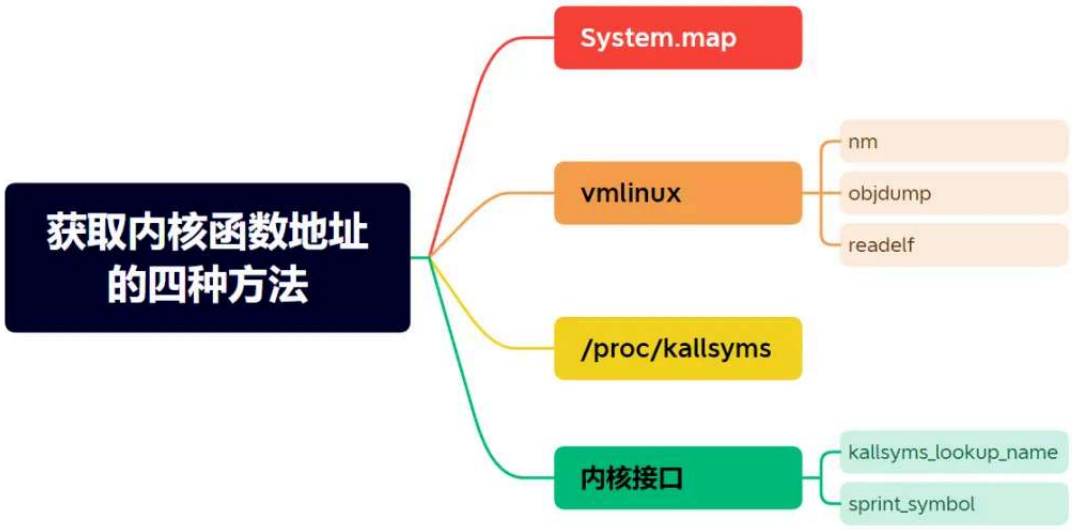
Linux内核调试篇——获取内核函数地址的四种方法

原创 Vincenterr 嵌入式Linux充电站 2023年11月16日 08:15 广东



在内核调试中，经常需要知道某个函数的地址，或者根据函数地址找到对应的函数，从而进行更深一步的debug。

下面介绍四种获取内核函数地址的方法：

1、System.map

在编译Linux内核时，会产生一个内核映像文件 `System.map` ，也叫内核符号表。

内核符号表是一个映射，它将内核代码段中的地址映射到对应的函数名或全局变量名。

在 `System.map` 文件中，每一行都包含一个内核符号，每个符号包含三部分：

- 地址：符号在内核内存中的地址。
- 类型：符号的类型。例如，"T"表示该符号是一个在代码段中的函数。
- 名称：符号的名字，可以是函数名或者变量名。

例如，我们要查找名为“ `do_fork` ”的函数的地址，可以使用以下命令：

```
grep ' do_fork' System.map
```

然后会得到类似于这样的输出：

```
c0105020 T do_fork
```

这代表， `c0105020` 就是函数 `do_fork` 的地址， `T` 代表该符号是个函数。

或者，直接打开 `System.map` 搜索对应函数名或者地址。 `System.map` 内容类似如下：

```
c0004000 A swapper_pg_dir
c0008000 T __init_begin
c0008000 T _sinittext
c0008000 T _stext
c0008000 T stext
c0008034 t __enable_mmu
c0008060 t __turn_mmu_on
......
```

A、T、t等都代表不同的类型，具体类型的定义可参考：

类型	说明
A	该符号的值是不能改变的，等于const
B	该符号来自于未初始化代码段bss段
C	该符号是通用的，通用的符号指未初始化的数据。当链接时，多个通用符号可能对应一个名称，如果该符号在某一个位置定义，这个通用符号被当做未定义的引用。不明白，内核中也没有该类型的符号
D	该符号位于初始化的数据段
G	位于初始化数据段，专门对应小的数据对象，比如global int x,对应的大数据对象为 数组类型等
I	到其他符号的间接引用，是对于a.out文件的GNU扩展，使用非常少
N	调试符号
R	只读代码段的符号
S	BSS段（未初始化数据段）的小对象符号
T	代码段符号，全局函数，t为局部函数
U	未定义的符号
V	该符号是一个weak object，当其连接到为定义的对象上上，该符号的值变为0
W	类似于V
-	该符号是a.out文件中的一个stabs symbol，获取调试信息
?	未知类型的符号

