Динамические библиотеки и плагины

Дмитрий Свиридкин
СПбГУ
24.02.2021

Сборка динамической библиотеки

gcc -o libимя_библиотеки.so.версия -shared -fPIC список_объектников_и_проч

Линковка с динамической библиотекой (статическое связывание с библиотекой)

gcc опции_сборки_и_объектники -L путь_где_искать_.so -lимя_библиотеки

В перечисленных при линковке библиотеках будут искаться неопределенные (U) в объектниках символы

При сборке исполнимого файла все символы должны быть определены
При сборке библиотеки — можно оставлять неопределенными.

main.c

libs_undef > C main.c > main(int, char * [])

```c
extern void hello(const char*);

int main(int argc, char* argv[]) {
    hello("world");
    return 0;
}
```

hello_private.c

libs_undef > C hello_private.c > hello_private(const char *)

```c
#include <stdio.h>

void hello_private(const char* name) {
    printf("Hello, %s", name);
}
```

hello_public.c

libs_undef > C hello_public.c > hello(const char *)

```c
extern void hello_private(const char*);


void hello(const char* name) {
    hello_private(name);
}
```

```
dmis@dmis-MS-7A15:~/LinuxEgs/libs_undef$ gcc -shared -fPIC -o libhprivate.so hello_private.c
dmis@dmis-MS-7A15:~/LinuxEgs/libs_undef$ gcc -shared -fPIC -o libhpublic.so hello_public.c
dmis@dmis-MS-7A15:~/LinuxEgs/libs_undef$ gcc -o hello main.c -L $(pwd) -lhpublic
/usr/bin/ld: /home/dmis/LinuxEgs/libs_undef/libhpublic.so: undefined reference to `hello_private'
```

Заставляем работать

```
is-MS-7A15:~/LinuxEgs/libs_undef$ gcc -shared -fPIC -o libhprivate.so hello_private.c
is-MS-7A15:~/LinuxEgs/libs_undef$ gcc -shared -fPIC -o libhpublic.so hello_public.c
is-MS-7A15:~/LinuxEgs/libs_undef$ gcc -o hello main.c -L $(pwd) -lhpublic -lhprivate
is-MS-7A15:~/LinuxEgs/libs_undef$ ldd hello
 linux-vdso.so.1 (0x00007ffd743b0000)
 libhpublic.so => not found
 libhprivate.so => not found
 libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f3ecb883000)
 /lib64/ld-linux-x86-64.so.2 (0x00007f3ecbaa9000)
is-MS-7A15:~/LinuxEgs/libs_undef$ ldd libhpublic.so
 statically linked
```

```
is-MS-7A15:~/LinuxEgs/libs_undef$ gcc -shared -fPIC -o libhpublic.so hello_public.c -L $(pwd) -lhprivate
is-MS-7A15:~/LinuxEgs/libs_undef$ gcc -o hello main.c -L $(pwd) -lhpublic
n/ld: warning: libhprivate.so, needed by /home/dmis/LinuxEgs/libs_undef/libhpublic.so, not found (try using -rpath or -rpath-link)
n/ld: /home/dmis/LinuxEgs/libs_undef/libhpublic.so: undefined reference to `hello_private'
2: error: ld returned 1 exit status
is-MS-7A15:~/LinuxEgs/libs_undef$ gcc -o hello main.c -L $(pwd) -lhpublic -lhprivate
is-MS-7A15:~/LinuxEgs/libs_undef$ ldd hello
 linux-vdso.so.1 (0x00007ffd6bf11000)
 libhpublic.so => not found
 libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007ff5bb750000)
 /lib64/ld-linux-x86-64.so.2 (0x00007ff5bb976000)
is-MS-7A15:~/LinuxEgs/libs_undef$ ldd libhpublic.so
 linux-vdso.so.1 (0x00007ffd77b84000)
 libhprivate.so => not found
```

```
mis-MS-7A15:~/LinuxEgs/libs_undef$ gcc -shared -fPIC -o libhpublic.so hello_public.c -L $(pwd) -lhprivate
mis-MS-7A15:~/LinuxEgs/libs_undef$ gcc -o hello main.c -L $(pwd) -Wl,-rpath-link,$(pwd) -lhpublic
mis-MS-7A15:~/LinuxEgs/libs_undef$ ldd hello
  linux-vdso.so.1 (0x00007ffc643db000)
  libhpublic.so => not found
  libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fec1740c000)
```

Не пишите сложные системы сборки вручную!

