**container\_of详解**

一、container\_of简介

container\_of是linux内核中实现的一个宏，可以从结构体变量成员的地址得到结构体变量本身的地址。

声明如下：

container\_of(pointer, strcut\_type,mem\_type);

用法如下：

输入三个参数（结构体变量成员的地址，结构体类型，结构体变量成员的名称），返回结构体变量的地址。

二、实现方式

下面是从linux内核中得到的container\_of声明的源码，从中可看到其实现原理。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <linux/kernel.h>

#define container\_of(ptr, type, member) ({ \

const typeof( ((type \*)0)->member ) \*\_\_mptr = (ptr); \

(type \*)( (char \*)\_\_mptr - offsetof(type,member) );})

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <linux/stddef.h>

#undef offsetof

#ifdef \_\_compiler\_offsetof

#define offsetof(TYPE,MEMBER) \_\_compiler\_offsetof(TYPE,MEMBER)

#else

#define offsetof(TYPE, MEMBER) ((size\_t) &((TYPE \*)0)->MEMBER)

#endif

再往下跟踪，在<linux/compiler-gcc4.h>中，可看到

#define \_\_compiler\_offsetof(a,b) \_\_builtin\_offsetof(a,b)

表示\_\_compiler\_offsetof(TYPE,MEMBER)是编译器的内建宏，功能应该类似于((size\_t) &((TYPE \*)0)->MEMBER)

综上：可以看到container\_of(ptr, type, member)相当于下面几步：

{

const typeof( ((type \*)0)->member ) \*\_\_mptr = (ptr);//\_\_mptr指针是member类型，被赋值为ptr

(type \*)( (char \*)\_\_mptr - offsetof(type,member) ); //通过offsetof(type,member)很巧妙的得到member在type中相对位移，从\_\_mptr中减去这个位移，就得到结构体type的起始地址，将起始地址强转成(type \*)类型，就得到了我们需要的东西。

}

三、下面是在应用层实现的类似container\_of功能的一个例子，从中可以体会到container\_of的实现思想。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

struct wyj\_temp

{

int a;

char b;

double c;

};

struct wyj\_temp temp[3]={

{1,'a',1.1},

{2,'b',2.2},

{3,'c',3.3},

};

int main()

{

char \*pch=&(temp[1].b);

struct wyj\_temp \*pt;

pt=(struct wyj\_temp \*)((char \*)pch-(char \*)(&(((struct wyj\_temp \*)0)->b)));//此句最关键：由temp[1].b的地址，得到temp[1]的地址

printf("%c\n",(pt-1)->b);

printf("%d\n",(pt)->a);

printf("%lf\n",(pt+1)->c);

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

当我们需要从type.mem的地址得到type的地址时。可以假设有一个存储在内存0地址处的type结构，因为它是从0地址开始的，所以它的mem成员的地址，就是这个mem成员到type首字节的位移量。有了这个位移量，在任何有type.mem地址的地方，只要向上递减此位移量，就可以得到type的首字节地址（因为type结构内各成员之间相对位置是不变的）。最后再将type首字节的地址转换成type \*类型指针就可以了。

上例的关键点在pt=(struct wyj\_temp \*)((char \*)pch-(char \*)(&(((struct wyj\_temp \*)0)->b)));/请各位详细体会。