2、vmlinux

vmlinux 也是 Linux 编译出来的内核映像文件，可以通过 nm 、 objdump 和 readelf 等工具来查看它的符号表，从而获取函数地址。

2.1 nm读取vmlinux

nm 命令是用于查看二进制文件（如可执行文件、目标文件、库）的符号表的工具，所以可以用 nm 命令来读取 vmlinux 里的符号表。

- nm 查找 vmlinux 中函数名为 do_fork 的地址：

```
nm vmlinux | grep "do_fork"
```

- nm 查找 vmlinux 中地址为 c0105020 的符号：

```
nm vmlinux | grep c0105020
```

nm 命令的输出，与 System.map 类似，会的得到类似于以下的输出：

```
c0105020 T do_fork
```

2.2 objdump反汇编vmlinux

objdump 工具用于反汇编，可以将 vmlinux 反汇编出来，得到的反汇编文件就会包含函数名和对应的地址，可以直接查看文本内容。

- 使用**objdump**查看**vmlinux**函数地址：

```
objdump -d vmlinux | grep "函数名"
```

这会返回与指定函数名匹配的符号的地址、类型和名称。 objdump 的 -d 选项表示反汇编代码。

也可以将vmlinux的内容全部反汇编出来，重定向到一个文件，然后直接查看文本内容：

```
objdump -D vmlinux > vmlinux_dump.txt
```

上述命令会将 vmlinux 所有内容反汇编，并将结果输出到 vmlinux_dump.txt 文件中， -D 表示反汇编所有代码段。

2.3 readelf读取vmlinux

使用readelf也可以读取vmlinux的符号表，命令如下：

```
readelf -s vmlinux
```

由于vmlinux比较大，如果要查找某个函数的地址，需要使用grep进行过滤。

例如，找到 do_fork 函数的地址，你可以使用以下命令：

```
readelf -s vmlinux | grep "do_fork"
```

然后，你就会看到类似这样的输出：

```
56481: c10601e0    96 FUNC    GLOBAL DEFAULT    1 do_fork
```

输出内容表示， do_fork 函数的地址是 c10601e0 。

3、/proc/kallsyms

vmlinux 是编译时生成的，假设你没有 vmlinux 这个文件，只有正在运行的 Linux 系统，此时也可以通过 /proc/kallsyms 获取函数地址。

/proc/kallsyms 是一个由运行中的内核动态生成的虚拟文件，它反映了当前运行的内核的状态。

通常这个节点是不会打开的，因为会增加内核大小，要使用这个节点，需要打开内核选项：

```
CONFIG_KALLSYMS=y
```

如果想获取 do_fork 函数的地址，可以使用以下命令：

```
cat /proc/kallsyms | grep " do_fork"
```

然后会返回一个包含 do_fork 函数地址的行，如：

```
ffffffff810b57b0 T do_fork
```

输出内容表示 `do_fork` 函数的地址是 `ffffffff810b57b0` 。

4、内核接口

也可以在代码中获取内核符号表，同样需要打开内核选项 `CONFIG_KALLSYMS=y` 。

kallsyms_lookup_name

- 已知函数名，获取地址：

```
kallsyms_lookup_name("函数名" )
```

该函数会返回对应函数名的地址。

sprint_symbol

- 已知地址，返回对应符号：

```
#include <linux/kallsyms.h>

int sprint_symbol(char *buffer, unsigned long address)
```

- buffer：符号名缓存区，保存结果。
- address：符号地址。

end

往期推荐

直到我干了底层开发，才知道不写业务代码有多爽

你解决bug的能力，暴露了你的水平

入职Linux驱动工程师后，我才知道的真相.....

很底层的性能优化：让CPU更快地执行你的代码

薪资倒挂，大家都沉默了...

机遇：我是如何走向Linux驱动的...

当我用几道题考了一遍做Linux驱动的同事.....

嵌入式Linux充电站
作者Vincent，分享一些嵌入式Linux、内核、RISC-V等知识。学习、沉淀、分享，才能有...
100篇原创内容

公众号

linux内核 28

linux内核 · 目录

上一篇
底层开发必知的三个内存结构概念

下一篇
Linux驱动工程师必知的三种获取结构体地址的方法