DEVGURU

Driver와 Application 간의 메모리 공유 방법

대부분 디바이스 드라이버 프로그래머들은 user-mode Apllication 과 드라이버 간에 메모리를 공유 하기를 원하였고, 여러가지 다양한 방법으로 user-mode Apllication 과 드라이버 간의 메모리 공유를 하였다.
다음은 가장 쉬운 user-mode Apllication 과 드라이버 간의 메모리 공유 기술이다 :

- Apllication은 IOCTL Code를 이용하여 Driver와 공유할 buffer의 pointer를 Driver에게 보낸다. 그후 그 buffer를 공유한다.
- Driver는 memory block을 할당 하고, memory block을 특정 user-mode Process의 주소 공간으로 변환한다. 그후 이 주소를 application에게 return 한다.

이 방법 외에도 몇 개의 다른 방법으로 memory sharing을 구현할수 있다. (paging file or memory mapped file)

IOCTL 을 이용한 Buffer 공유 방법:

IOCTL 은 Driver 와 user-mode 간의 메모리 공유 방법에서 가장 간단한 방법이다.
Application에서 할당 받은 공유할 buffer의 base address와 길이를 DeviceIoControl()의 매개변수로 하여 호출한다.
IOCTL 을 이용한 방법에서두 2가지의 transfer type을 사용하여 메모리 공유를 할수 있다.
- METHOD_DIRECT(using MDL)
  이 transfer type을 사용한다면 user buffer는 locked 된 상태이고 driver는 이 buffer를 사용하기 위하여 MmGetSystemAddressForMdlSafe() 함수를 이용하여 Kernel virtual address space안에 mapping 되어있는 address를 얻어서 사용한다. 이 transfer type을 사용한다면 driver는 어떠한 IRQL과 어떠한 process(arbitrary process)에서 사용 가능 하다.

- METHOD_NEITHER
  이 transfer type을 이용한 메모리 공유 방법에는 몇가지 제한과 경고를 가지고 있다.
Driver는 반드시 요청을 내려보낸 process의 context안에서 buffer를 접근 해야한다.
그 이유는 buffer는 user virtual address space에 존재 하기 때문이고, 이로 인하여 driver는 device stack 중에 가장 위단(top of stack)에 존재해야 한다. top of stack Driver 보다 아래(하위 layer)단의 intermediate of file system drivers는 사용할수 없으며 항상 PASSSIVE_LEVEL에서만 사용 가능하다는 것이다. 그 이유는 I/O Manager 이 user

buffer를 locked 시키지 않기 때문에 언제 page out이 일어 날지 모르는 일이다. 이 메모리를 lock 시키고 싶다면 MDL을 사용해야 할것이다.
또 다른 재약은 User application은 non-cached memory 또는 physically contiguous memory를 할당 할수 없다는 것이다.

IOCTLs를 이용한 방법에는 조심해야(하지 말아야) 할 사항들이 있다. IOCTL를 Complete 한 후에 buffer를 접근해서는 안된다. Application이 갑자기 종료 되어 을 때 Driver는 무심코 전혀 상관 없는 memory에 overwrite 하게 되고 언제가는 system crash 일어 날것이다.
또 하나는 METHOD_DIRECT를 사용 할때, MDL을 포함한 Irp를 complete 시킨후 이전에 MmGetSystemAddressForMdlSafe() 함수를 이용하여 얻은 Kernel virtual address를 접근하려고 시도 하려 할 때 이것은 system crash를 발생할것이다. Driver writter들은 이러한 문제점들을 피해야 한다.

Example)
Application :

```
#define DEV_SHAREMEM CTL_CODE(FILE_DEVICE_UNKNOWN,0x800, METHOD_NEITHER,
                                                        FILE_ANY_ACCESS)
main()
{
        …
        BYTE * pbuffer;
        Pbuffer = malloc(…);
        …
        DeviceIoControl(…,pbuffer, size, … );
}
```

Driver :

```
CTL_CODE(FILE_DEVICE_UNKNOWN,0x800, METHOD_NEITHER,
FILE_ANY_ACCESS)
DispatchDeviceIoControl(…)
{
        …
        BYTE * pShareMem = pIrp->UserBuffer;
        …
        …
}
```

Mapping Kernel Memory To User Mode:

두번째 방법에서는 Kernel mode에서 buffer를 할당 받고, 특정 process의 user virtual address spacce에 mapping을 하는 방법이다.
이 방법은 몇 개의 Kernel mode 함수를 이용하여 아주 간단하게 구현 할 수 있다.

