Linux内核中container_of函数详解

[日期：2016-08-22] 来源：Linux社区 作者：Linux [字体：大 中 小]

在**Linux** 内核中，**container_of** 函数使用非常广，例如 **Linux**内核链表 **list_head**、工作队列**work_struct**中。

在Linux 内核中有一个大名鼎鼎的宏container_of()，这个宏是用来干嘛的呢？我们先来看看它在内核中是怎样定义的。

```
/**
 * container_of - cast a member of a structure out to the containing structure
 * @ptr:    the pointer to the member.
 * @type:   the type of the container struct this is embedded in.
 * @member: the name of the member within the struct.
 *
 */
#define container_of(ptr, type, member) ({          \
    const typeof(((type *)0)->member) * __mptr = (ptr); \
    (type *)((char *)__mptr - offsetof(type, member)); })
```

呵呵，乍一看不知道是什么东东。

我们先来分析一下container_of(ptr,type,member),这里面有ptr,type,member分别代表指针、类型、成员。看一个例子:

```
Struct test
 {
        int i;
        int j;
        char k;
 };
 Struct test temp;
```

现在呢如果我想通过temp.j的地址找到temp的首地址就可以使用container_of(&temp.j,struct test,j);

现在我们知道container_of()的作用就是通过一个结构变量中一个成员的地址找到这个结构体变量的首地址。

下面来看看比较复杂的内容:

```
const typeof(((type *)0)->member) * __mptr = (ptr);
(type *)((char *)__mptr - offsetof(type, member));
```

我们用上面的struct test张展一下

Const typeof(((struct test *)0)->j) * __mptr = (&temp.j); **//(sturct test *)0** 表示数据段基址

其中，typeof是GNU C对标准C的扩展，它的作用是根据变量获取变量的类型。因此，上述代码的作用是首先使用typeof获取结构体成员j的类型为int,然后顶一个int指针类型的临时变量__mptr,并将结构体变量中的成员的地址赋给临时变量__mptr。

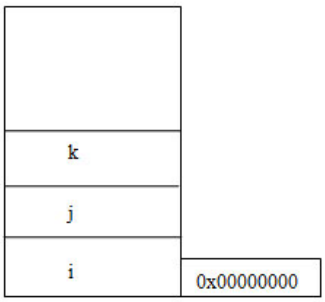
(struct test *)((char *)__mptr - offsetof(struct test,j));

接着我们来看一下offsetof(struct test,j)，他在内核中如下定义

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

展开(size_t)&((struct test *)0)->j,这是什么东东？

一开始也不明白，这里要感谢曹忠明老师的热心帮助，一语惊醒梦中人,呵呵，可以是这样理解。



其中size_t是整型,那么我们可以知道最终的结果是一个整形值，也就是j相对于0地址的偏移量。也许现在你会问，整出这么个玩意干嘛，下面看个列子:

```c
#include <stdio.h>

struct test
{
    char i;
    char  j;
    char k;
};


int main()
{
    struct test temp;

    printf("&temp = %p\n",&temp);
    printf("&temp.k = %p\n",&temp.k);
    printf("&((struct test *)0)->k = %d\n",((size_t)&((struct test *)0)->k));

    return 0;
}
```

程序运行结果:

```
&temp = 0xbfa1022d
&temp.k = 0xbfa1022f
&((struct test *)0)->k = 2
```

发现没有如果把第二个值 减去最后一个值，就能得到第一个值。

在回首一下它:

(struct test *)((char *)__mptr - offsetof(struct test,j));

是不是可以获得结构体变量temp的首地址呀，是不是太精妙了呀，linux内核中随随便便一个宏就有如此精妙，呵呵，想想对linux了解非常多的牛人，还有很长一段路。

本文永久更新链接地址：http://www.linuxidc.com/Linux/2016-08/134481.htm