CMake подтягивает транзитивные зависимости

s_undef > M CMakeLists.txt

```
1   project(HelloTransitive)
2
3   set(CMAKE_POSITION_INDEPENDENT_CODE True)
4
5   add_library(HelloPrivate SHARED hello_private.c)
6
7   add_library(HelloPublic SHARED hello_public.c)
8   target_link_libraries(HelloPublic PRIVATE HelloPrivate)
9
10  add_executable(Hello main.c)
11  target_link_libraries(Hello PRIVATE HelloPublic)
```

```
dmis@dmis-MS-7A15:~/LinuxEgs/libs_undef$ mkdir build
dmis@dmis-MS-7A15:~/LinuxEgs/libs_undef$ cd build/
dmis@dmis-MS-7A15:~/LinuxEgs/libs_undef/build$ cmake ..
-- The C compiler identification is GNU 10.2.0
-- The CXX compiler identification is GNU 10.2.0
dmis@dmis-MS-7A15:~/LinuxEgs/libs_undef/build$ cmake --build .
Scanning dependencies of target HelloPrivate
[ 16%] Building C object CMakeFiles/HelloPrivate.dir/hello_private.c.o
[ 33%] Linking C shared library libHelloPrivate.so
[ 33%] Built target HelloPrivate
Scanning dependencies of target HelloPublic
[ 50%] Building C object CMakeFiles/HelloPublic.dir/hello_public.c.o
[ 66%] Linking C shared library libHelloPublic.so
[ 66%] Built target HelloPublic
Scanning dependencies of target Hello
[ 83%] Building C object CMakeFiles/Hello.dir/main.c.o
[100%] Linking C executable Hello
[100%] Built target Hello
```

Еще немного про видимость символов

strip для выкидывания лишних символов

```
dmis@dmis-MS-7A15:~/LinuxEgs/libs_undef$ nm libhpublic.so
0000000000004028 b completed.0
                 w __cxa_finalize
0000000000001060 t deregister_tm_clones
00000000000010d0 t __do_global_dtors_aux
0000000000003e48 d __do_global_dtors_aux_fini_array_entry
0000000000004020 d __dso_handle
0000000000003e50 d _DYNAMIC
0000000000001138 t _fini
0000000000001110 t frame_dummy
0000000000003e40 d __frame_dummy_init_array_entry
00000000000020c0 r __FRAME_END__
0000000000004000 d _GLOBAL_OFFSET_TABLE_
                 w __gmon_start__
0000000000002000 r __GNU_EH_FRAME_HDR
0000000000001119 T hello
                 U hello_private
0000000000001000 t _init
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
0000000000001090 t register_tm_clones
0000000000004028 d __TMC_END__
dmis@dmis-MS-7A15:~/LinuxEgs/libs_undef$ strip libhpublic.so
dmis@dmis-MS-7A15:~/LinuxEgs/libs_undef$ nm libhpublic.so
nm: libhpublic.so: no symbols
```

