

# 第 0019 讲 8 进程虚拟区间实战分析

## 一、Linux 内核，进程虚拟地址

1、Linux 内核中，每个进程都有自己的虚拟地址空间，其中包含用户态和内核态两部分。

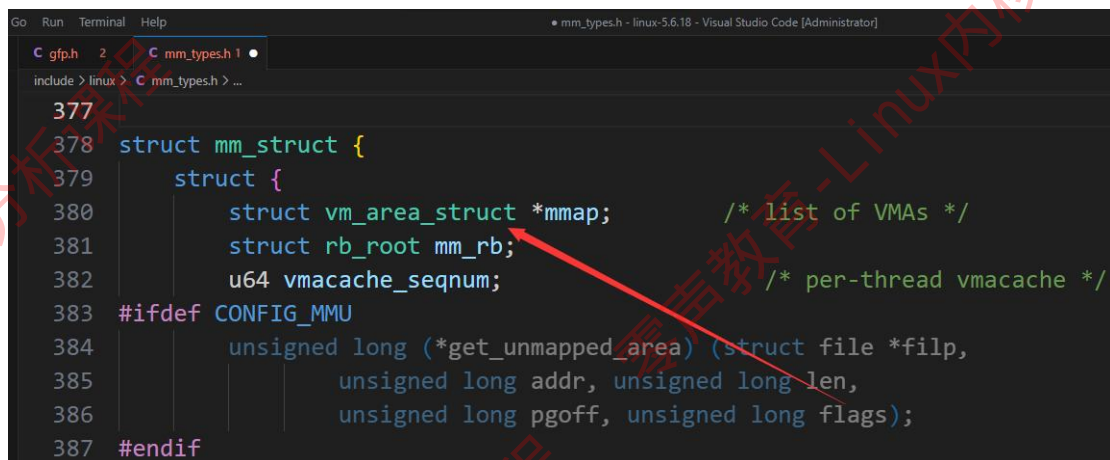
## 2、Linux 进程虚拟地址空间组成

- 用户态虚拟地址空间
- 内核态虚拟地址空间
- 内核栈
- 共享库区域
- 内核模块区域

## 3、输出进程虚拟区间

要输出进程虚拟区间，则必须要掌握两个结构体类型，分别为：

struct mm\_struct/struct vm\_area\_struct，其中源码如下：



```
377
378 struct mm_struct {
379     struct {
380         struct vm_area_struct *mmap;           /* list of VMAs */
381         struct rb_root mm_rb;
382         u64 vmacache_seqnum;                  /* per-thread vmacache */
383 #ifdef CONFIG_MMU
384         unsigned long (*get_unmapped_area) (struct file *filp,
385         unsigned long addr, unsigned long len,
386         unsigned long pgoff, unsigned long flags);
387 #endif
```

```
Go Run Terminal Help • mm_types.h - linux-5.6.18 - Visual Studio Code [Administrator]
C gfp.h 2 C mm_types.h 1
include > linux > C mm_types.h > 53 vm_area_struct
285
286 /*
287  * This struct defines a memory VMM memory area. There is one of these
288  * per VM-area/task. A VM area is any part of the process virtual memory
289  * space that has a special rule for the page-fault handlers (ie a shared
290  * library, the executable area etc).
291  */
292 struct vm_area_struct {
293     /* The first cache line has the info for VMA tree walking. */
294
295     unsigned long vm_start;    /* Our start address within vm_mm. */
296     unsigned long vm_end;    /* The first byte after our end address
297                                within vm_mm. */
298
299     /* linked list of VM areas per task, sorted by address */
300     struct vm_area_struct *vm_next, *vm_prev;
301
302     struct rb_node vm_rb;
```

```
// 对进程虚拟区间抽象的数据结构类型
struct vm_area_struct {

    /* The first cache line has the info for VMA tree
walking. */

    // 线性区的第一个线性地址

    // 线性区之后的第一个线性地址

