RING0 操作注册表和 RING3 的区别也不大，同样是“获得句柄->执行操作->关闭句柄”的模式，同样也只能使用内核 API 不能使用 WIN32API。不过内核里有一套 RTL 函数，把 Zw 系列的注册表函数进行了封装，也就是说，只剩下“执行操作”这一步了。

接下来说说注册表的本质。注册表其实是文件，它存储在 c:\windows\system32\config 这个目录下（打开目录，看到那几个带锁图标的文件就是。为什么带锁？因为被 SYSTEM 进程独占访问了）。注册表文件被称为 HIVE 文件，此格式是微软专用的，不公开，每个系统都不一定相同，但总体来说变化不大。当系统关机之前，或者调用 ZwFlushKey 时，把内存中的修改存入磁盘。另外，我们用 WINDOWS 自带的注册表编辑器看到的注册表有 5 个根项，其实这是“幻象”，真正的根项只有两个：HKEY_LOCAL_MACHINE 和 HKEY_USERS。HKEY_CLASSES_ROOT 和 HKEY_CURRENT_CONFIG 其实都是 HKEY_LOCAL_MACHINE 的“下属”。HKEY_CURRENT_USER 则是 HKEY_USERS 的“下属”，独立列出来只为了方便操作。

关于注册表的操作不多，无非就是：新建 KEY、重命名 KEY、删除 KEY、新建/设置 VALUE、读取 VALUE、删除 VALUE、枚举子 KEY 和 VALUE。这里之所以用英文，是因为中文的表达较为混乱。比如 KEY，有的翻译为项，有的翻译为键；VALUE 有的翻译为值，有的翻译为键值。所以为了统一说法，就用英文 KEY 和 VALUE 了。

1.新建 KEY

```
void RegCreateKey(LPWSTR KeyName)
{
    OBJECT_ATTRIBUTES objectAttributes;
    UNICODE_STRING usKeyName;
    NTSTATUS ntStatus;
    HANDLE hRegister;
    RtlInitUnicodeString( &usKeyName, KeyName);
    InitializeObjectAttributes(&objectAttributes,
                               &usKeyName,
                               OBJ_CASE_INSENSITIVE,//对大小写敏感
                               NULL,
                               NULL );
    ZwCreateKey(&hRegister,  KEY_ALL_ACCESS,  &objectAttributes,  0,  NULL,
REG_OPTION_NON_VOLATILE, NULL);
    ZwClose(hRegister);
}
```

2.重命名 KEY（这里用到的关键 API『ZwRenameKey』没有导出，需要自己定位，大家可以先用硬编码测试。在 WINDNG 里输入 x nt!ZwRenameKey 即可获得函数地址）

```
typedef NTSTATUS (__fastcall *ZWRENAMEKEY)
(IN HANDLE KeyHandle,
IN PUNICODE_STRING ReplacementName);

//修改这个地址!!
ZWRENAMEKEY ZwRenameKey=0xFFFFF80012345678;
```

```
void RegRenameKey(UNICODE_STRING usOldKeyName, UNICODE_STRING usNewKeyName)
{
    OBJECT_ATTRIBUTES objectAttributes;
    HANDLE hRegister;
    NTSTATUS ntStatus;
    InitializeObjectAttributes(&objectAttributes,
                               &usOldKeyName,
                               OBJ_CASE_INSENSITIVE,//对大小写敏感
                               NULL,
                               NULL );
    ntStatus = ZwOpenKey( &hRegister, KEY_ALL_ACCESS, &objectAttributes);
    if(NT_SUCCESS(ntStatus))
    {
        ntStatus = ZwRenameKey(hRegister,&usNewKeyName);
        ZwFlushKey(hRegister);
        ZwClose(hRegister);
    }
}
```

