让我们继续在《内核读写内存浮点数》的基础之上做一个简单的延申，如何实现多级偏移读写，其实很简单，读写函数无需改变，只是在读写之前提前做好计算工作，以此来得到一个内存偏移值，并通过调用内存写入原函数实现写出数据的目的。

以读取偏移内存为例，如下代码同样来源于本人的`LyMemory`读写驱动项目，其中核心函数为`WIN10_ReadDeviationIntMemory()`该函数的主要作用是通过用户传入的基地址与偏移值，动态计算出当前的动态地址。

函数首先将基地址指向要读取的变量，并将其转换为`LPCVOID`类型的指针。然后将指向变量值的缓冲区转换为`LPVOID`类型的指针。接下来，函数使用`PsLookupProcessByProcessId`函数查找目标进程并返回其`PEPROCESS`结构体。随后，函数从偏移地址数组的最后一个元素开始迭代，每次循环都从目标进程中读取4字节整数型数据，并将其存储在`Value`变量中。然后，函数将基地址指向`Value`和偏移地址的和，以便在下一次循环中读取更深层次的变量。最后，函数将基地址指向最终变量的地址，读取变量的值，并返回。

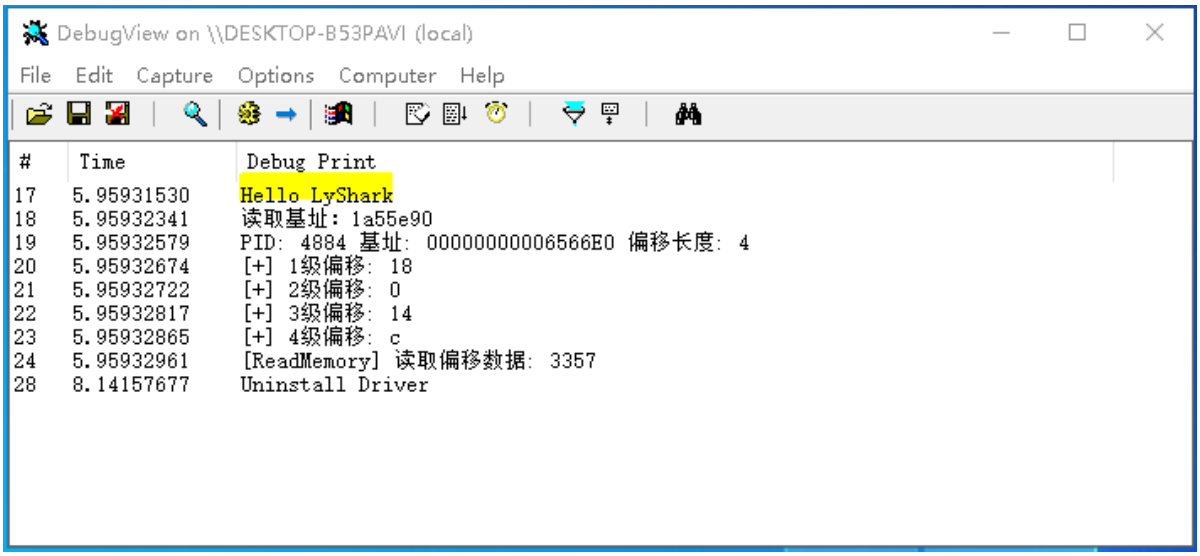
如下案例所示，用户传入进程基址以及`offset`偏移值时，只需要动态计算出该偏移地址，并与基址相加即可得到动态地址。

```
#include <ntifs.h>
#include <ntintsafe.h>
#include <windef.h>

// 普通Ke内存读取
NTSTATUS KeReadProcessMemory(PEPROCESS Process, PVOID SourceAddress, PVOID
TargetAddress, SIZE_T Size)
{
    PEPROCESS SourceProcess = Process;
    PEPROCESS TargetProcess = PsGetCurrentProcess();
    SIZE_T Result;
    if (NT_SUCCESS(MmCopyVirtualMemory(SourceProcess, SourceAddress,
TargetProcess, TargetAddress, Size, KernelMode, &Result)))
        return STATUS_SUCCESS;
    else
        return STATUS_ACCESS_DENIED;
}

// 读取整数内存多级偏移
/*
  Pid: 目标进程的进程ID。
  Base: 变量的基地址。
  offset: 相对基地址的多级偏移地址，用于定位变量。
  len: 偏移地址的数量。
*/
INT64 WIN10_ReadDeviationIntMemory(HANDLE Pid, LONG Base, DWORD offset[32], DWORD
len)
{
    INT64 Value = 0;
    LPCVOID pbase = (LPCVOID)Base;
    LPVOID rbuffer = (LPVOID)&Value;

    PEPROCESS Process;
    PsLookupProcessByProcessId((HANDLE)Pid, &Process);

    for (int x = len - 1; x >= 0; x--)
    {
```

```
        __try
        {
            KeReadProcessMemory(Process, pbase, rbuffer, 4);
            pbase = (LPCVOID)(Value + offset[x]);
        }
        __except (EXCEPTION_EXECUTE_HANDLER)
        {
            return 0;
        }
    }

    __try
    {
        DbgPrint("读取基址：%x \n", pbase);
        KeReadProcessMemory(Process, pbase, rbuffer, 4);
    }
    __except (EXCEPTION_EXECUTE_HANDLER)
    {
        return 0;
    }

    return Value;
}

// 驱动卸载例程
VOID UnDriver(PDRIVER_OBJECT driver)
{
    DbgPrint("Uninstall Driver \n");
}

// 驱动入口地址
NTSTATUS DriverEntry(IN PDRIVER_OBJECT Driver, PUNICODE_STRING RegistryPath)
{
    DbgPrint("Hello LyShark \n");

    DWORD PID = 4884;
    LONG PBase = 0x6566e0;
    LONG Size = 4;
    DWORD Offset[32] = { 0 };

    Offset[0] = 0x18;
    Offset[1] = 0x0;
    Offset[2] = 0x14;
    Offset[3] = 0x0c;

    // 读取内存数据
    INT64 read = WIN10_ReadDeviationIntMemory(PID, PBase, Offset, Size);

    DbgPrint("PID: %d 基址: %p 偏移长度: %d \n", PID, PBase, Size);
    DbgPrint("[+] 1级偏移: %x \n", Offset[0]);
    DbgPrint("[+] 2级偏移: %x \n", Offset[1]);
    DbgPrint("[+] 3级偏移: %x \n", Offset[2]);
    DbgPrint("[+] 4级偏移: %x \n", Offset[3]);

    DbgPrint("[ReadMemory] 读取偏移数据: %d \n", read);
```

```
    Driver->DriverUnload = UnDriver;
    return STATUS_SUCCESS;
}
```

