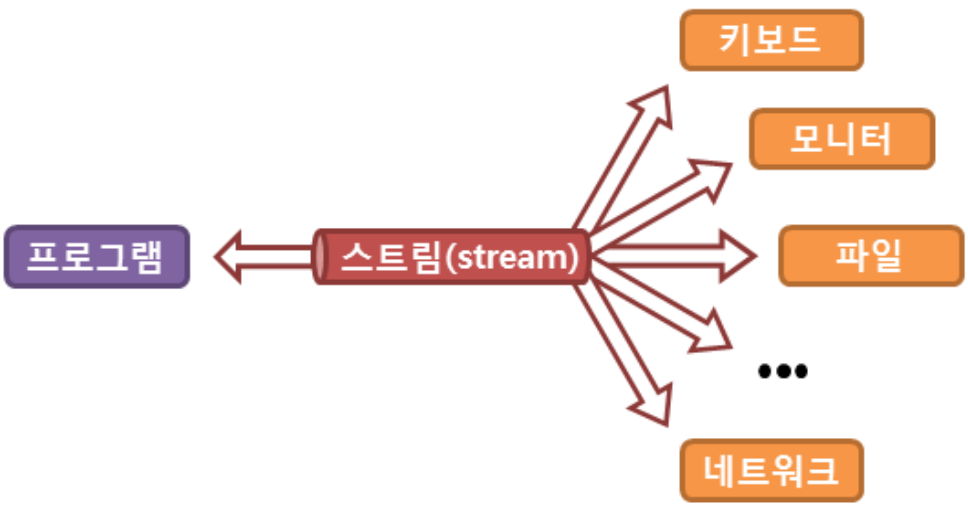
스트림(stream)

Byte 단위의 데이터 흐름.

C++ 프로그램은 파일이나 콘솔의 입출력을 직접 다루지 않고, 스트림(stream)이라는 흐름을 통해 다룹니다.
스트림(stream)이란 실제의 입력이나 출력이 표현된 데이터의 이상화된 흐름을 의미합니다.
즉, 스트림은 운영체제에 의해 생성되는 가상의 연결 고리를 의미하며, 중간 매개자 역할을 합니다.



X 버퍼가 사용되는 순간 : ① 입력 장치를 통해 데이터를 입력받고, 출력 장치를 통해 데이터를 출력할 때.

② 프로세스에서 다른 프로그램을 읽거나 쓸 때.

2) 버퍼링(Buffering)을 하는 이유는 무엇인가?

버퍼링이란 데이터를 전송 하는데 있어서 목적지로 바로 보내는 것이 아니라, **중간에 버퍼(여분의 임시 저장소)를 둬서 전송하고자 하는 데이터를 임시 저장해 두는 것을 말한다.** 임시 저장된 데이터가 어느 정도 채워지면 한꺼번에 데이터를 전송하게 된다.

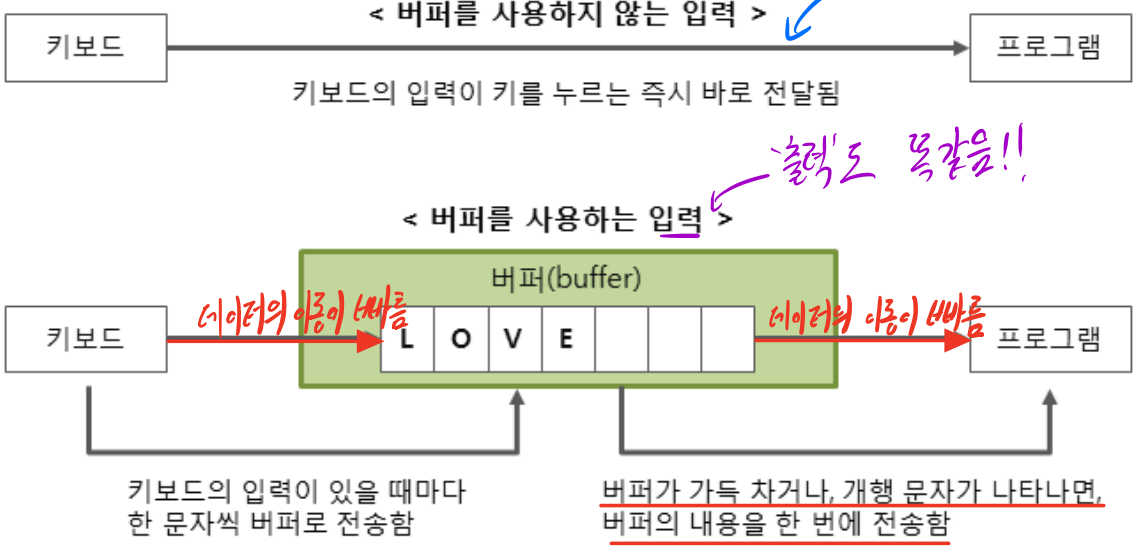
↳ 즉, 데이터가 chunk로 분해되어 전송된다.

