: ...
- 我现在离线

« 上一页 1 2 下一页 »
论坛首页 → 编程语言技术版

跳转论坛: 编程语言技术

- 首页
- 资讯
- 精华
- 论坛
- 问答
- 博客
- 专栏
- 群组
- 招聘
- 搜索

- 广告服务
- ITeye黑板报
- 联系我们
- 友情链接