드라이버는 공유할 메모리를 할당할 것이다. 특정 목적에 사용사용 할 것이라면, 예들 들어 DMA 전송을 위하여 사용할것이라면 AllocateCommonBuffer()를 이용할 것이다. 그러나 별다른 특정 목적이 없다면 non-pagepool한 memory 영역을 할당 할것이다.
뒤이어 IoAllocateMdl()을 이용하여 buffer를 지시할(설명할) MDL을 할당할 것이며 추가적으로 I/O Manager의 look-aside list에 MDL의 fixed part 부분을 을 할당할 것 이다. 다음으로 MDL의 variable part위하여 MmBuildMdlForNonPagedPool() 함수를 호출 할 것 이다.

특정, 현재 context를 가지고 있는 process (Kenerl memory를 공유 하고 싶어하는 Application)의 주소로 mapping하기 위하여 MmMapLockedPagesSpecifyCache()(Win2K를 위해) 또는 MmMapLockedPage() (NT V4를 위해) 함수를 이용할 것이다.
MmMapLocedXXX() 함수는 반드시 buffer를 mapping 하기 원하는 context내에서 호출되어져야 만한다.

아래 예제를 Kernel mode memory address를 이용하여 user mode쪽으로 mapping하는 예제 입니다.

```
PVOID
CreateAndMapMemory()
{
    PVOID buffer;
    PMDL  mdl;
    PVOID userVAToReturn;

    //
    // Allocate a 4K buffer to share with the application
    //
    buffer = ExAllocatePoolWithTag(NonPagedPool,
                                   PAGE_SIZE,
                                   'ShareMem');
```

```
    if(!buffer)  {
        return(NULL);
    }

    //
    // Allocate and initalize an MDL that describes the buffer
    //
    mdl = IoAllocateMdl(buffer,
                        PAGE_SIZE,
                        FALSE,
                        FALSE,
                        NULL);

    if(!mdl)  {
        ExFreePool(buffer);
        return(NULL);
    }

    //
    // Finish building the MDL -- Fill in the "page portion"
    //
    MmBuildMdlForNonPagedPool(mdl);

#if NT_40

    //
    // Map the buffer into user space
    //
    // NOTE: This function bug checks if out of PTEs
    //
    userVAToReturn = MmMapLockedPages(mdl,
                                      UserMode);

#else

    //
```

```
    // The preferred V5 way to map the buffer into user space
    //
    userVAToReturn =
         MmMapLockedPagesSpecifyCache(mdl,          // MDL
                                      UserMode,     // Mode
                                      MmCached,     // Caching
                                      NULL,         // Address
                                      FALSE,        // Bugcheck?
                                      NormalPagePriority); // Priority

    //
    // If we get NULL back, the request didn't work.
    // I'm thinkin' that's better than a bug check anyday.
    //
    if(!userVAToReturn)  {

        IoFreeMdl(mdl);
        ExFreePool(buffer);
        return(NULL);
    }

#endif

    //
    // Store away both the mapped VA and the MDL address, so that
    // later we can call MmUnmapLockedPages(StoredPointer, StoredMdl)
    //
    StoredPointer = userVAToReturn;
    StoredMdl = mdl;

    DbgPrint("UserVA = 0x%0xWn", userVAToReturn);

    return(userVAToReturn);
}
```

이 방법에서 또한 손실을 안고 있다. MmMapLockedXXX() 함수는 만드시 mapping 하기 원하는 process context 내에서 호출이 되어져야 하기 때문이다. 우리가 처음에 얘기 했던 IOCTL 의 METHODE_NEITHER 방법보다 더 flexible 하지 못하다. 그러나 이 방법은 MmMapLockedXXX() 함수만이 해당 process context 내에서 호출이 되어지면 되고, top of stack 에서 만 사용 가능 했던 IOCTL 과는 틀리게 그보다더 아래 단에서두 사용이 가능하다. 많은 OEM Device 들의 Driver는 top of stack이 아니다.

이 방법을 사용 하면 반드시 page를 unmap 시켜야 한고 이 일은 IRP_MJ_CLOSE에서 보다는 IRP_MJ_CLEANUP에서 해야한다. 그 이유는 IRP_MJ_CLEANUP은 requesting thread 내에서 실행 되어진기 때문이다.

지금까지 우리는 kernel mode와 user mode간의 메모리 공유에 대한 두가지 방법을 살펴 보았다. 첫번째는 user mode에서 buffer를 할당하고 IOCTLs를 이용하여 kernel Mode로 내려 보내는 방법. 두번째는 Kernel mode에서 buffer를 할당하여 특정 process context(우리가 공유하려는 user-mode application)에서 MmMapLockedXXX()함수를 사용하는 방법이다.
비교적 두방법 모두 간단하며, 몇 가지의 rule만 지키면 된다.

본 column에 대한 문의 사항은 저희 홈페이지 QnA을 이용하시면 됩니다.