3.删除 KEY

```
void RegDeleteKey(LPWSTR KeyName)
{
    OBJECT_ATTRIBUTES objectAttributes;
    UNICODE_STRING usKeyName;
    NTSTATUS ntStatus;
    HANDLE hRegister;
    RtlInitUnicodeString( &usKeyName, KeyName);
    InitializeObjectAttributes(&objectAttributes,
                               &usKeyName,
                               OBJ_CASE_INSENSITIVE,//对大小写敏感
                               NULL,
                               NULL );
    ntStatus = ZwOpenKey( &hRegister, KEY_ALL_ACCESS, &objectAttributes);
    if (NT_SUCCESS(ntStatus))
    {
        ntStatus = ZwDeleteKey(hRegister);
        ZwClose(hRegister);
    }
}
```

4.新建/设置 VALUE

```
void RegSetValueKey(LPWSTR REG_KEY_NAME, LPWSTR REG_VALUE_NAME, DWORD DataType, PVOID DataBuffer, DWORD DataLength)
```

```
{
    OBJECT_ATTRIBUTES objectAttributes;
    UNICODE_STRING usKeyName,usValueName;
    NTSTATUS ntStatus;
    HANDLE hRegister;
    ULONG Type;
    RtlInitUnicodeString(&usKeyName, REG_KEY_NAME);
    RtlInitUnicodeString(&usValueName, REG_VALUE_NAME);
    InitializeObjectAttributes(&objectAttributes,
                               &usKeyName,
                               OBJ_CASE_INSENSITIVE,//对大小写敏感
                               NULL,
                               NULL );
    ntStatus = ZwOpenKey(&hRegister, KEY_ALL_ACCESS, &objectAttributes);
    if (NT_SUCCESS(ntStatus))
    {
        ntStatus = ZwSetValueKey(hRegister, &usValueName, 0, DataType,
DataBuffer, DataLength);
        ZwFlushKey(hRegister);
        ZwClose(hRegister);
    }
}
```

5.读取 VALUE

```
typedef struct _KEY_VALUE_PARTIAL_INFORMATION {
  ULONG TitleIndex;
  ULONG Type;
  ULONG DataLength;
  UCHAR Data[1];
} KEY_VALUE_PARTIAL_INFORMATION, *PKEY_VALUE_PARTIAL_INFORMATION;

NTSTATUS RegQueryValue(UNICODE_STRING usKeyName, PUNICODE_STRING pValueName,
PKEY_VALUE_PARTIAL_INFORMATION *pkvpi)
{
    ULONG ulSize;
    NTSTATUS ntStatus;
    PKEY_VALUE_PARTIAL_INFORMATION pvpi;
    OBJECT_ATTRIBUTES objectAttributes;
    HANDLE hRegister;
    InitializeObjectAttributes(&objectAttributes,
                               &usKeyName,
                               OBJ_CASE_INSENSITIVE,//对大小写敏感
                               NULL,
                               NULL );
```

```
    ntStatus = ZwOpenKey( &hRegister, KEY_ALL_ACCESS, &objectAttributes);
    if(!ntStatus)
    {
        return ntStatus;
    }
    ntStatus = ZwQueryValueKey(hRegister,
                               pValueName,
                               KeyValuePartialInformation ,
                               NULL,
                               0,
                               &ulSize);
    if (ntStatus==STATUS_OBJECT_NAME_NOT_FOUND || ulSize==0)
    {
        return STATUS_UNSUCCESSFUL;
    }
    pvpi = (PKEY_VALUE_PARTIAL_INFORMATION)ExAllocatePool(PagedPool,ulSize);
    ntStatus = ZwQueryValueKey(hRegister,
                               pValueName,
                               KeyValuePartialInformation ,
                               pvpi,
                               ulSize,
                               &ulSize);
    if (!NT_SUCCESS(ntStatus))
    {
        return STATUS_UNSUCCESSFUL;
    }
    *pkvpi=pvpi;    //这里的 pvpi 是没有释放的，用完要释放 ExFreePool(pvpi);
    return STATUS_SUCCESS;
}
```