    unsigned long vm_start;    /* Our start address
within vm_mm. */

    unsigned long vm_end;    /* The first byte after
our end address

                                within vm_mm. */
```

```

/* linked list of VM areas per task, sorted by address
*/

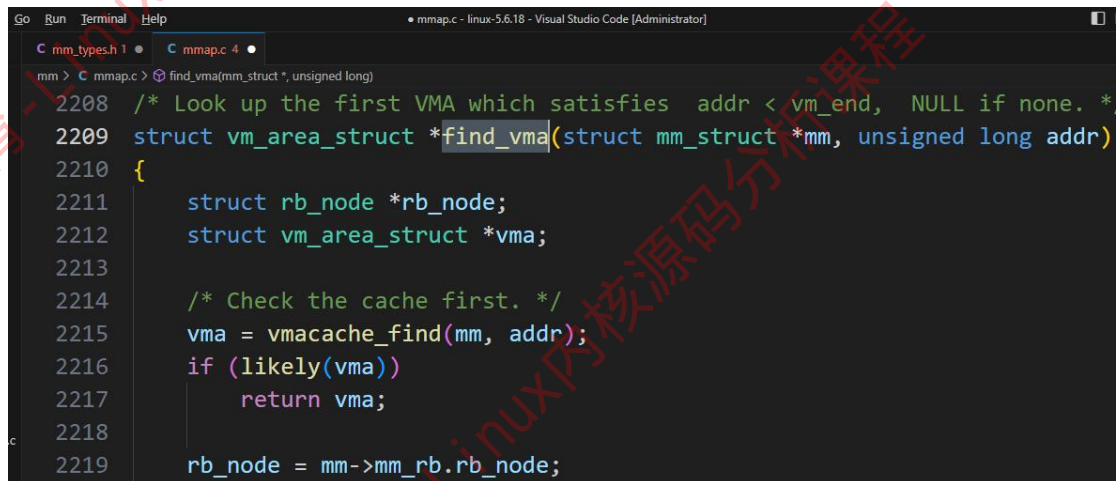
// 进程链表中的下一个线性区及上一个线性区

struct vm_area_struct *vm_next, *vm_prev;

struct rb_node vm_rb; // 用于红黑树的数据

```

#### 4、find\_vma() 函数原型



```

Go Run Terminal Help • mmap.c - linux-5.6.18 - Visual Studio Code [Administrator]
C mm_types.h 1 • C mmap.c 4 •
mm > C mmap.c > find_vma(mm_struct *, unsigned long)
2208 /* Look up the first VMA which satisfies addr < vm_end, NULL if none. */
2209 struct vm_area_struct *find_vma(struct mm_struct *mm, unsigned long addr)
2210 {
2211     struct rb_node *rb_node;
2212     struct vm_area_struct *vma;
2213
2214     /* Check the cache first. */
2215     vma = vmacache_find(mm, addr);
2216     if (likely(vma))
2217         return vma;
2218
2219     rb_node = mm->mm_rb.rb_node;

```

mm: 表示进程整个用户空间的抽象，一个进程只有一个 mm\_struct 结构；

addr: 表示进程用户空间中一虚拟地址，它属于某一虚拟区间。

此函数返回一个结构类型指针，该指针指向描述进程中虚拟地址 addr 所在虚拟区间的结构体。

## 二、项目实战分析

```
root@ubuntu: /home/vico/Desktop/vma
File Edit View Search Terminal Help
root@ubuntu:/home/vico/Desktop/vma# ls
Makefile      Module.symvers  vmaaddress.ko   vmaaddress.mod.c  vmaaddress.o
modules.order  vmaaddress.c    vmaaddress.mod  vmaaddress.mod.o
root@ubuntu:/home/vico/Desktop/vma# insmod vmaaddress.ko
root@ubuntu:/home/vico/Desktop/vma# dmesg -c
[10152.840046] Prompt:addrss = 0x55c921791001
[10152.840047] Prompt:vmap->vm_start = 0x55c921791000
[10152.840047] Prompt:vmap->vm_end = 0x55c921793000
root@ubuntu:/home/vico/Desktop/vma# rmmod vmaaddress.ko
root@ubuntu:/home/vico/Desktop/vma# dmesg -c
[10200.445536] Prompt:Normal exit module.
root@ubuntu:/home/vico/Desktop/vma#
```