编译并运行如上这段代码，则可获取到 `PID=4884` 的 `PBase` 的动态地址中的数据，如下图所示；



至于如何将数据写出四级偏移的基址上面，则只需要取出 `pbase` 里面的基址，并通过原函数 `WIN10_WriteProcessMemory` 直接写出数据即可，此出的原函数在 `《内核MDL读写进程内存》` 中已经做了详细介绍，实现写出代码如下所示；

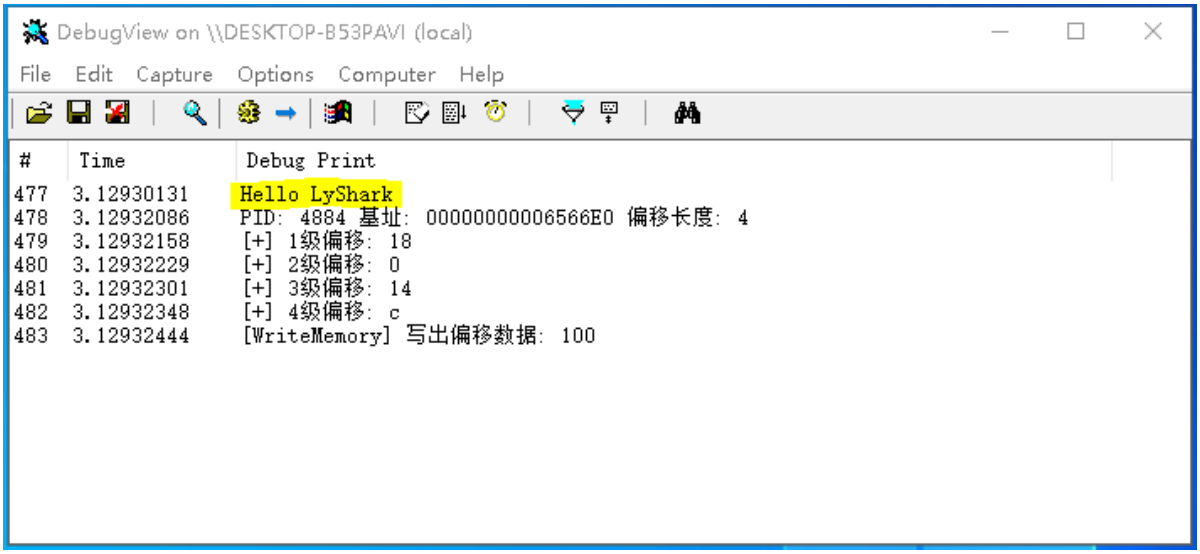
```
#include <ntifs.h>
#include <ntintsafe.h>
#include <windef.h>

// 普通Ke内存读取
NTSTATUS KeReadProcessMemory(PEPROCESS Process, PVOID SourceAddress, PVOID
TargetAddress, SIZE_T Size)
{
    PEPROCESS SourceProcess = Process;
    PEPROCESS TargetProcess = PsGetCurrentProcess();
    SIZE_T Result;
    if (NT_SUCCESS(MmCopyVirtualMemory(SourceProcess, SourceAddress,
TargetProcess, TargetAddress, Size, KernelMode, &Result)))
        return STATUS_SUCCESS;
    else
        return STATUS_ACCESS_DENIED;
}

// Win10 内存写入函数
BOOLEAN WIN10_WriteProcessMemory(HANDLE Pid, PVOID Address, SIZE_T BYTE_size,
PVOID VirtualAddress)
{
    PVOID buff1;
    VOID *buff2;
    int MemoryNumerical = 0;
    KAPC_STATE KAPC = { 0 };

    PEPROCESS Process;
    PsLookupProcessByProcessId((HANDLE)Pid, &Process);
```

```
    __try
    {
        //分配内存
        buff1 = ExAllocatePoolWithTag((POOL_TYPE)0, BYTE_size, 1997);
        buff2 = buff1;
        *(int*)buff1 = 1;
        if (MmIsAddressValid((PVOID)VirtualAddress))
        {
            // 复制内存
            memcpy(buff2, VirtualAddress, BYTE_size);
        }
        else
        {
            return FALSE;
        }

        // 附加到要读写的进程
        KeStackAttachProcess((PRKPROCESS)Process, &KAPC);
        if (MmIsAddressValid((PVOID)Address))
        {
            // 判断地址是否可写
            ProbeForWrite(Address, BYTE_size, 1);
            // 复制内存
            memcpy(Address, buff2, BYTE_size);
        }
        else
        {
            return FALSE;
        }
        // 剥离附加的进程
        KeUnstackDetachProcess(&KAPC);
        ExFreePoolWithTag(buff2, 1997);
    }
    __except (EXCEPTION_EXECUTE_HANDLER)
    {