6.删除 VALUE

```
void RegDeleteValueKey(LPWSTR REG_KEY_NAME, LPWSTR REG_VALUE_NAME)
{
    OBJECT_ATTRIBUTES objectAttributes;
    UNICODE_STRING usKeyName,usValueName;
    NTSTATUS ntStatus;
    HANDLE hRegister;
    RtlInitUnicodeString(&usKeyName, REG_KEY_NAME);
    RtlInitUnicodeString(&usValueName, REG_VALUE_NAME);
    InitializeObjectAttributes(&objectAttributes,
                               &usKeyName,
                               OBJ_CASE_INSENSITIVE,//对大小写敏感
                               NULL,
                               NULL );
```

```
    ntStatus = ZwOpenKey(&hRegister, KEY_ALL_ACCESS, &objectAttributes);
    if (NT_SUCCESS(ntStatus))
    {
        ntStatus = ZwDeleteValueKey(hRegister,&usValueName);
        ZwFlushKey(hRegister);
        ZwClose(hRegister);
    }
}
```

7.枚举子 KEY

```
VOID EnumerateSubItemRegTest()
{
    #define                                          MY_REG_SOFTWARE_KEY_NAME
    L"\\Registry\\Machine\\Software\\xxxxxxxx"
    UNICODE_STRING RegUnicodeString;
    HANDLE hRegister;
    //初始化 UNICODE_STRING 字符串
    RtlInitUnicodeString( &RegUnicodeString, MY_REG_SOFTWARE_KEY_NAME);
    OBJECT_ATTRIBUTES objectAttributes;
    //初始化 objectAttributes
    InitializeObjectAttributes(&objectAttributes,
                               &RegUnicodeString,
                               OBJ_CASE_INSENSITIVE,//对大小写敏感
                               NULL,
                               NULL );
    //打开注册表
    NTSTATUS ntStatus = ZwOpenKey( &hRegister,
                               KEY_ALL_ACCESS,
                               &objectAttributes);
    if (NT_SUCCESS(ntStatus))
    {
        KdPrint(("Open register successfully\n"));
    }
    ULONG ulSize;
    //第一次调用 ZwQueryKey 为了获取 KEY_FULL_INFORMATION 数据的长度
    ZwQueryKey(hRegister,
        KeyFullInformation,
        NULL,
        0,
        &ulSize);
    PKEY_FULL_INFORMATION pfi =
        (PKEY_FULL_INFORMATION)
        ExAllocatePool(PagedPool,ulSize);
    //第二次调用 ZwQueryKey 为了获取 KEY_FULL_INFORMATION 数据的数据
```

```
    ZwQueryKey(hRegister,
        KeyFullInformation,
        pfi,
        ulSize,
        &ulSize);
    for (ULONG i=0;i<pfi->SubKeys;i++)
    {
        //第一次调用 ZwEnumerateKey 为了获取 KEY_BASIC_INFORMATION 数据的长度
        ZwEnumerateKey(hRegister,
                i,
                KeyBasicInformation,
                NULL,
                0,
                &ulSize);
        PKEY_BASIC_INFORMATION pbi =
            (PKEY_BASIC_INFORMATION)
            ExAllocatePool(PagedPool,ulSize);
        //第二次调用 ZwEnumerateKey 为了获取 KEY_BASIC_INFORMATION 数据的数据
        ZwEnumerateKey(hRegister,
                i,
                KeyBasicInformation,
                pbi,
                ulSize,
                &ulSize);
        UNICODE_STRING uniKeyName;
        uniKeyName.Length =
        uniKeyName.MaximumLength =
        (USHORT)pbi->NameLength;
        uniKeyName.Buffer = pbi->Name;
        KdPrint(("The %d sub item name:%wZ\n",i,&uniKeyName));
        ExFreePool(pbi);
    }
    ExFreePool(pfi);
    ZwClose(hRegister);
}
```

