fopen이 실패햇을 때 왜 실패했는지 알길이 없다. 그래서 에러넘버라는 것을 제공해주는데 어떤 원인으로 오류가 발생했는지 알려주는 정수값인데, 그것을 통해 왜 죽엇는지 알수있다.

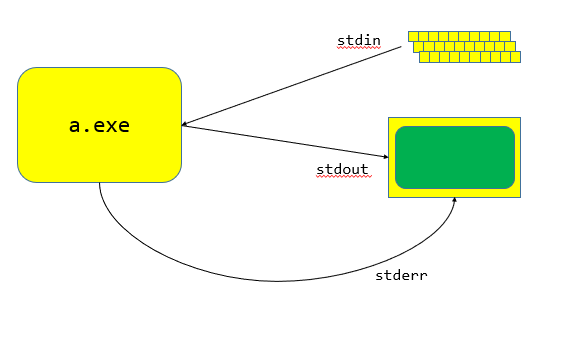
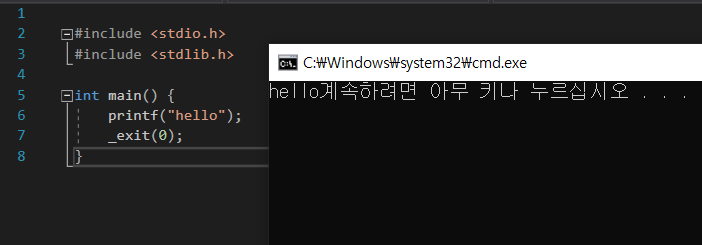
표준 라이브러리는 함수 호출 실패시, 이를 식별할수 있도록 에러코드를 제공한다. 이를 사용하기 위해서는 errno.h 파일을 포함시켜야한다.

표준 라이브러리는 에러코드를 문자열로 함수또한 제공해준다.

위와 같은 함수를 계속해서 쓰는 것은 비효율적이므로 함수를 만들어 쓰자!

* 그러나 이미 표준 라이브러리에서 제공되는 함수가 존재하는데 그 이름은 perror이다!

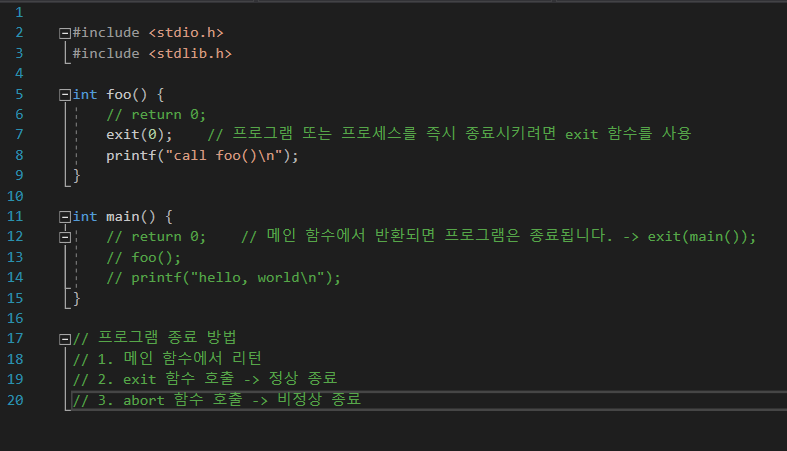
그냥 이거 쓰자!!



stdout / stderr 둘다 모니터에 출력이지만 딱 하나의 차이는 버퍼의 유무 차이이다.

printf는 라인버퍼라 개행을 만나면 buffer에 쌓여있는 데이터가 전송되는데 만약 printf(“hello”); 같은것을 실행하다가 만약 비정상적으로 종료되면 그 printf의 내용은 사라진다.

stderr를 통해 전송을 하면 바로바로 모니터로 출력되므로 비정상종료된다 하더라도 전송은 정상적으로 이뤄진다. fprintf(stderr,…);

