驱动使用 MDL 方式读写内存

转载

(-: LYSM :-) 于 2020-06-24 16:18:16 发布 3343 收藏 17

分类专栏： Windows 驱动开发

原文链接：https://www.write-bug.com/article/2136.html

背景

通常，我们在内核中修改内存的时候，都是通过修改 CR0 寄存器，关闭内存写保护属性，然后再写入内存的方式来修改内存。我个人不喜欢这种方式，因为总感觉我们使用没有线程接口函数的方法，总感觉不太稳定，而且，在 64 位程序下，CR0 方式不再适用了。

所以，我强烈推荐在内核下使用 MDL 方式来修改内存，在 32 位内核和 64 位内核下同样有效。

实现过程

内存描述符列表 (MDL) 是一个系统定义的结构，通过一系列物理地址描述缓冲区。MDL的全称是 Memory Descriptor List，即内存描述符表。可以通过MDL描述一块内存区域，在MDL中包含了该内存区域的起始地址、拥有者进程、字节数量、标记等信息。

MDL 是用来建立一块虚拟地址空间与物理页面之间的映射。对这句话的理解是使用 MDL 来修改内核内存的关键。当我们要对一块内核内存进行修改的时候，我们先为这块内存创建 MDL，那么就会建立一块新的虚拟内存空间，与将要修改内存对应的物理空间相映射。也就是说，同一块物理空间，映射了两块不同的虚拟内存地址。我们可以通过这两个虚拟内存地址，来操作这块物理内存，这便是 MDL 修改内存的实现思路。

那么，使用 MDL 方式修改内存具体的实现流程如下：

- 首先，给定缓冲区的起始地址和长度，调用 MmCreateMdl 函数分配一个足够大的 MDL 结构来映射给定的缓冲区
- 然后，调用 MmBuildMdlForNonPagedPool 函数来更新 MDL 对物理内存的描述。
- 最后，调用 MmMapLockedPages 函数将 MDL 中描述的物理页面映射到虚拟内存中，并返回映射的虚拟内存地址，这样我们就可以通过新映射的虚拟内存地址，操作同一块物理页面了，以此实现修改指定内存的数据。

代码：

```
BOOLEAN MDLWriteMemory(PVOID pBaseAddress, PVOID pWriteData, SIZE_T writeDataSize)
{
    PMDL pMdl = NULL;
    PVOID pNewAddress = NULL;
    // 创建 MDL
    pMdl = MmCreateMdl(NULL, pBaseAddress, writeDataSize);
    if (NULL == pMdl)
    {
        return FALSE;
    }
    // 更新 MDL 对物理内存的描述
    MmBuildMdlForNonPagedPool(pMdl);
    // 映射到虚拟内存中
      pNewAddress = MmMapLockedPages(pMdl, KernelMode);
    if (NULL == pNewAddress)
    {
        IoFreeMdl(pMdl);
    }
    // 写入数据
    RtlCopyMemory(pNewAddress, pWriteData, writeDataSize);
    // 释放
    MmUnmapLockedPages(pNewAddress, pMdl);
    IoFreeMdl(pMdl);
    return TRUE;
}
```