

# 운영체제 과제(2)

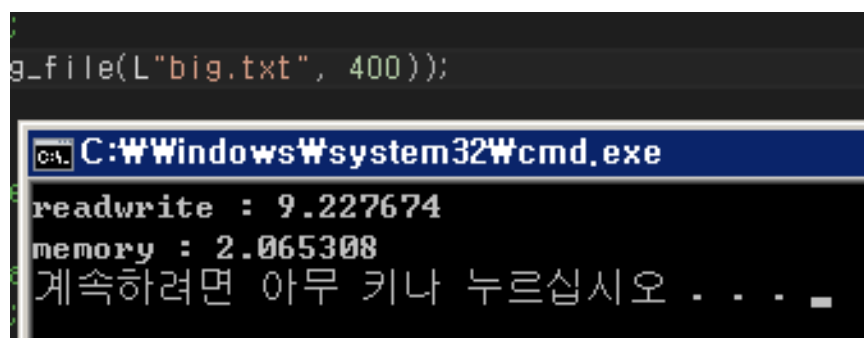
- 함수 시간 비교 -

- 5GB 이상일 때 -

트랙 :	취약점 분석
이름 :	유동균
제출날짜 :	07월 21일

## 1. file\_copy\_using\_memory\_map()과 file\_copy\_using\_read\_write() 시간 비교

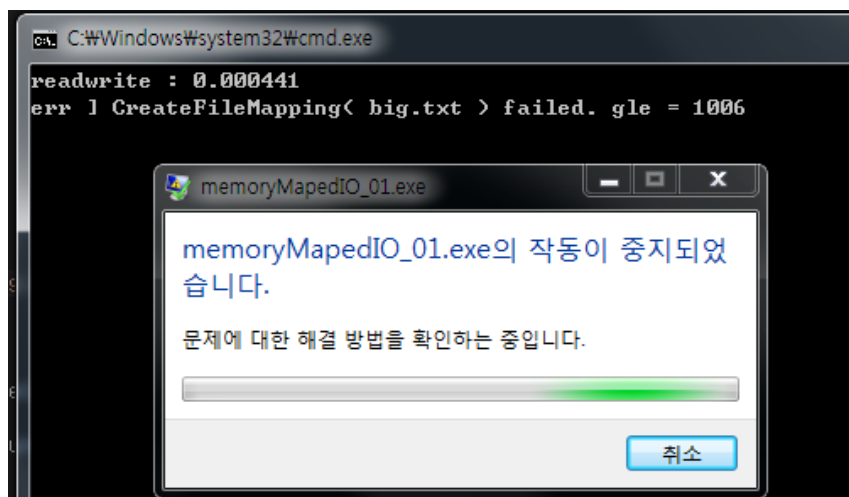
- SSD인 컴퓨터에서 테스트를 해보면 read\_write가 빠르게 나온다. 아마 저장장치의 성능 차이라고 생각이 된다.
- 일반적인 결과를 얻기 위해 HDD가 있는 컴퓨터에서 테스트를 해보면 read\_write는 9.2초, memory\_mapped\_IO는 2.0초가 나온 것을 확인 할 수 있다.



```
g_file(L"big.txt", 400));  
  
C:\Windows\system32\cmd.exe  
readwrite : 9.227674  
memory : 2.065308  
계속하려면 아무 키나 누르십시오 . . .
```

## 2. 5GB 이상일 때, 발생하는 오류의 원인과 해결 코드

- IA-32 기반 시스템에서 하나의 프로세스에서 PAE 기술을 사용하지 않고 사용 가능한 최대 크기는 4GB로 제한된다. 때문에 MemoryMapped IO를 이용해, 4GB 이상의 파일을 생성할 경우, 메모리를 정상적으로 참조하지 못하여 에러를 발생시킨다.



- MemoryMapped IO를 이용해 4GB 이상의 파일을 생성하려면 한 파일을 여러 번 나눠서 생성해 줘야 한다.
- 이것을 위해서 (노용환 멘토님이 참고 자료로 주신) FileIoHelperClass를 이용하여 파일을 제어해야 된다.

```
while (offset.QuadPart < Size.QuadPart) {
    if ((Size.QuadPart - offset.QuadPart) > (LONGLONG)bufSize) {
        bufSize = (unsigned long long)(1024 * 1024 * 128);
    }
    else {
        bufSize = (unsigned long long)(Size.QuadPart - offset.QuadPart);
    }
    Fhelper_read->FIOReadFromFile(offset, bufSize, buf);
    Fhelper_write->FIOWriteToFile(offset, bufSize, buf);

    offset.QuadPart += (LONGLONG)bufSize;
}
```

- 파일을 128MB 단위로 잘라서 파일을 생성한다. 여기서 파일이 생성된 크기를 판단하여 다음 내용은 그 다음에 써야 하므로 offset 변수를 통해 파일이 쓰여지고 있는 상황을 판단한다.
- 결과적으로 다음과 같이 복사된 것을 확인할 수 있다.

