```
<mm/mmap.c>

pgprot_t protection_map[16] = {
    __P000, __P001, __P010, __P011, __P100, __P101, __P110, __P111,
    __S000, __S001, __S010, __S011, __S100, __S101, __S110, __S111
};
```

protection_map[]数组的每个成员代表一个属性的组合，如__P000 表示无效的 PTE 属性，__P001 表示只读属性，__P100 表示可执行属性（PAGE_EXECONLY）等。

```
#define __P000  PAGE_NONE
#define __P001  PAGE_READONLY
#define __P010  PAGE_READONLY
#define __P011  PAGE_READONLY
#define __P100  PAGE_EXECONLY
#define __P101  PAGE_READONLY_EXEC
#define __P110  PAGE_READONLY_EXEC
#define __P111  PAGE_READONLY_EXEC

#define __S000  PAGE_NONE
#define __S001  PAGE_READONLY
#define __S010  PAGE_SHARED
#define __S011  PAGE_SHARED
#define __S100  PAGE_EXECONLY
#define __S101  PAGE_READONLY_EXEC
#define __S110  PAGE_SHARED_EXEC
#define __S111  PAGE_SHARED_EXEC
```

下面以只读属性（PAGE_READONLY）来看，它究竟包含哪些页表项的标志位。

```
#define PAGE_READONLY   __pgprot(_PAGE_DEFAULT | PTE_USER | PTE_RDONLY | PTE_NG |
PTE_PXN | PTE_UXN)

#define _PAGE_DEFAULT     (_PROT_DEFAULT | PTE_ATTRINDX(MT_NORMAL))

#define _PROT_DEFAULT     (PTE_TYPE_PAGE | PTE_AF | PTE_SHARED)
```

把上述的宏全部展开，我们可以得到如下页表项的标志位。

- PTE_TYPE_PAGE：表示这是一个基于页面的页表项，即设置页表项的 Bit[1:0]。
- PTE_AF：设置访问位。
- PTE_SHARED：设置内存共享属性。
- MT_NORMAL：设置内存属性为 normal。
- PTE_USER：设置 AP 访问位，允许通过用户权限访问该内存。
- PTE_NG：设置该内存对应的 TLB 只属于该进程。
- PTE_PXN：表示该内存不能在特权模式下执行。
- PTE_UXN：表示该内存不能在用户模式下执行。
- PTE_RDONLY：表示只读属性。

4.4.5 查找 VMA

通过虚拟地址来查找 VMA 是内核中常用的操作，内核提供一个接口函数来实现这个查找