과제 4: Flash Memory에서의 Sector Mapping FTL 구현

**주의: 프로그램 소스를 그대로 복사를 하거나 살짝 고쳐서 제출하는 경우 관련 학생들 모두 그 과제에 대해 0점 처리합니다. 두 번 이상 적발되는 경우 전체 과제 점수가 0점이며 D 이하의 학점이 부여됩니다.

1. 개요

Sector mapping 기법(강의자료 "Flash Memory Overview"의 19쪽)을 따르는 FTL을 구현한다. 다음과 같은 제약 사항을 따라야 한다.

- 파일 I/O 연산은 system call 또는 C 라이브러리만을 사용한다.
- 아래의 (1), (2), (3), (4)의 기능을 ftlmgr.c에 구현한다.
- sectormap.h와 fdevicedriver.c는 주어진 그대로 사용하며 수정해서는 안된다.
- 네 개의 함수를 테스트하기 위해 main() 함수를 본인 스스로 만들기 바라며, file system의 역할을 수행하는 main() 함수에서의 대략적인 시나리오는 (1) flash memory file 생성 및 초기화(과제3에서 수행) (2) ftl_open() 호출 (3) ftl_write()나 ftl_read() 호출하여 테스트하는 것으로 이루어진다. 테스트 프로그램을 실행하기 전에 이전에 사용했던 flash memory file은 삭제해야 한다.

(1) ftl_open() 구현

일반적으로 FTL을 구현하기 위해서는 논리주소를 물리주소를 변환하는 address mapping table이 필요하다. Sector mapping FTL에 맞는 address mapping table을 생각해 보고 이에 맞는 data structure를 정의해서 사용한다 (강의자료 참조). 이러한 data structure를 이용하여 address mapping table을 하나 생성한다 (즉, 변수 선언).

ftl_open()에서는 일반적으로 여러 초기화 작업을 하는데, 예를 들면, 생성한 address mapping table에서 lpn에 매핑되는 ppn의 값을 -1로 초기화를 한다. ftl_open()이 호출되는 시점은 flash memory를 최초로 사용하는 때이다. 따라서 file system의 어떤 lpn도 사용되지 않았으며 당연히 lpn에 매핑되는 ppn도 존재하지 않는다. 그런 의미에서 -1로 설정하는 것이다.

그 이외에 초기화할 작업이 필요한 경우가 존재한다면 이 함수에서 수행한다.

**file system에서 ftl_write()나 ftl_read()를 최초로 호출하기 전에 반드시 ftl_open()를 호출해야 한다.

(2) ftl_write(int lsn, char *sectorbuf) 구현

File system이 ftl_write()를 호출할 때 인자값으로 lsn(=lpn)과 sectorbuf에 저장되어

있는 512B 데이터를 전달한다. FTL은 이 데이터를 flash memory에 쓰기를 해야 하는데, 이때 어떤 물리적 페이지(ppn)에 써야할 지를 결정해야 한다. 이것은 sector mapping 기법의 동작 원리대로 결정되어야 한다. 예를 들면, file system이 ftl_write(lsn=5, sectorbuf)를 호출하면 sector mapping FTL은 address mapping table의 lpn=5를 찾고 이것에 매핑되는 ppn값을 확인한다. 만약 ppn=-1이라면 file system은 lpn=5 위치에 최초로 쓰기 작업을 수행한다는 것을 의미한다. 그렇지 않는 경우라면 file system은 lpn=5 위치에 이미 존재하는 데이터를 갱신(update)를 하기를 원한다는 것을 의미한다. 두 경우 모두, sector mapping FTL은 비어있는 ppn 중에서 하나를 선택해서 그 값으로 address mapping table의 해당 ppn을 갱신한다. 즉, lpn=5에 매핑되는 ppn 값으로 저장한다. File system이 이 함수를 호출할 때 lpn 위치에 최초 쓰기를 하는지 아니면 덮어 쓰기(overwrite)를 하는지 체크하는 과정이 필요하다.