- На работу динамической библиотеки не повлияет (динамические символы не трогает)
- Статическую — может поломать
- Может повлиять на печать stacktrace в отладчике (адреса вместо имен)
- Уменьшает размер бинаря

```
dmis@dmis-MS-7A15:~/LinuxEgs/libs_undef$ nm -D -C libhpublic.so
                 w __cxa_finalize
                 w __gmon_start__
0000000000001119 T hello
                 U hello_private
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
```

```cpp
#include <cmath>

float private_function(float f)
{
    return std::abs(f);
}

float __attribute__ ((visibility ("default"))) public_function_from_static_lib() {
    return 42;
}

extern "C" float __attribute__ ((visibility ("default"))) public_function(float f)
{
    return private_function(f);
}
```

```
dmis@dmis-MS-7A15:~/LinuxEgs/link_script$ g++ -shared test.cpp -o libtest.so -fvisibility=hidden
dmis@dmis-MS-7A15:~/LinuxEgs/link_script$ nm -DC libtest.so
                 w __cxa_finalize
                 w __gmon_start__
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
000000000000114a T public_function
0000000000001138 T public_function_from_static_lib()
0000000000001169 W std::abs(float)
```

```
dmis@dmis-MS-7A15:~/LinuxEgs/link_script$ g++ -shared test.cpp -o libtest.so -fvisibility=hidden -fvisibility-inlines-hidden
dmis@dmis-MS-7A15:~/LinuxEgs/link_script$ nm -DC libtest.so
                 w __cxa_finalize
                 w __gmon_start__
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
000000000000112a T public_function
0000000000001118 T public_function_from_static_lib()
```

link_script > libtest.map

```
1  LIBTEST {
2      global:
3          public_function;
4
5      local:
6          *;
7  };
```

```
dmis@dmis-MS-7A15:~/LinuxEgs/link_script$ g++ -shared test.cpp -o libtest.so -fvisibility=hidden -fvisibility-inlines-hidden \
> -Wl,--version-script=libtest.map
dmis@dmis-MS-7A15:~/LinuxEgs/link_script$ nm -DC libtest.so
                 w __cxa_finalize
                 w __gmon_start__
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
0000000000000000 A LIBTEST
000000000000112a T public_function
```

libdl и плагины

iface.h

plugins_egs > iface.h > ...

```c
#ifdef __cplusplus
extern "C" {
#endif

int plugin_eval(int a, int b);

#ifdef __cplusplus
}
#endif
```

sum.c

plugins_egs > sum.c > plugin_eval(int, int)

```c
#include "iface.h"

int plugin_eval(int a, int b) {
    return a + b;
}
```

mul.c

plugins_egs > mul.c > plugin_eval(int, int)

```c
#include "iface.h"

int plugin_eval(int a, int b) {
    return a * b;
}
```

main.cpp

plugins_egs > main.cpp > main(int, char * [])

```cpp
typedef int (*eval_func)(int, int);

int main(int argc, char* argv[]){
    if (argc <= 1) {
        std::cout << "provide path to plugin";
        return EXIT_FAILURE;
    }

    int a = 0, b = 0;
    std::cin >> a >> b;

    std::unique_ptr<void, decltype([](void* plugin){
        dlclose(plugin);
    })> plugin { dlopen(argv[1], RTLD_LAZY | RTLD_LOCAL) };

    if (!plugin) {
        std::cout << dlerror() << "\n";
        return EXIT_FAILURE;
    }


    auto eval = reinterpret_cast<eval_func>(dlsym(plugin.get(), "plugin_eval"));

    if (!eval) {
        std::cout << dlerror() << "\n";
        return EXIT_FAILURE;
    }

    std::cout << eval(a, b) << "\n";
}
```

```
dmis@dmis-MS-7A15:~/LinuxEgs/plugins_egs$ gcc -o libmul.so -shared -fPIC mul.c
dmis@dmis-MS-7A15:~/LinuxEgs/plugins_egs$ gcc -o libsum.so -shared -fPIC sum.c
dmis@dmis-MS-7A15:~/LinuxEgs/plugins_egs$ g++ -std=c++20 -o test_plugin main.cpp -ldl
dmis@dmis-MS-7A15:~/LinuxEgs/plugins_egs$ ./test_plugin libmul.so
1 5
libmul.so: cannot open shared object file: No such file or directory
dmis@dmis-MS-7A15:~/LinuxEgs/plugins_egs$ ./test_plugin ./libmul.so
1 4
4
dmis@dmis-MS-7A15:~/LinuxEgs/plugins_egs$ ./test_plugin ./libsum.so
1 4
5
```

Если передать путь. dlopen будет искать по этому пути
Иначе будет искать по тем путям, что указаны в LD_LIBRARY_PATH или по умолчанию

У dlopen есть вариация dlmopen, позволяющая грузить в разные пространства имен. Можно грузить плагины, зависящие от разных, конфликтующих версий библиотек.

