编译环境 redhat5.4 64bit

执行complie.sh编译

主要实现动态库，在动态库中实现加法、减法、乘法、除法功能。

当删除某个动态库，不影响其他动态库的功能。

编译脚本：complie.sh

源文件：add.c div.c mul.c sub.c test.c 废弃：source.c

生成文件：libadd.so libdiv.so libmul.so libsub.so test 废弃：libsource.so.1

参考文章

http://www.cnblogs.com/Anker/p/3746802.html

**1、前言**

　　为了使程序方便扩展，具备通用性，可以采用插件形式。采用异步事件驱动模型，保证主程序逻辑不变，将各个业务已动态链接库的形式加载进来，这就是所谓的插件。linux提供了加载和处理动态链接库的系统调用，非常方便。本文先从使用上进行总结，涉及到基本的操作方法，关于动态链接库的本质及如何加载进来，需要进一步学习，后续继续补充。如何将程序设计为插件形式，挖掘出主题和业务之间的关系，需要进一步去学习。

**2、生产动态链接库**

编译参数 **gcc -fPIC -shared**

例如将如下程序编译为动态链接库libcaculate.so，程序如下：

[复制代码](javascript:void(0);)

int add(int a,int b)

{

return (a + b);

}

int sub(int a, int b)

{

return (a - b);

}

int mul(int a, int b)

{

return (a \* b);

}

int div(int a, int b)

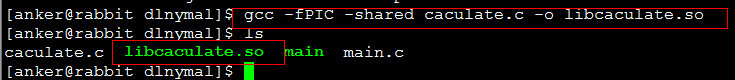
{

return (a / b);

}

[复制代码](javascript:void(0);)

编译如下： **gcc -fPIC -shared caculate.c -o libcaculate.so**



**3、dlopen、dlsym函数介绍**

      在linux上man dlopen可以看到使用说明，函数声明如下：

[复制代码](javascript:void(0);)

#include <dlfcn.h>

void \*dlopen(const char \*filename, int flag);

char \*dlerror(void);

void \*dlsym(void \*handle, const char \*symbol);

int dlclose(void \*handle);

[复制代码](javascript:void(0);)

　　dlopen以指定模式打开指定的动态连接库文件，并返回一个句柄给调用进程，dlerror返回出现的错误，dlsym通过句柄和连接符名称获取函数名或者变量名，dlclose来卸载打开的库。 dlopen打开模式如下：

**RTLD\_LAZY 暂缓决定，等有需要时再解出符号   
　　RTLD\_NOW 立即决定，返回前解除所有未决定的符号。**

采用上面生成的libcaculate.so，写个测试程序如下：

[复制代码](javascript:void(0);)

1 #include <stdio.h>

2 #include <stdlib.h>

3 #include <dlfcn.h>

4

5 //动态链接库路径

6 #define LIB\_CACULATE\_PATH "./libcaculate.so"

7

8 //函数指针

9 typedef int (\*CAC\_FUNC)(int, int);

10

11 int main()

12 {

13 void \*handle;

14 char \*error;

15 CAC\_FUNC cac\_func = NULL;

16

17 //打开动态链接库

18 handle = dlopen(LIB\_CACULATE\_PATH, RTLD\_LAZY);

19 if (!handle) {

20 fprintf(stderr, "%s\n", dlerror());

21 exit(EXIT\_FAILURE);

22 }

23

24 //清除之前存在的错误

25 dlerror();

26

27 //获取一个函数

28 \*(void \*\*) (&cac\_func) = dlsym(handle, "add");

29 if ((error = dlerror()) != NULL) {

30 fprintf(stderr, "%s\n", error);

31 exit(EXIT\_FAILURE);

32 }

33 printf("add: %d\n", (\*cac\_func)(2,7));

34

35 cac\_func = (CAC\_FUNC)dlsym(handle, "sub");

36 printf("sub: %d\n", cac\_func(9,2));

37

38 cac\_func = (CAC\_FUNC)dlsym(handle, "mul");

39 printf("mul: %d\n", cac\_func(3,2));

40

41 cac\_func = (CAC\_FUNC)dlsym(handle, "div");

42 printf("div: %d\n", cac\_func(8,2));

43

44 //关闭动态链接库

45 dlclose(handle);

46 exit(EXIT\_SUCCESS);

47 }

[复制代码](javascript:void(0);)

编译选项如下：**gcc -rdynamic -o main main.c -ldl**

测试结果如下所示：

