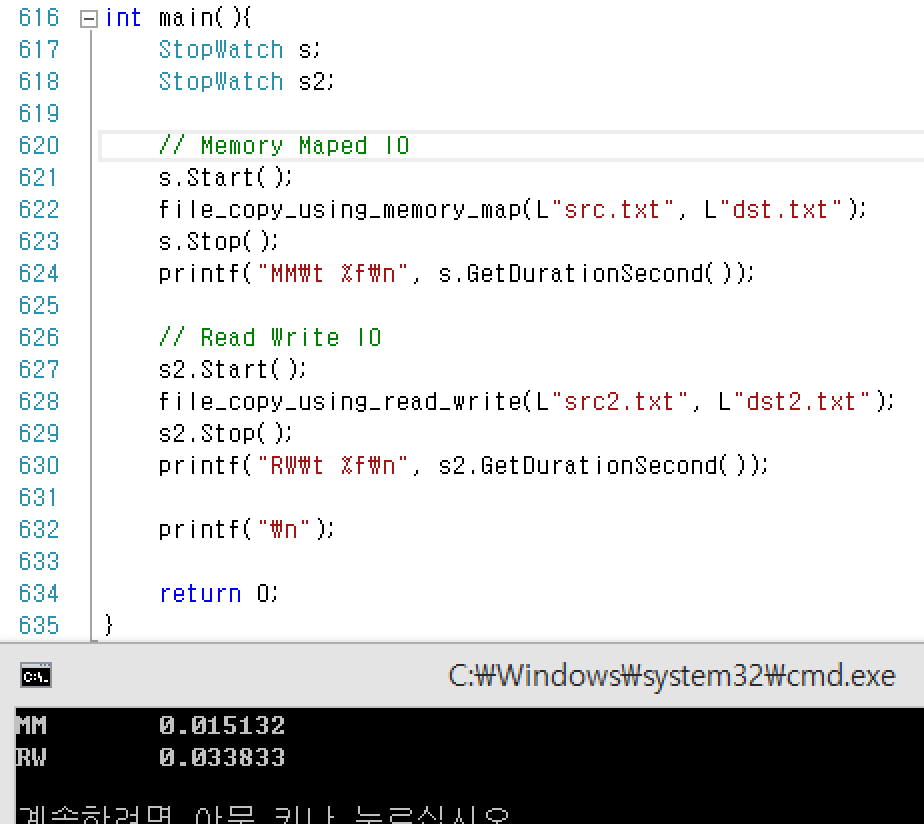
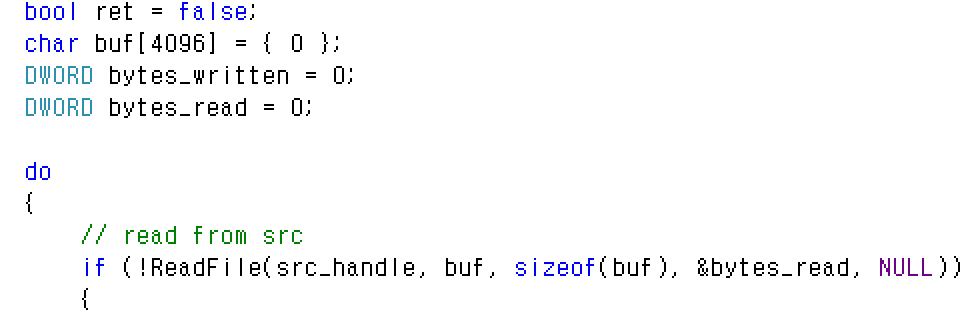
**운영체제 OS HW#2 보고서**

