

제 PC의 경우에는 500MB 이상의 파일부터 MapViewOfFile 함수에서 에러를 리턴하였습니다. GetLastError()함수로 가져온 에러 코드는 0x8이었고 Windows MSDN에서 System Error Code 페이지에서 이 코드번호를 찾아본 결과

ERROR_NOT_ENOUGH_MEMORY

Not enough storage is available to process this command.

라는 에러 메시지임을 알아내었습니다. 결국 MapViewOfFile함수를 실행하기에 메모리가 부족하다는 것을 알 수 있었습니다.

맵뷰는 파일을 그대로 메모리에 맵핑시키는 것이므로 이러한 에러가 발생한 이유는 **메모리에 할당할 수 있는 최대사이즈를 넘어가는 파일을 넘어갔기 때문**이라고 생각됩니다.

그런데 제 PC의 램 메모리는 8GB인데 500MB에서부터 에러가 나는 것이 이상했고 더 검색을 해본 결과 Stackoverflow의 코멘트에서 그 해답을 찾을 수 있었습니다.

“So MapViewOfFile normally just chooses an address where it can fit the file view's bytes continuously in memory. - Saqlain Mar 23 '13 at 16:13” 라는 구절을 발견했고

저의 컴퓨터에서 사용가능한 램은 남아있을 수 있지만 **500MB 이상의 “continuously” 즉 연속된 메모리가 부족했었**기에 MapViewOfFile함수가 에러를 리턴했다고 추측할 수 있었습니다.

```
LPVOID WINAPI MapViewOfFile(  
    _In_ HANDLE hFileMappingObject,  
    _In_ DWORD dwDesiredAccess,  
    _In_ DWORD dwFileOffsetHigh,  
    _In_ DWORD dwFileOffsetLow,  
    _In_ SIZE_T dwNumberOfBytesToMap  
);
```

MapViewOfFile 함수는 이러한 문제를 해결하기 위해서 OffsetHigh 와 OffsetLow라는 인자를 통해서 파일의 오프셋에서부터 일정 길이를 Mapping 할 수 있도록 구현되어 있었습니다. Offset이 32bit 두 개로 나뉘어있는 이유는 단일 파일이 가질 수 있는 크기가 32bit로 표현가능한 범위를 넘을 수 있어 64bit로 표현하기 위해서입니다.

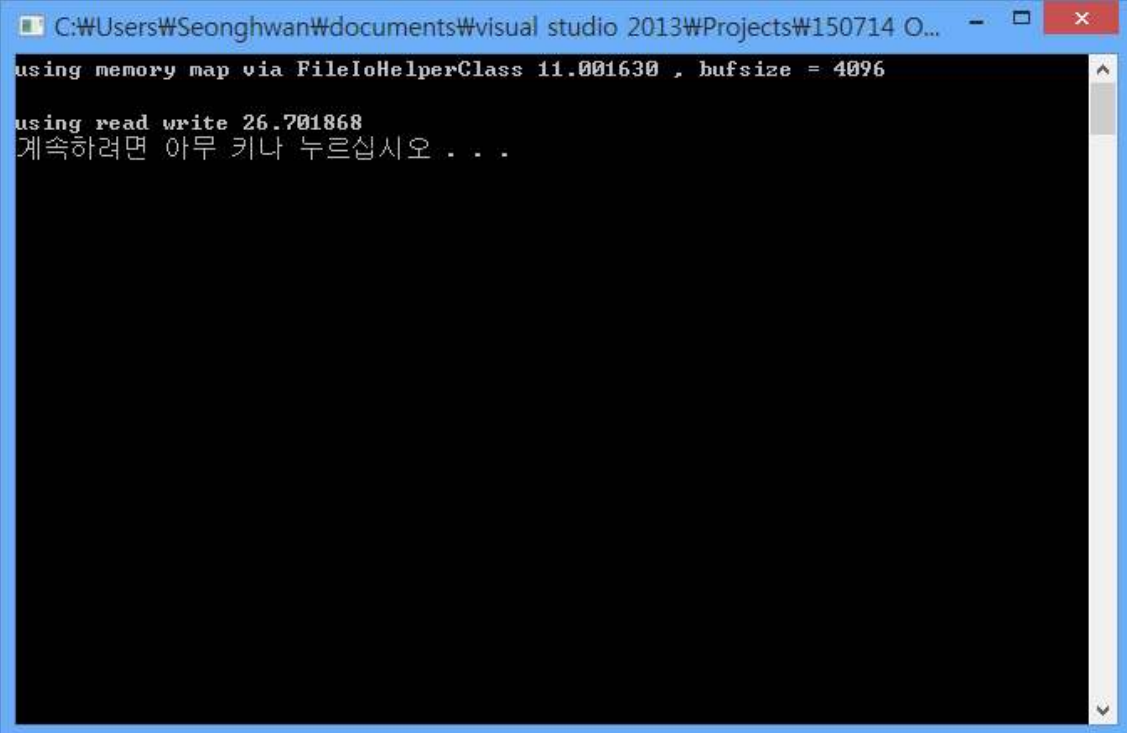
이러한 내용을 알게 된 후 멘토님의 Github에서 FileIoHelperClass.cpp 과 그 소스코드가 의존하고 있는 여러 소스코드들을 가져와 프로그래밍을 하였습니다. 직관적으로 구현되어 있어 쓰는데 큰 어려움은 없었지만 이해를 빠르게 하기 위해서 MapViewOfFile 함수를 검색해서 FIOReference 함수부터 차례차례 거꾸로 이해해 나간 뒤에 클래스를 사용할 수 있었습니다.

```

FileIoHelper Fi_read= FileIoHelper();
FileIoHelper Fi_write = FileIoHelper();
Fi_read.FIopenForRead(L"make.bin");
file_size_read.QuadPart = (1024 * 1024) * 4096LL;
Fi_write.FIcreateFile(L"copy.bin", file_size_read);
while (DT_SUCCEEDED(Fi_read.FIReadFromFile(file_offset, BUF_SIZE, buf))){
    Fi_write.FIWriteToFile(file_offset, BUF_SIZE, buf);
    file_offset.QuadPart += BUF_SIZE;
    //printf("%d", file_offset.QuadPart);
}
Fi_read.FIclose();
Fi_write.FIclose();
}

```

결국 구현한 코드는 위와 같습니다.



```

C:\Users\Seonghwan\documents\visual studio 2013\Projects\150714 O...
using memory map via FileIoHelperClass 11.001630 , bufsize = 4096

using read write 26.701868
계속하려면 아무 키나 누르십시오 . . .

```

MMIO를 통한 4G 파일의 복사는 11.0초가 걸렸으며, read_write를 사용한 복사는 26.7초가 걸렸습니다. 두 방식의 공정한 비교를 위해서 한번에 읽는 버퍼의 크기를 4096으로 동일하게 하여 수행하였습니다.