cc_binary测试记录

由 刘怀业 创建, 最终由 孙超 修改于 2021-04-12

更新0409

为了使单例模式正常工作，和现有代码兼容，尽力尝试使用 RTLD_LOCAL(dlopen) + -rdynamic=1 方式加载so，在解决了之前gflags报错退出问题之后，现在遇到的问题如下：

1. 加载so后执行so里的某些函数会core，典型如：BusinessData::LoadSkuSizeDict() 将此函数改名，运行正常，但随后core在BusinessData的析构函数里，因此怀疑是searcher和plugin里都有相同的类和相同的成员函数导致符号混乱。
2. so里定义的gflags参数不能正确读取。

虽然 RTLD_LOCAL + -rdynamic=1 方式加载self_contained so使单例模式看起来工作正常，但目前看来带来的不确定不稳定因素很多，不建议使用这种方式。建议参考dlopen官方manpage用DEEPBIND加载self_contained so，用地址传递单例实例。

DEEPBIND加载plugin，正确获取plugin里独立定义的gflags参数方法如下：

1. searcher端，在调用gflags解析参数之前将main函数参数复制一份（不能仅仅复制argv指针，应该做深度拷贝），参考 searcher/server/args.h
2. main_site端，IPlugin接口及其实现提供一个ParseArgs(int argc, char **argv)接口，在searcher加载plugin so成功后调用此接口，在其中调用gflags解析参数，相当于main参数被解析2次，分别在searcher和main_site中，参考 main_site: plugin/plugin.h Plugin:ParseArgs()

经测试，这种方法可以正确读取在plugin so里独立定义的gflags参数。（无需像上文那样修改gflags源代码，因为这是DEEPBIND）

单例模式仍然采用searcher获取单例，告知plugin地址的方式，用nestdict ServableManager测试可行

相关测试代码链接见文末，已更新。

更新0408

经查，“xx is being linked both statically and dynamically into this executable” 是gflags库内部报错，代码位于 gflags.cc FlagRegistry::RegisterFlag() 函数

```
ReportError(DIE, "ERROR: something wrong with flag '%s' in file '%s'. "
"One possibility: file '%s' is being linked both statically "
"and dynamically into this executable.\n",
flag->name(),
flag->filename(), flag→filename());
```

将第一个参数 DIE 改成 DO_NOT_DIE 可解决加载报错退出问题。

测试结果 RTLD_LOCAL + -rdynamic=1 小程序测试正常，但searcher+libapp.so 启动报错

```
E20210408 19:12:45.178655 52205 searcher.cc:249] MemInfo after start: pid=52205, virt_mem=174.76M, phy_mem=127.19M
2021-04-08 19:12:45.212601: E searcher/server/main.cc:120] Register to JSF, iface:com.jd.search.Searcher, alias:t0p_ht_0
2021-04-08 19:12:45.220248: E searcher/server/main.cc:123] Register to JSF, iface:com.jd.search.Searcher, alias:t0p_ht_0_debug
[2021-04-08 19:12:45][external/jcommon/common/common.h:88] umpInit completed (result:0)
2021-04-08 19:12:45.222281: E searcher/server/main.cc:141] [searcher server started]
E20210408 19:12:45.512120 58221 feature_operator.h:76] Fail to init fop of features like 25, status:-2
[external/jcommon/common/common.h:99] signo: 11
E20210408 19:12:45.538501 58218 feature_operator.h:76] Fail to init fop of features like 524400, status:-2
```

测试结果

cc_binary编出self contained so，用现在的searcher加载时dlopen失败报错：
ERROR: something wrong with flag 'logtostderr' in file 'external/com_github_google_glog/src/logging.cc'. One possibility: file 'external/com_github_google_glog/src/logging.cc' is being linked both statically and dynamically into this executable.

经查，logtostderr在glog源码中也是使用gflags定义格式：

GLOG_DEFINE_bool(logtostderr, BoolFromEnv("GOOGLE_LOGTOSTDERR", false),
"log messages go to stderr instead of logfiles");

所以这种加载错误目前来看是由gflags库引起。

当前dlopen的参数选项是：RTLD_LOCAL，改成 RTLD_DEEPBIND 后加载正常，但plugin so里定义的gflags参数无法正确读入，第三方库中的singleton实例如nestdict::ServableManager 在searcher和main_site中是两个独立的实例（地址不一样），并非预期的全局唯一实例。

经开发的独立测试程序实验，-rdynamic 链接选项对so是否成功加载及单例模式能否正常工作也有决定作用，测试结果如下：

	-rdynamic=0	**-rdynamic=1**
RTLD_LOCAL	正常加载，但singleton不能正常工作(地址不同)	正常加载，singleton工作正常（两边地址相同）；但两边不能同时有glog gflags，否则dlopen出错：linked both statically and dynamically into this executable.
RTLD_DEEPBIND	无论-rdynamic是否指定，均能正常加载so，但singleton不能正常工作（两边地址不同）	

对于singleton模式不能正常工作问题，经测试，可以由主程序获取全局唯一实例，dlopen成功加载plugin so后，将实例地址告知plugin，plugin可正常使用，两边修改相互可见。

相关代码链接

searcher: http://gerrit.jd.com/#/c/182103/

main_site: http://gerrit.jd.com/#/c/183225/

无标签