**BoB 4기 취약점분석트랙**

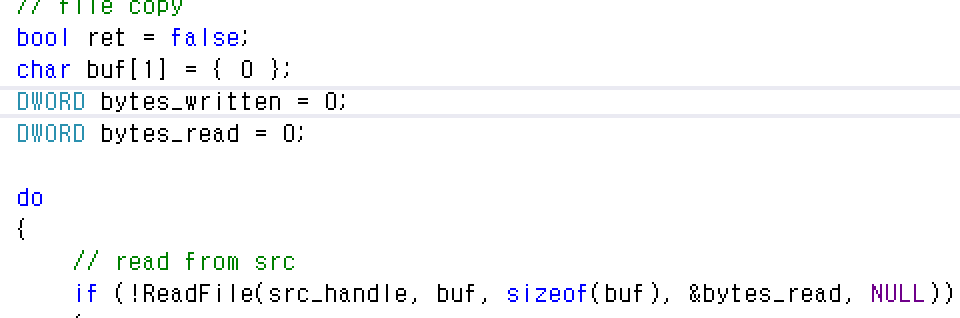
**정구범**

* [file\_copy\_using\_memory\_map] 과 [file\_copy\_using\_read\_write] 비교시간 출력

  
이론 상 memory maped IO가 더 빨라야 정상이나, 실제 결과로는 Read Write를 사용한 복사가 2배정도 빠른 결과를 보여주고 있다. 이는



read write함수내에서 4096씩 읽기 때문에 4096배 더 빠른 결과를 보이기 때문이다. 이를 원래의 의도한 결과인 공정한 결과를 내기 위해서는 buf의 크기를 단순히 1로 바꾸어 주기만 하면 된다.



그렇게 되면, memory maped io와 동일하게 읽게 되며, 속도 차이는



약 454배 빠른 결과를 보이게 되며, 원래 의도한 결과가 나오는 것을 확인 할 수 있다.

* 파일 크기를 5G 이상하였을때, 오류가 발생되는 원인과 해결 코드 작성

mapviewoffile은 4GB이상의 파일을 메모리에 맵핑하려고 할 때 에러를 발생한다.

achine generated alternative text:
c+users+goobeom+documents+visual studio 20

에러번호 8번을 내면서 에러가 나게 되는데, 이 번호를 오류조회 해보면, 사용 가능한 저장소 부족 에러이다. 따라서, 4gb이상의 주소를 매핑할 수 없기 때문에 에러가 나게 되는 것이다.(32bit운영체제는 메모리 주소의 크기가 4gb이므로, 그 이상되는 데이터를 매핑하는 것이 불가능 하다.) 따라서, 4GB이상의 파일들은 한번에 모든 파일을 메모리에 올리면 안되고, offset을 사용하여, 원하는 부분만을 선택하여 메모리에 올려야 한다.

즉, 전체를 매핑할 수 없다. 따라서, 일부분만 선택하여 올리게 되는 것이다. 제공해주신 소스는 전체를 모두 올리는 소스이므로 당연히 4GB이상을 메모리에 맵핑할 수 없으며, mapviewoffile api의 offset인자를 사용하는 방법으로 수정할 필요가 있다.

해결코드 작성을 위해서는 여러가지 방법이 있을 수 있지만, 여기서는 4GB이상의 파일을 매핑할 때에는 원하는 부분의 offset을 선택하도록 구현하였다.

이를 위해서 이전의 OpenFileContext 함수의 뒤에,

/Users/ninetiger/Library/Group Containers/UBF8T346G9.Office/msoclip1/01/837D8DB9-216C-764B-ACEB-AD0253FEFBC0.png

pOffset과 dwoffsetSize를 추가하였고, 이는 offset과 그 offset으로 부터의 크기를 입력받아서, 4gb중 어느 부분을 입력받을 지를 선택하는 것이다.

이렇게 수정된 MapViewOfFile 함수는 4gb보다 작은 파일을 넣게 된다면, 기본 인자값으로 0이 들어가 처리되므로, 기존과 같은 동작을 하게 되며, 인자값을 추가로 입력하였을 경우에만, 추가의 기능을 수행하게 된다.

achine generated alternative text:
MapViewDfFi 
ReadDnlY 
-OLowPart 
dwoffset

또한, MapViewOfFile을 추가 된 인자 값을 받도록 수정하면, 기능이 완성된다.