        return FALSE;
    }
    return FALSE;
}

// 写入整数内存多级偏移
INT64 WIN10_WriteDeviationIntMemory(HANDLE Pid, LONG Base, DWORD offset[32],
DWORD len, INT64 SetValue)
{
    INT64 Value = 0;
    LPCVOID pbase = (LPCVOID)Base;
    LPVOID rbuffer = (LPVOID)&Value;

    PEPROCESS Process;
    PsLookupProcessByProcessId((HANDLE)Pid, &Process);

    for (int x = len - 1; x >= 0; x--)
    {
        __try
        {
            KeReadProcessMemory(Process, pbase, rbuffer, 4);
```

```
            pbase = (LPCVOID)(Value + offset[x]);
        }
        __except (EXCEPTION_EXECUTE_HANDLER)
        {
            return 0;
        }
    }

    __try
    {
        KeReadProcessMemory(Process, pbase, rbuffer, 4);
    }
    __except (EXCEPTION_EXECUTE_HANDLER)
    {
        return 0;
    }

    // 使用原函数写入
    BOOLEAN ref = WIN10_WriteProcessMemory(Pid, (void *)pbase, 4, &SetValue);
    if (ref == TRUE)
    {
        DbgPrint("[内核写成功] # 写入地址: %x \n", pbase);
        return 1;
    }
    return 0;
}

// 驱动卸载例程
VOID UnDriver(PDRIVER_OBJECT driver)
{
    DbgPrint("Uninstall Driver \n");
}

// 驱动入口地址
NTSTATUS DriverEntry(IN PDRIVER_OBJECT Driver, PUNICODE_STRING RegistryPath)
{
    DbgPrint("Hello LyShark \n");

    DWORD PID = 4884;
    LONG PBase = 0x6566e0;
    LONG Size = 4;
    INT64 SetValue = 100;

    DWORD Offset[32] = { 0 };

    Offset[0] = 0x18;
    Offset[1] = 0x0;
    Offset[2] = 0x14;
    Offset[3] = 0x0c;

    // 写出内存数据
    INT64 write = WIN10_WriteDeviationIntMemory(PID, PBase, Offset, Size,
SetValue);

    DbgPrint("PID: %d 基址: %p 偏移长度: %d \n", PID, PBase, Size);
    DbgPrint("[+] 1级偏移: %x \n", Offset[0]);
```

```
    DbgPrint("[+] 2级偏移: %x \n", Offset[1]);
    DbgPrint("[+] 3级偏移: %x \n", Offset[2]);
    DbgPrint("[+] 4级偏移: %x \n", Offset[3]);

    DbgPrint("[WriteMemory] 写出偏移数据: %d \n", SetValue);

    Driver->DriverUnload = UnDriver;
    return STATUS_SUCCESS;
}
```

