과제 4: Buffer-based Block Mapping FTL 구현

1. 개요

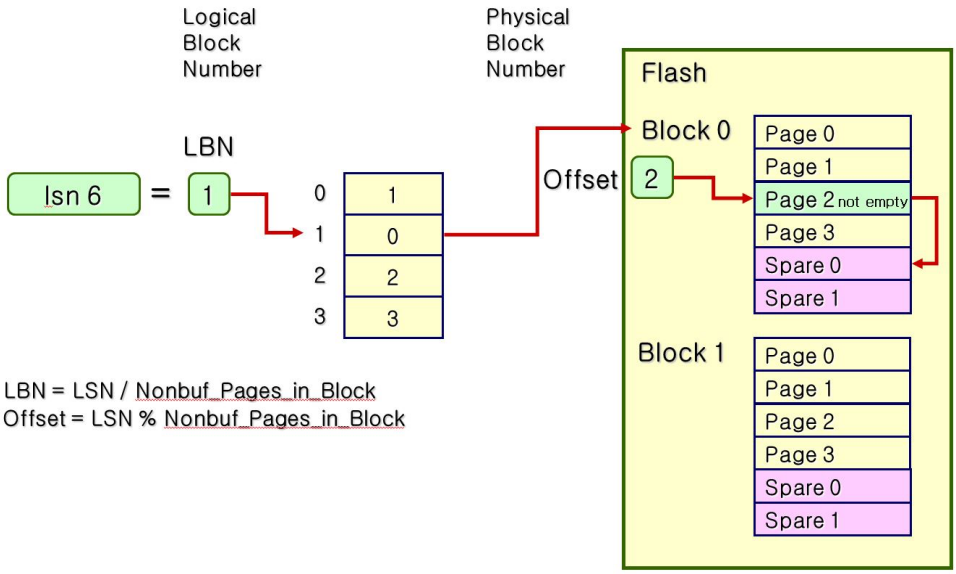
과제 3에서 구현한 Block mapping FTL을 bubfer의 개념을 추가한 Buffer-based Block Mapping FTL을 구현하시오.

네 개의 소스 파일이 주어지며 이것을 이용하여 상기의 FTL을 구현하시오.

(1) blkmap.h : flash memory를 구성하는 block, page, sector, spare area 등의 상수 변수가 정의되어 있으며, 반드시 이 상수 변수를 이용하여 프로그래밍해야 함
(2) devicedriver.c : flash memory에 page 단위로 데이터를 읽고 쓰기 위한 dd_read()와 dd_write() 함수와, 또한 block을 소거하는 dd_erase() 함수가 정의되어 있음
(3) ftlmgr.c : 상기의 FTL 기법을 따르는 ftl_open(), ftl_write(), ftl_read()를 구현해야 됨
(4) main.c : ftl_write())와 ftl_read() 테스트를 위한 것으로 필요 시 활용할 수 있음

** Makefile은 필요 시 수정해서 사용하기 바라며 그렇지 않은 경우 각 C파일을 컴파일해서 object 파일을 만들고 linking을 하기 바람 (제발 C파일이나 헤더파일에 또다른 C파일을 include하지 마세요)

2. Buffer-based Block Mapping FTL



플래시 메모리의 각 블록은 non-buffer page들과 buffer page들로 나뉘며, 전자는 과제 3의 block mapping 기법처럼 사용되며, 후자는 전자에서 overwrite가 발생하면 그것을 처리하는 buffer의 개념으로 사용된다. 과제 3의 block mapping과 다른 점은 각 블록마다 buffer 개념을 구현하기 위한 별도의 공간(page)을 두고 있다는 것이다. 수업 시간에 배운 block mapping과 hybrid mapping의 조합이라고 생각하면 이해가 쉽다.

2. 주의 사항

- blkmap.h와 devicedriver.c 파일에서 주의사항을 반드시 숙지하여 지켜야 하며, 그렇지 않을 경우 채점 시 컴파일 에러가 발생하고 불이익을 받을 수 있음
- 입출력 포맷은 별도로 존재하지 않으며, ftlmgr.c만 완벽하게 구현하면 됨

3. 개발 환경

- OS : Linux 우분투 버전 16.04
- 컴파일러 : gcc 5.04
- 반드시 Linux 우분투와 gcc 환경을 준수해야 하며, 이를 따르지 않아서 발생하는 불이익은 학생이 책임져야 함!!

4. 제출물

- ftlmgr.c를 하위폴더 없이(최상위 위치에) zip파일로 압축하여 myclass.ssu.ac.kr 과제 게시판에 제출 (소스파일, 헤더파일, zip파일은 반드시 소문자로 작성)
- 압축한 파일은 반드시 학번_4.zip (예시 20061084_4.zip)과 같이 작성하며, 여기서 4은 세 번째 과제임을 의미함
- 채점 프로그램상 오류가 날 수 있으니 꼭 위 사항을 준수하기 바람!

과제 4부터는 과제 명세서를 제대로 지키지 않아서 발생하는 불이익은 더 이상 정상 참작하지 않으며, 그 이유는 과제 명세서를 제대로 지키는 것도 실력이라고 판단하기 때문입니다.