Сам себе плагин

```cpp
#include "iface.h"

extern "C" {
int plugin_eval(int a, int b){
    return a * a + b * b;
}

}

using eval_func = decltype(&plugin_eval);

int main(int argc, char* argv[]){
    int a = 0, b = 0;
    std::cin >> a >> b;

    const char* filename = [argc, argv]()->const char*{
        if (argc <= 1) {
            return nullptr;
        }
        return argv[1];
    }();

    std::unique_ptr<void, decltype([](void* plugin){
        dlclose(plugin);
    })> plugin { dlopen(filename, RTLD_LAZY | RTLD_LOCAL) };

    if (!plugin) {
        std::cout << dlerror() << "\n";
        return EXIT_FAILURE;
    }


    auto eval = reinterpret_cast<eval_func>(dlsym(plugin.get(), "plugin_eval"));

    if (!eval) {
        std::cout << dlerror() << "\n";
        return EXIT_FAILURE;
    }

    std::cout << eval(a, b) << "\n";
}
```

```
dmis@dmis-MS-7A15:~/LinuxEgs/plugins_egs$ g++ -std=c++20 -rdynamic -o test_plugin main.cpp -ldl
dmis@dmis-MS-7A15:~/LinuxEgs/plugins_egs$ ./test_plugin
1 4
17
dmis@dmis-MS-7A15:~/LinuxEgs/plugins_egs$ ./test_plugin ./libsum.so
1 4
5
dmis@dmis-MS-7A15:~/LinuxEgs/plugins_egs$ ./test_plugin ./test_plugin
1 4
./test_plugin: cannot dynamically load position-independent executable
```

В новых версиях glibc фичу с загрузкой exe поломали. Но можно починить.

Initialization order fiasco

```cpp
// sourceSIOF1.cpp

int quad(int n) {
    return n * n;
}

auto staticA  = quad(5);
```

```cpp
// mainSOIF1.cpp

#include <iostream>

extern int staticA;   // (1)
auto staticB = staticA;

int main() {

    std::cout << std::endl;

    std::cout << "staticB: " << staticB << std::endl;

    std::cout << std::endl;

}
```

Код работает в зависимости от порядка обработки единиц трансляции

вариант решения

```cpp
// sourceSIOF2.cpp

int quad(int n) {
    return n * n;
}

int& staticA() {

    static auto staticA  = quad(5);   // (1)
    return staticA;

}
```

```cpp
// mainSOIF2.cpp

#include <iostream>

int& staticA();           // (2)

auto staticB = staticA(); // (3)

int main() {
```

Что если библиотеке нужна инициализация?

libA.c

ctors > libA.c > compute_something(int)

```c

extern int get_square(int x);

int compute_something(int x) {
    return x + get_square(5);
}

```

main.cpp

ctors > main.cpp > main()

```cpp
#include <iostream>


extern "C" int get_square(int x);
extern "C" void initialize_library(void);
extern "C" int compute_something(int x);

int main() {
    std::cout << "square 5: " << get_square(5) << "\n"; // 25 ?
    std::cout << "compute: " << compute_something(5) << "\n"; // 30 ?
    initialize_library();
    std::cout << "square 5:" << get_square(5) << "\n";
    std::cout << "compute: " << compute_something(5) << "\n"; // 30 ?
}
```

libB.c

ctors > libB.c > get_square(int)

```c

#define SQUARES_CNT 25
static int squares_LUT[SQUARES_CNT];

void initialize_library(void) {
    int add = 1;
    int cur_sqr = 0;
    for (int i = 0; i < SQUARES_CNT; ++i) {
        squares_LUT[i] = cur_sqr;
        cur_sqr += add;
        add += 2;
    }
}

int get_square(int x) {
    if (x < 0 ) {
        x = -x;
    }
    if (x < SQUARES_CNT) {
        return squares_LUT[x];
    } else {
        return x * x;
    }
}
```

Что если нужна инициализация/деинициализация?