运行如上代码将在`0x6566e0`所在的基址上，将数据替换为`100`，实现效果图如下所示；



那么如何实现读写内存浮点数，字节集等多级偏移呢?

其实我们可以封装一个`WIN10_ReadDeviationMemory`函数，让其只计算得出偏移地址，而所需要写出的类型则根据自己的实际需求配合不同的写入函数完成，也就是将两者分离开，如下则是一段实现计算偏移的代码片段，该代码同样来自于本人的`LyMemory`驱动读写项目；

```
#include <ntifs.h>
#include <ntintsafe.h>
#include <windef.h>

// 普通Ke内存读取
NTSTATUS KeReadProcessMemory(PEPROCESS Process, PVOID SourceAddress, PVOID
TargetAddress, SIZE_T Size)
{
    PEPROCESS SourceProcess = Process;
    PEPROCESS TargetProcess = PsGetCurrentProcess();
    SIZE_T Result;
    if (NT_SUCCESS(MmCopyVirtualMemory(SourceProcess, SourceAddress,
TargetProcess, TargetAddress, Size, KernelMode, &Result)))
        return STATUS_SUCCESS;
    else
        return STATUS_ACCESS_DENIED;
}

// 读取多级偏移内存动态地址
DWORD64 WIN10_ReadDeviationMemory(HANDLE Pid, LONG Base, DWORD offset[32], DWORD
len)
{
```

```
    INT64 Value = 0;
    LPCVOID pbase = (LPCVOID)Base;
    LPVOID rbuffer = (LPVOID)&Value;

    PEPROCESS Process;
    PsLookupProcessByProcessId((HANDLE)Pid, &Process);

    for (int x = len - 1; x >= 0; x--)
    {
        __try
        {
            KeReadProcessMemory(Process, pbase, rbuffer, 4);
            pbase = (LPCVOID)(Value + offset[x]);
        }
        __except (EXCEPTION_EXECUTE_HANDLER)
        {
            return 0;
        }
    }

    return pbase;
}

// 驱动卸载例程
VOID UnDriver(PDRIVER_OBJECT driver)
{
    DbgPrint("Uninstall Driver \n");
}

// 驱动入口地址
NTSTATUS DriverEntry(IN PDRIVER_OBJECT Driver, PUNICODE_STRING RegistryPath)
{
    DbgPrint("Hello LyShark \n");

    DWORD PID = 4884;
    LONG PBase = 0x6566e0;
    LONG Size = 4;

    DWORD Offset[32] = { 0 };

    Offset[0] = 0x18;
    Offset[1] = 0x0;
    Offset[2] = 0x14;
    Offset[3] = 0x0c;

    // 写出内存数据
    DWORD64 offsets = WIN10_ReadDeviationMemory(PID, PBase, Offset, Size);

    DbgPrint("PID: %d 基址: %p 偏移长度: %d \n", PID, PBase, Size);
    DbgPrint("[+] 1级偏移: %x \n", Offset[0]);
    DbgPrint("[+] 2级偏移: %x \n", Offset[1]);
    DbgPrint("[+] 3级偏移: %x \n", Offset[2]);
    DbgPrint("[+] 4级偏移: %x \n", Offset[3]);

    DbgPrint("[CheckMemory] 计算偏移地址: %x \n", offsets);
```

```
    Driver->DriverUnload = UnDriver;
    return STATUS_SUCCESS;
}
```

运行如上代码将动态计算出目前偏移地址的 `pbase` 实际地址，实现效果图如下所示；