作者：代兴　邮箱：503268771@qq.com　Vx 公众号：东方架构师

# LDS 脚本的含义

LDS 脚本的核心功能是绑定 symbol 到 section，并且设置 symbol 的内存地址。
LDS 脚本功能很多，这里说明主要功能。

LDS 脚本的简化格式为:

```
SECTIONS
{
    . = 0x3000003  ;          /*  重置当前地址  */
    . = ALIGN(8) ;            /*  对齐当前地址  */
    section_mark_name :       /*  设置 section  */
    {
        tmp_data_begin = . ; /*  设置符号的地址  */
        *(mark_name);         /*  收集 mark_name 标记 */
    }
}
```

`SECTIONS` 表示设置自定义的 section。
`. = 0x3000003` 表示重置当前地址为 0x3000003。数值 0x3000003 为示例。
`. = ALIGN(8)` 把当前地址按照 8 字节对齐。数值 8 为示例。
`section_mark_name` 表示开始设置一个 section。名称 section_mark_name 为示例。
`tmp_data_begin = .` 表示把当前地址赋给 tmp_data_begin。名称 tmp_data_begin 为示例。
`*(mark_name)` 表示在全部文件中收集 mark_name 标记的符号。名称 mark_name 为示例，左侧的*表示全部文件。
源码使用 `__attribute__((section("mark_name")))` 给符号打上 section 标记。

# 用 C 程序分析 LDS 脚本

编写代码： main.c

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

// 普通的变量。在 .data
int32_t param_tree = 0xEEEEEEEE;

// 普通的方法。在 .text
void func_print(char *desc, void *addr)
{
    printf(" %16s    addr = %10p \n", desc, addr);
}
```

```
// ----------------------------------------

// 自定义 section
__attribute__((section("eee_data")))
int32_t eee_color = 0xA1A2A3A4;

// 自定义 section
__attribute__((section("eee_data")))
int64_t eee_height = 0x7172737475767778;

// 自定义 section
__attribute__((section("eee_data_2")))
int32_t eee_speed = 0xB1B2B3B4;

// 自定义 section
__attribute__((section("www_data")))
int32_t www_fish = 0x91929394;

// 占位的符号。标记内存地址。
float eee_data_begin;
char eee_data_inner;
short eee_data_end;

// 自定义 section
__attribute__((section("ddd_func"))) int ddd_func_dance(int time)
{
    return time + 300;
}

int main()
{
    // 普通的变量。在 .data
    printf("\n Param in default section : \n");
    func_print("param_tree", &param_tree);

    // 自定义 section 的变量
    printf("\n Param in custom section : \n");
    func_print("eee_color", &eee_color);
    func_print("eee_height", &eee_height);
    func_print("eee_speed", &eee_speed);
    func_print("www_fish", &www_fish);

    // 占位的符号。标记内存地址。
    printf("\n Place symbol : \n");
    func_print("eee_data_begin", &eee_data_begin);
    func_print("eee_data_inner", &eee_data_inner);
    func_print("eee_data_end", &eee_data_end);

    // 普通的方法。在 .text
```

```
    printf("\n Func in default section : \n");
    func_print("func_print", func_print);

    // 自定义 section
    printf("\n Func in custom section : \n");
    func_print("ddd_func_dance", ddd_func_dance);

    return 0;
}
```

编写脚本： main.lds

```
SECTIONS
{
    . = 0x3000003 ;          /*  重置当前地址  */

    section_ddd_func :       /*  设置 section  */
    {
        *(ddd_func);         /*  收集 ddd_func 标记 */
    }

    . = 0x7000006 ;          /*  重置当前地址  */
    . = ALIGN(8) ;           /*  对齐当前地址  */

    section_eee_data :       /*  设置 section  */
    {
        eee_data_begin = . ;     /*  设置符号的地址  */
        *(eee_data);             /*  收集 eee_data 标记 */
        eee_data_inner = . ;     /*  设置符号的地址  */
        *(eee_data_2);           /*  收集 eee_data_2 标记 */
        eee_data_end = . ;       /*  设置符号的地址  */
    }

    section_www_data :       /*  设置 section  */
    {
        *(www_data);         /*  收集 www_data 标记 */
    }
}
```

编译代码：