버퍼링은 사용하는 이유는 ? 성능의 향상이 그 이유이다. 데이터를 키보드로부터 입력받거나, 데이터를 모니터로 출력하는 것은 여러분이 생각하는 것보다 **시간이 걸리는 작업들이다.**

그럼에도 불구하고 이러한 작업들(I/O 작업이라 한다)은 컴퓨터가 처리해야 하는 많은 일들과 비교해서 상대적으로 중요도가 떨어지는 작업들이다. 즉 지금 컴퓨터가 바쁘다면, 이러한 입 · 출력과 관련된 일들은 뒤로 미뤄도 된다는 것이다.

버퍼(buffer)

스트림은 내부에 버퍼(buffer)라는 임시 메모리 공간을 가지고 있습니다.
이러한 버퍼를 이용하면 입력과 출력을 좀 더 효율적으로 처리할 수 있게 됩니다.



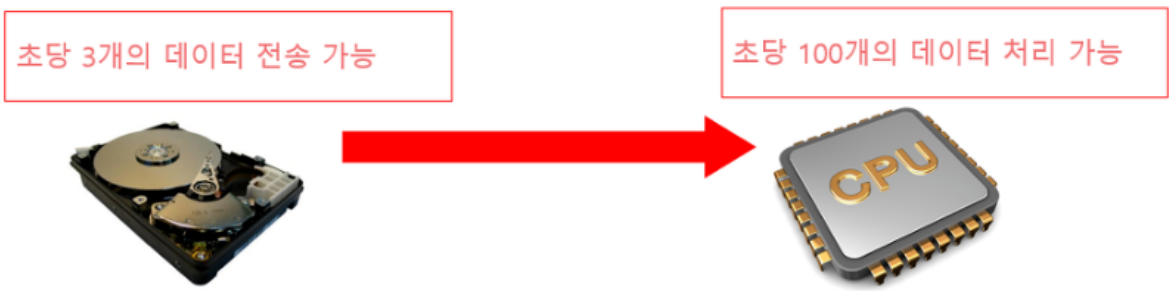
버퍼를 사용하면서 얻을 수 있는 장점은 다음과 같습니다.

1. 문자를 하나씩 전달하는 것이 아닌 묶어서 한 번에 전달하므로, 전송 시간이 적게 걸려 성능이 향상됩니다.
2. 사용자가 문자를 잘못 입력했을 경우 수정을 할 수가 있습니다.

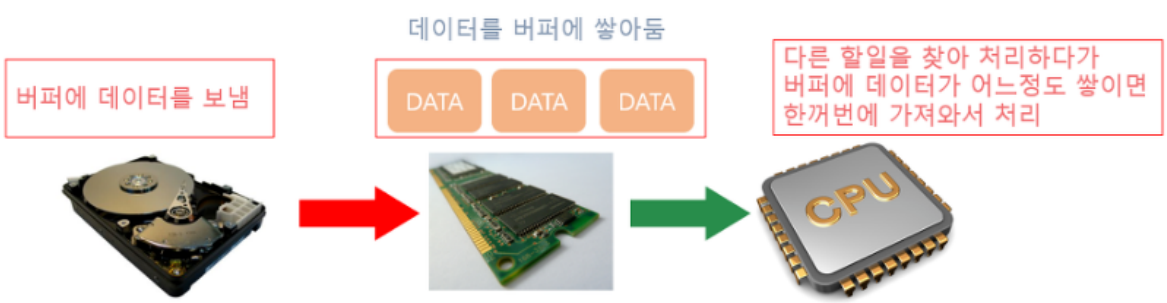
프로그래밍에서의 버퍼란 어떤것일까요?
프로그래밍이나 운영체제에서 사용하는 버퍼는 거의 대부분 **CPU**와 **보조기억장치** 사이에서 사용되는 임시 저장 공간을 의미합니다.

