list_entry()函数 ——从获取目标进程到linux内核双链表的思想与实现

```
list_entry(task -> task.next, struct task_struct, tasks) //LKD_Chapter_3_page_26
对于给定进程，获取链表中的下一个进程
```

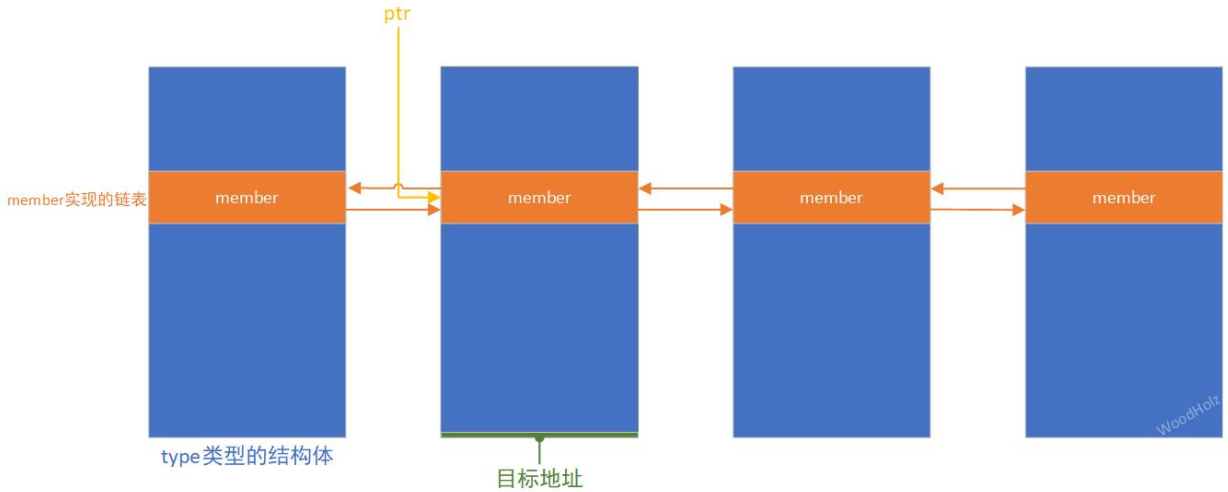
```

#define list_entry(ptr, type, member) \
    container_of(ptr, type, member)

#define container_of(ptr, type, member) ({          \
    const typeof( ((type *)0)->member ) *__mptr = (ptr); // (1)
    (type *)( (char *)__mptr - offsetof(type,member) );}) // (2)

#define offsetof(TYPE, MEMBER) ((size_t) &((TYPE *)0)->MEMBER)
```

用途： 利用结构体的已知**成员**（一般是**包含**在结构体中的指针）获得结构体对象的首地址，即获得结构体指针。

（1）定义一个常量指针__mptr，并将ptr赋值给它。合法性检查

（2）用当前节点地址ptr值剪掉member离type结构体首地址的距离，最后就得到了ptr节点指向的节点的type类型结构体的首地址。

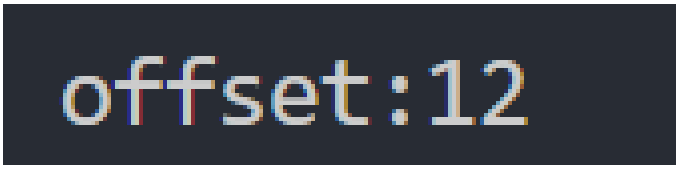
示意图



验证offset计算偏移量

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

```
2
3  /*
4  typedef struct listhead
5  {
6
7      listhead * prev;
8      listhead * next;
9  }listhead;
10 */
11
12 typedef struct
13 {
14     long long num;
15     char name;
16     int num_1;
17     //listhead tasks;
18 }node;
19
20 int main()
21     {
22         printf("offset:%u\n", \
23         &((node *) 0) -> num_1);
24     }
```



感想

1. 利用这个知识应该可以改进之前写的eBPF程序，返回一些有用的数据
2. 发现我的读书学习过分看重次要问题了，是在阅读LKD进程部分时不知道list_entry才去查的，但是留有一个印象就好，应该把重点放在阅读进程，线程主要问题的学习和理解上。