```
gcc  main.c main.lds  -o main

readelf -a main > main.elf.txt
objdump -D main > main.dump.txt
```

运行代码：

```
[root@local lds]# ./main

 Param in default section :
```

```
      param_tree    addr =   0x60103c

Param in custom section :
       eee_color    addr =  0x7000008
      eee_height    addr =  0x7000010
       eee_speed    addr =  0x7000018
        www_fish    addr =  0x700001c

Place symbol :
  eee_data_begin    addr =  0x7000008
  eee_data_inner    addr =  0x7000018
    eee_data_end    addr =  0x700001c

Func in default section :
      func_print    addr =   0x4005ed

Func in custom section :
  ddd_func_dance    addr =  0x3000003
```

查看符号和地址：

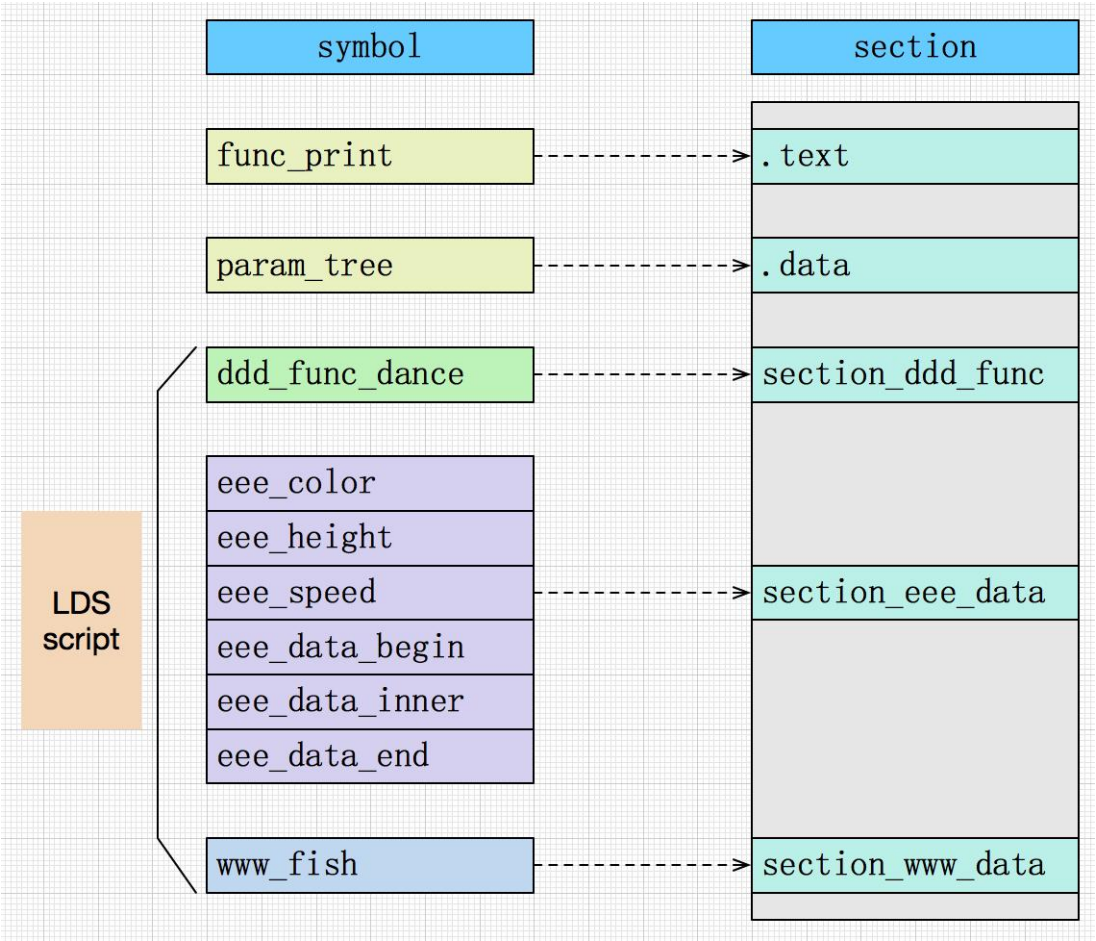
查看文件 main.elf.txt ，找到相关的符号。