Проверка в каждой функции —> замедление

Функция инициализации наружу —> неправильное использование

Решение: конструкторы

main.cpp

ctors > main.cpp > ...

```cpp
#include <iostream>


extern "C" int get_square(int x);
//extern "C" void initialize_library(void);
extern "C" int compute_something(int x);

int main() {
    std::cout << "square 5: " << get_square(5) << "\n"; // 25 ?
    std::cout << "compute: " << compute_something(5) << "\n"; // 30 ?
    // initialize_library();
    std::cout << "square 5:" << get_square(5) << "\n";
    std::cout << "compute: " << compute_something(5) << "\n"; // 30 ?
}
```

libB.c

ctors > libB.c > initialize_library(void)

```c

#define SQUARES_CNT 25
static int squares_LUT[SQUARES_CNT];

static void __attribute__ ((constructor)) initialize_library(void) {
    int add = 1;
    int cur_sqr = 0;
    for (int i = 0; i < SQUARES_CNT; ++i) {
        squares_LUT[i] = cur_sqr;
        cur_sqr += add;
        add += 2;
    }
}

int get_square(int x) {
```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE 1: bash

```
dmis@dmis-MS-7A15:~/LinuxEgs/ctors$ gcc -shared -fPIC -o libB.so libB.c
dmis@dmis-MS-7A15:~/LinuxEgs/ctors$ LD_LIBRARY_PATH=$(pwd) ./exe
square 5: 25
compute: 30
square 5:25
compute: 30
```

Конструкторы/деструкторы библиотеки

```
/*  functions constructX, destructX use attributes 'constructor' and
    'destructor' to create prioritized entries in the .ctors, .dtors
    ELF sections, respectively.

    NOTE: priorities 0-100 are reserved
*/
void construct1 () __attribute__ ((constructor (101)));
void construct2 () __attribute__ ((constructor (102)));
void destruct1 () __attribute__ ((destructor (101)));
void destruct2 () __attribute__ ((destructor (102)));
```

```
#define SECTION( S ) __attribute__ ((section ( S )))

void test (void) {
printf("\n\ttest() utilizing -- (.section .ctors/.dtors) w/o priority\n");
}

void (*funcptr1)(void) SECTION(".ctors") =test;
void (*funcptr2)(void) SECTION(".ctors") =test;
void (*funcptr3)(void) SECTION(".dtors") =test;
```

Конструкторы/деструкторы C++

main.cpp

ctors > main.cpp > main()

```cpp
#include <iostream>


extern "C" int get_square(int x);
//extern "C" void initialize_library(void);
extern "C" int compute_something(int x);

int main() {
    std::cout << "square 5: " << get_square(5) <
    std::cout << "compute: " << compute_somethin
    // initialize_library();
    std::cout << "square 5:" << get_square(5) <<
    std::cout << "compute: " << compute_somethin
}
```

main.cpp | libB.c | libB.cpp

ctors > libB.cpp > get_square(int)

```cpp
#define SQUARES_CNT 25
static int squares_LUT[SQUARES_CNT];

namespace {
struct Init {
    Init() {
        int add = 1;
        int cur_sqr = 0;
        for (int i = 0; i < SQUARES_CNT; ++i) {
            squares_LUT[i] = cur_sqr;
            cur_sqr += add;
            add += 2;
        }
    }
} init {};
}

extern "C" int get_square(int x) {
```

```cpp
#include <iostream>
#include <stdio.h>

static void __attribute__ ((constructor)) LibraryCtor (void) {
    printf("Library Ctor\n");
    //std::cout << "Library Ctor\n";
}
static void __attribute__ ((destructor)) LibraryDctor (void) {
    printf("Library Dctor\n");
    //std::cout << "Library Dctor\n";
}

namespace {
    struct StatObj {
        StatObj() {
            std::cout << "StatObj Ctor\n";
        }
        ~StatObj() {
            std::cout << "StatObj Dctor\n";
        }
    } init {};
}

int main() {
    std::exit(0);
}
```

```
dmis@dmis-MS-7A15:~/Linux
Library Ctor
StatObj Ctor
StatObj Dctor
Library Dctor
```