CPU는 기술이 발전함에 따라 1초에 수십억 bit 그 이상의 데이터를 처리할 수 있습니다.
(수학에 약해 정확한 계산은 하지 않지만 CPU가 크게 나누었을 때 컴퓨터에서 속도가 가장 빠른 장치입니다.)

그러나 보조기억 장치의 경우 데이터를 주고 받는데에 많은 시간이 필요합니다.
물리적인 HDD(하드디스크)의 경우 데이터를 전송할 수 있는 속도는 빨라도 초당 100mb ~ 300mb 입니다.



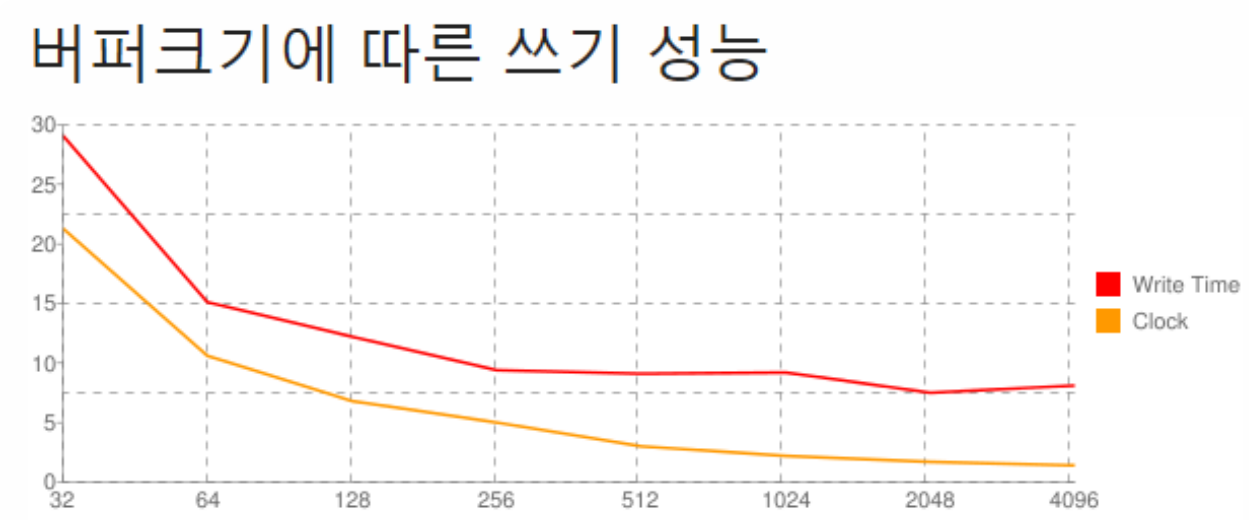
간단하게 얘기 하자면 CPU는 1초에 100개의 데이터를 처리할 수 있지만 정작 처리할 데이터를 가지고 있는 보조 기억 장치는 데이터를 1초에 세 개밖에 보내주지 않는 것입니다. CPU입장에서는 아무리 일을 열심히 하고 싶어도 데이터를 보내주지 않기 때문에 능력에 비해 97 개 만큼 효율성을 잃게 됩니다.