```
Symbol table '.symtab' contains 77 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
    50: 00000000004005ed    44 FUNC    GLOBAL DEFAULT   13 func_print
    51: 0000000007000018     1 OBJECT  GLOBAL DEFAULT   28 eee_data_inner
    55: 000000000700001c     2 OBJECT  GLOBAL DEFAULT   28 eee_data_end
    56: 0000000007000018     4 OBJECT  GLOBAL DEFAULT   28 eee_speed
    59: 0000000007000008     4 OBJECT  GLOBAL DEFAULT   28 eee_color
    65: 0000000007000010     8 OBJECT  GLOBAL DEFAULT   28 eee_height
    71: 000000000700001c     4 OBJECT  GLOBAL DEFAULT   29 www_fish
    72: 000000000060103c     4 OBJECT  GLOBAL DEFAULT   24 param_tree
    74: 0000000003000003    17 FUNC    GLOBAL DEFAULT   27 ddd_func_dance
    76: 0000000007000008     4 OBJECT  GLOBAL DEFAULT   28 eee_data_begin
```

查看文件 main.dump.txt ，找到相关的符号。

| 段 section | 符号 |
| --- | --- |
| 默认的代码段 | Disassembly of section .text:<br>00000000004005ed <func_print>:<br>0000000000400619 <main>: |
| 默认的数据段 | Disassembly of section .data:<br>000000000060103c <param_tree>: |
| 自定义的代码段 | Disassembly of section section_ddd_func:<br>0000000003000003 <ddd_func_dance>: |
| 自定义的数据段 | Disassembly of section section_eee_data:<br>0000000007000008 <eee_color>:<br>0000000007000010 <eee_height>:<br>0000000007000018 <eee_data_inner>: |
| 自定义的数据段 | Disassembly of section section_www_data:<br>000000000700001c <www_fish>: |

symbol 和 section 的映射图：
源码中，symbol 分散在各处，使用 LDS 脚本，自定义 section 包含哪些 symbol。




比较符号的地址：
变量 param_tree 的地址为 0x60103c，在 section .data。
变量 eee_color 的地址为 0x7000008，在 section_eee_data。
变量 eee_speed 的地址为 0x7000018，在 section_eee_data。
LDS 脚本，使用 `*(eee_data)` 、`*(eee_data_2)` ，把 eee_data、eee_data_2 标记的符号，都放入 section_eee_data ，在一块内存区。

比较 section 的起始地址：
LDS 脚本，设置 section_eee_data 之前，使用 `. = 0x7000006` 重置当前地址，又使用 `. = ALIGN(8)` 把当前地址按照 8 字节对齐，所以当前地址为 0x7000008。之后，设置 section_eee_data，所以 section_eee_data 的起始地址为 0x7000008。eee_color 是 section_eee_data 的第一个符号，所以 eee_color 的地址为 0x7000008。
LDS 脚本，设置 section_ddd_func 之前，使用 `. = 0x3000003` 重置当前地址，并且没有做内存对齐。ddd_func_dance 是 section_ddd_func 的第一个符号，所以 ddd_func_dance 的地址为 0x3000003。

比较占位符的地址：

符号表中，eee_color 和 eee_data_begin 的地址都为 0000000007000008。
LDS 脚本，设置 section_eee_data 时，首先使用 `eee_data_begin = .` 把当前地址赋给 eee_data_begin，然后收集 `*(eee_data)` 对应的符号。
eee_color 是 `*(eee_data)` 标记的第一个符号，所以把当前地址赋给 eee_color，进而导致 eee_color 和 eee_data_begin 的地址相同。

问题：文件 main.dump.txt 中为什么没有找到符号 eee_speed?
LDS 脚本，先使用 `eee_data_inner = .` 把当前地址赋给 eee_data_inner，然后收集 `*(eee_data_2)` 对应的符号。
eee_speed 使用 eee_data_2 标记，并且是 eee_data_2 标记的第一个符号。所以，eee_data_inner 和 eee_speed 的地址都为 0000000007000018。
文件 main.dump.txt，根据地址反向查找符号，首先找到 eee_data_inner，所以显示为 `0000000007000018 <eee_data_inner>:` 。