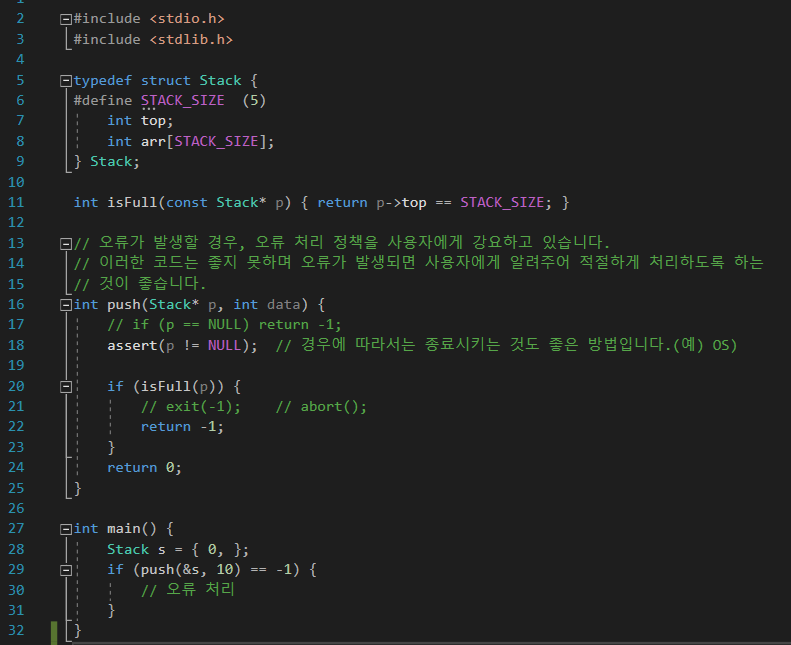
함수가 호출된 시점에 해당 프로세스를 중지시키고 싶으면 return이 아닌 exit를 사용하면 된다.

->프로그램 또는 프로세스를 즉시 종료시키기 위해선 exit()함수 사용!

🡪프로그램 종료방법

1. 메인 함수에서 리턴 (= exit(main); )
2. exit()함수 호출 -> 정상종료
3. abort()함수 호출 -> 비정상종료

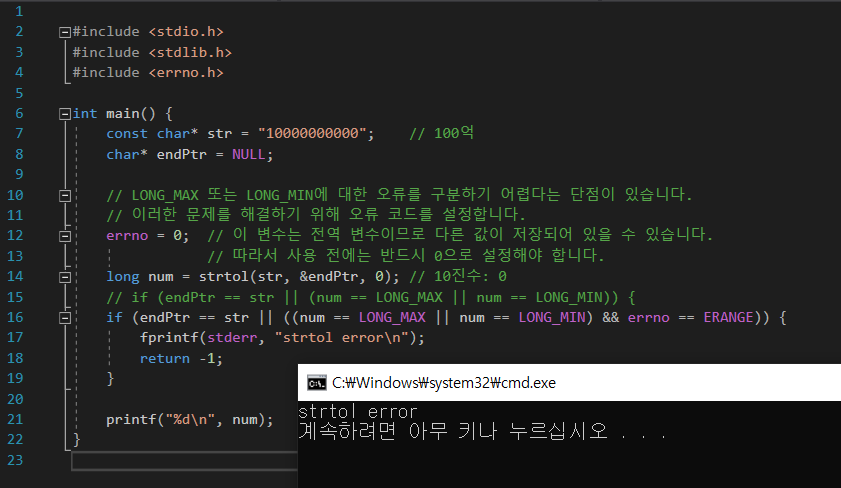
cf. assert함수는 abort 함수로 처리하여 종료하는 식이다.

오류가 발생할 경우, 오류 처리 정책을 사용자에게 강요하고 있다.

이러한 코드는 좋지 못하며 오류가 발생되면 사용자에게 알려주어 적절하게 처리하도록 하는 것이 좋다.

스택이 가득찬 것은 가볍지만 스택이 NULL인 것은 결코 가볍지 않다.. 그러므로 무조건 모든 오류를 사용자에게 전가해도 상관없지만, 사용자의 생명과 직결되는 예외들은 무조건 return -1;로 사용자에게 전가하는것보다도 경우에 따라서 프로그램을 종료해야 하는 경우에는 assert를 쓰는것이 좋다.

->경우에 따라서는 종료시키는것도 좋은 방법이다. (ex. OS)

LONG\_MAX 또는 LONG\_MIN에 대한 오류를 구분하기 어렵다는 단점이 있다.

이러한 문제를 해결하기 위해 오류 코드를 설정한다. 그 오류 코드 errno는 전역변수로 설정하는데 다른값이 저장되 있을수도있다. 따라서 사용전에는 반드시 0으로 설정해야한다.