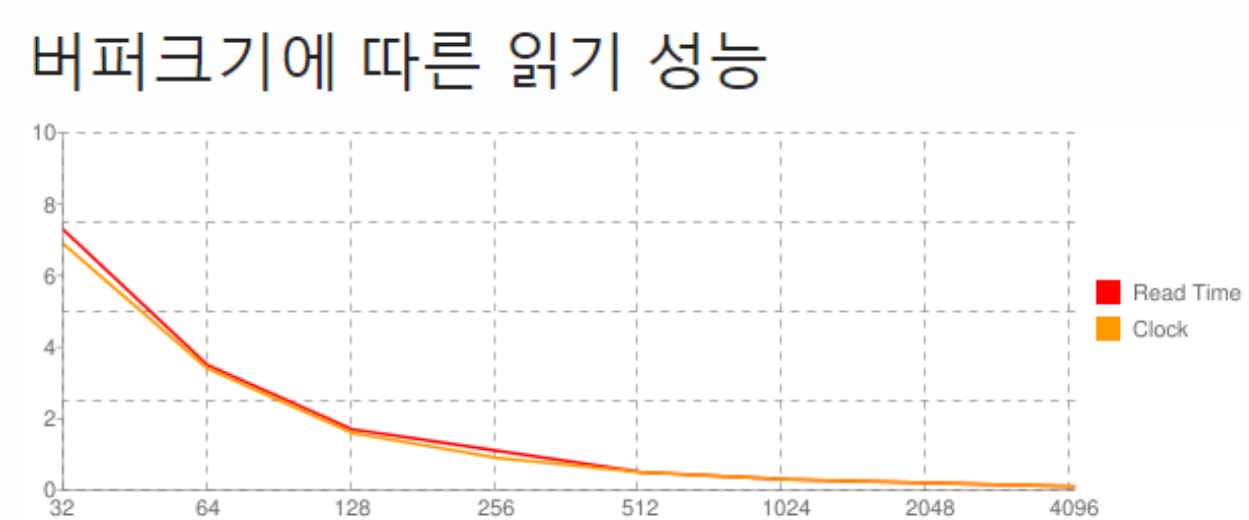


이 때 버퍼를 사용하게 됩니다. 버퍼는 CPU 내부에 있는 캐시메모리 보다는 느리지만 보조 기억 장치보다 훨씬 빠른 주기억 장치(RAM)를 이용합니다. ① 보조기억장치는 주기억장치의 버퍼로 마련해둔 공간에 데이터를 열심히 보내 쌓아 둡니다.
② CPU는 처리가 빠르므로 밀려있는 다른일을 처리한 후 시간이 남을때 가끔 버퍼를 확인하여 데이터가 모두 쌓였는지 확인하고 모두 쌓였다면 가져다 한꺼번에 처리합니다.
그 덕분에 CPU는 100퍼센트의 효율로 연산을 할 수 있습니다.
이렇게 버퍼라는 것은 속도차가 큰 두 대상이 입출력을 수행할 때 효율성을 위해 사용하는 임시 저장공간이라고 할 수 있겠습니다.

파일을 읽고 쓸때, 한번에 읽어들이는 데이터의 크기에 따라 읽기/쓰기 성능이 달라진다는 것은 상식선에서 알고 있을 것이다. 대략 알고 있는 바로는 1024 바이트 단위로 읽어올 때 가장 효과적인 것으로 알고 있다. 실제, 이러한 우리의 상식이 올바른지를 확인하기 위해서 버퍼 크기에 따른 읽기/쓰기 성능에 대한 자료를 만들어보기로 했다.

다음은 테스트를 위해서 만든 간단한 프로그램(:12)이다. 32 부터 4096 까지 버퍼사이즈를 2배씩= 증가시키면서, 동일한 데이터를 읽고 쓰는데 걸리는 시간을 체크했다. 파일의 크기는 256M로 했다.





결과

결과는 예상했던바와 크게 다르지 않다. 단 쓰기에 있어서, 버퍼의 크기가 256를 넘어가게 되면, 쓰기시간에 있어서 성능의 개선이 기대되지 않는 반면, Clock 시간은 꾸준히 감소함을 알 수 있다. 그렇다면, Clock 시간을 감안해서 1024 byte 정도를 선택하는게 가장 무난할듯 싶다.

읽기의 경우도 1024 byte(:12)정도로 버퍼크기를 잡는게 가장무난할듯 하다. 파일의 크기가 크다면 1024 byte이상을 잡아도 어느정도의 성능향상은 기대할 수 있을 것 같다.

이상 버퍼크기에 따른 읽기/쓰기 성능측정을 해보았는데, 이것은 DB와 같이, 랜덤 access가 일어나는 어플리케이션을 위해서는 쓸만한 정보를 주기 힘들 것이다. 시간이 된다면 랜덤 access의 성능측정을 해보는 것도 재미있을 것 같다. 이외에 seek(:12)시간등에 대한 성능측정을 해보는 것도 괜찮을것 같다.