비어있는 ppn을 할당한 후, FTL은 ppn이 가리키는 flash memory의 페이지에 쓰기 연산을 수행한다. 이때 flash device driver의 dd_write() 함수를 호출한다(과제 3 참고). FTL이 쓰기 연산을 할 때는 sector 단위가 아니라 page 단위임을 주의해야 한다. Flash device driver의 dd_write(int ppn, char *pagebuf)를 호출하기 전에 pagebuf에 file system으로부터 받은 sector 데이터와 lsn(lpn)을 각각 sector 영역과 spare 영역에 저장하고 이 pagebuf를 인자값으로 전달한다. 여기서 lpn을 spare에 저장하는 이유는 ppn이 lpn에 매핑되어 있다는 것을 표현하기 위한 것이다.

File system이 계속 쓰기 요청을 하면 결국 비어있는 ppn이 남아 있지 않는 상황이 발생할 수 있다. 이런 경우 flash memory에서 garbage 블록을 선정해서 빈 ppn을 만들어 낼 수 있다. 이때 예비 블록으로 남겨 둔 free block을 활용한다. garbage 블록의 선정을 위해 각자 본인의 알고리즘을 생각해 보기 바라며, 또한 garbage 블록의 빠른 선정을 위해 garbage 블록을 어떻게 관리하는 게 좋을지도 생각해 보기 바란다 (예를 들면, linked list). 참고로 garbage block은 새로운 free block이 되기 때문에 free block의 pbn은 바뀔 수 있다.

** dd_write()를 호출하기 전에 pagebuf에 sector 데이터와 spare 데이터를 저장할 때 memcpy()를 쓰면 편리하며 물론 다른 방식을 사용해도 됨

(3) ftl_read(int lsn, char *sectorbuf) 구현

File system이 ftl_read()를 호출할 때 인자값으로 lsn(=lpn)과 flash memorydml 512B 데이터를 저장할 sectorbuf를 인자값으로 전달한다. FTL은 address mapping table에서 주어진 lsn에 매핑되어 있는 ppn값을 구하고, 이것이 가리키는 flash memory의 페이지를 읽는다. 이때 flash device driver의 dd_read() 함수를 호출한다. 그리고 읽어 온 페이지에서 sector 영역에 저장되어 있는 데이터를 file system이 인자로 제공한 sectorbuf에 저장하여 전달한다.

** dd_read()를 통해 flash memory에서 페이지를 읽어 온 후 file system에 그 페이지

의 sector 데이터를 전달하기 전에 sectorbuf에 저장해야 하는데 이때도 memcpy()를 사용하면 편리함

(4) ftl_print() 구현

일반적으로 FTL이 제공해야 할 함수는 ftl_open(), ftl_write(), ftl_read() 세 개뿐이며, 이 함수는 단지 FTL의 address mapping table을 확인하기 위한 용도로 사용하기 위한 것이다. 이 함수는 화면에 lpn, ppn, free block의 pbn을 다음과 같이 출력해야 한다. (flash memory의 가용 블록 수가 3, 블록당 페이지 수가 4이고, 이 함수를 호출하는 시점에서 free block의 pbn=3라고 가정한다)

```
lpn ppn
0   -1
1   -1
2   -1
3   -1
4   2
5   3
6   4
7   5
8   0
9   1
10  -1
11  -1
free block's pbn=3
```

2. 개발 환경

- OS: Linux 우분투 버전 18.0.4
- 컴파일러: gcc 7.5

** 반드시 이 환경을 준수해야 하며, 이를 따르지 않아서 발생하는 불이익은 본인이 책임져야 함

3. 제출물

- 프로그래밍한 소스파일 ftlmgr.c를 하위폴더 없이(최상위 위치에) zip파일로 압축하여 myclass.ssu.ac.kr 과제 게시판에 제출한다 (모든 제출 파일들의 파일명은 반드시 소문자로 작성). 제공된 sectormap.h와 fdevicedriver.c는 제출할 필요가 없음.
- 압축한 파일은 반드시 학번_4.zip (예시 20061084_4.zip)과 같이 작성하며, 여기서 4는 네 번째 과제임을 의미함

** 채점 프로그램상 오류가 날 수 있으니 꼭 위 사항을 준수하기 바라며, 이를 따르지 않아서 발행하는 불이익은 본인이 책임져야 함