8.枚举子 VALUE

```
VOID EnumerateSubValueRegTest()
{
    #define                                          MY_REG_SOFTWARE_KEY_NAME
    L"\\Registry\\Machine\\Software\\xxxxxxxx"
    UNICODE_STRING RegUnicodeString;
    HANDLE hRegister;
    //初始化 UNICODE_STRING 字符串
```

```
RtlInitUnicodeString( &RegUnicodeString, MY_REG_SOFTWARE_KEY_NAME);
OBJECT_ATTRIBUTES objectAttributes;
//初始化 objectAttributes
InitializeObjectAttributes(&objectAttributes,
                           &RegUnicodeString,
                           OBJ_CASE_INSENSITIVE,//对大小写敏感
                           NULL,
                           NULL );
//打开注册表
NTSTATUS ntStatus = ZwOpenKey( &hRegister,
                           KEY_ALL_ACCESS,
                           &objectAttributes);
if (NT_SUCCESS(ntStatus))
{
    KdPrint(("Open register successfully\n"));
}
ULONG ulSize;
//查询 VALUE 的大小
ZwQueryKey(hRegister,
    KeyFullInformation,
    NULL,
    0,
    &ulSize);
PKEY_FULL_INFORMATION pfi =
    (PKEY_FULL_INFORMATION)
    ExAllocatePool(PagedPool,ulSize);
ZwQueryKey(hRegister,
    KeyFullInformation,
    pfi,
    ulSize,
    &ulSize);
for (ULONG i=0;i<pfi->Values;i++)
{
    //查询单个 VALUE 的大小
    ZwEnumerateValueKey(hRegister,
            i,
            KeyValueBasicInformation,
            NULL,
            0,
            &ulSize);
    PKEY_VALUE_BASIC_INFORMATION pvbi =
        (PKEY_VALUE_BASIC_INFORMATION)
        ExAllocatePool(PagedPool,ulSize);
    //查询单个 VALUE 的详情
```

```
        ZwEnumerateValueKey(hRegister,
                i,
                KeyValueBasicInformation,
                pvbi,
                ulSize,
                &ulSize);
        UNICODE_STRING uniKeyName;
        uniKeyName.Length =
        uniKeyName.MaximumLength =
        (USHORT)pvbi->NameLength;
        uniKeyName.Buffer = pvbi->Name;
        KdPrint(("The %d sub value name:%wZ\n",i,&uniKeyName));
        if (pvbi->Type==REG_SZ)
        {
            KdPrint(("The sub value type:REG_SZ\n"));
        }
        else if (pvbi->Type==REG_MULTI_SZ)
        {
            KdPrint(("The sub value type:REG_MULTI_SZ\n"));
        }
        else if (pvbi->Type==REG_DWORD)
        {
            KdPrint(("The sub value type:REG_DWORD\n"));
        }
        else if (pvbi->Type==REG_BINARY)
        {
            KdPrint(("The sub value type:REG_BINARY\n"));
        }
        ExFreePool(pvbi);
    }
    ExFreePool(pfi);
    ZwClose(hRegister);
}
```

最后总结一下几个常见的、和注册表相关的 Zw 函数的功能（详细说明可以到此处查看，用 Ctrl+F 打开搜索，然后寻找 KEY 关键字即可）：

ZwCreateKey	创建 KEY
ZwDeleteKey	删除 KEY
ZwDeleteValueKey	删除 VALUE
ZwEnumerateKey	枚举 KEY
ZwEnumerateValueKey	枚举 VALUE
ZwFlushKey	把注册表缓存的内容写入磁盘
ZwOpenKey	打开 KEY
ZwQueryKey	查询 KEY 的信息

ZwQueryValueKey	查询 VALUE 的信息
ZwSetValueKey	设置 VALUE 的内容

课后作业：大家试一下把这些源码组装起来，弄成一个简易注册表编辑器。