**Pintos Project 5: Filesystem**

담당 교수 : 박성용 교수님

학번 / 이름 : 20181662 / 이건영

개발 기간 : 2022.12.21~2022.12.26

1. **개발 목표**

Pintos의 File system을 수정하여 extensible file, subdirectory, buffer cache의 기능을 구현한다.

1. **개발 범위 및 내용**
   1. **개발 범위**
2. Extensible file & file growth

파일에 block을 할당하는 방식을 수정하여 파일의 크기가 늘어날 수 있도록 한다.

1. Subdirectory

기존의 root 디렉토리만 존재하던 pintos의 구조에서 계층 구조의 디렉토리를 사용할 수 있도록 구현한다.

1. Buffer cache

Buffer cache를 이용하여 디스크 입출력 시 걸리는 시간을 줄인다.

* 1. **개발 내용**
* Extensible file & file growth
  + Index structure와 management

direct, single indirect, double indirect 방식을 사용하여 index structure를 구성하였다. 이를 관리하기 위해 map\_table을 사용하였다.

* Subdirectory
  + Directory entry 관리 방법
* Buffer cache
  + Buffer cache eviction 방식

Clock 알고리즘을 사용하였다. clock pointer가 가르키는 cache의 reference bit이 false라면 해당 slot을 victim으로 선정하였다.

* + Buffer cache flush 방식

Cache의 valid bit가 true이며 dirty bit이 true라면 해당 내용을 디스크에 write하도록 구현하였다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

2022.12.21~2022.12.22 : Buffer Cache 구현

2022.12.22~2022.12.25 : Extensible file 구현

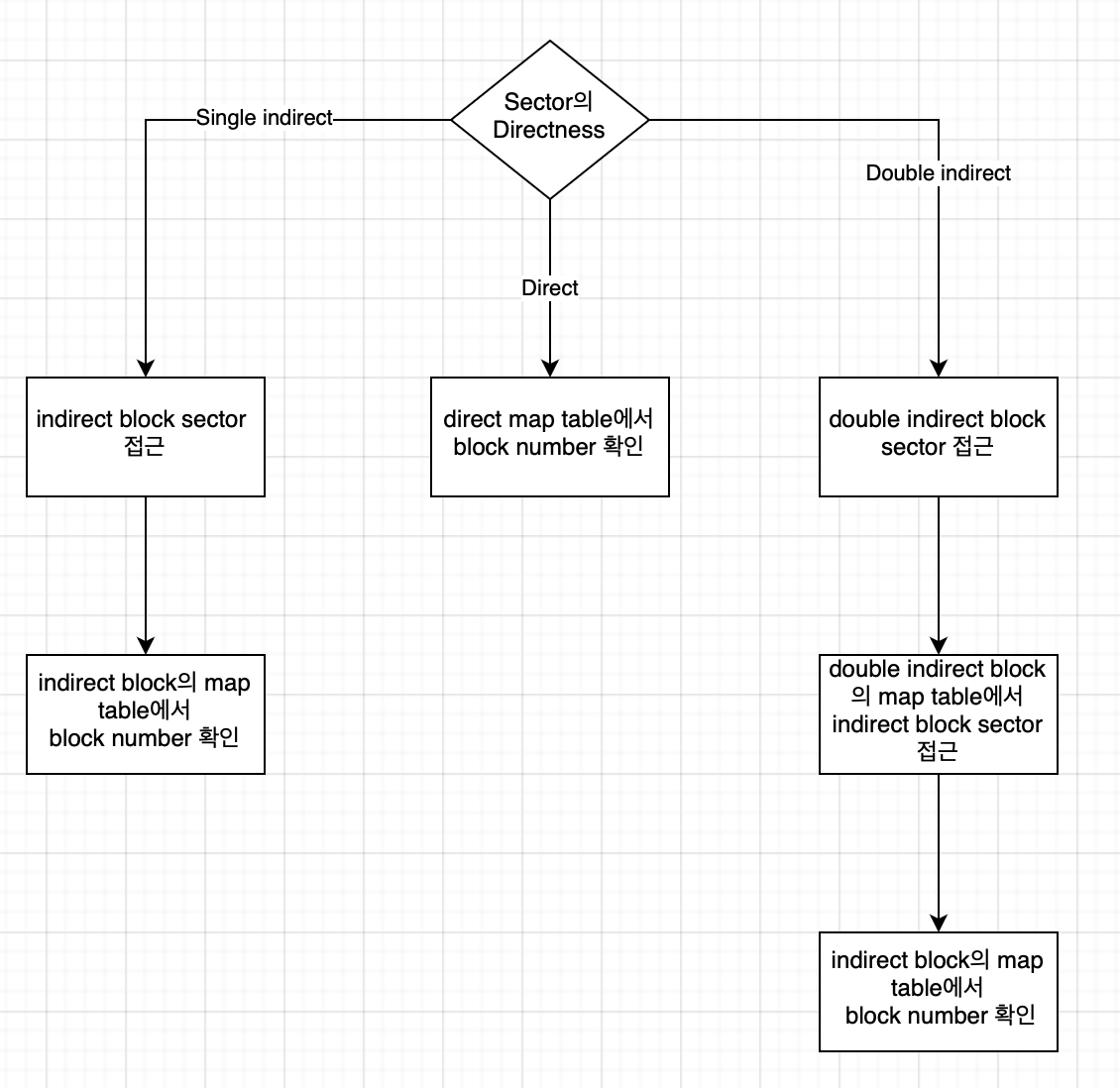
* 1. **개발 방법**

Buffer cache의 구현을 위해 새 파일인 cache.c와 cache.h를 만들고 그 안에 buffer cache entry 구조체와 관련 함수를 구현한다.

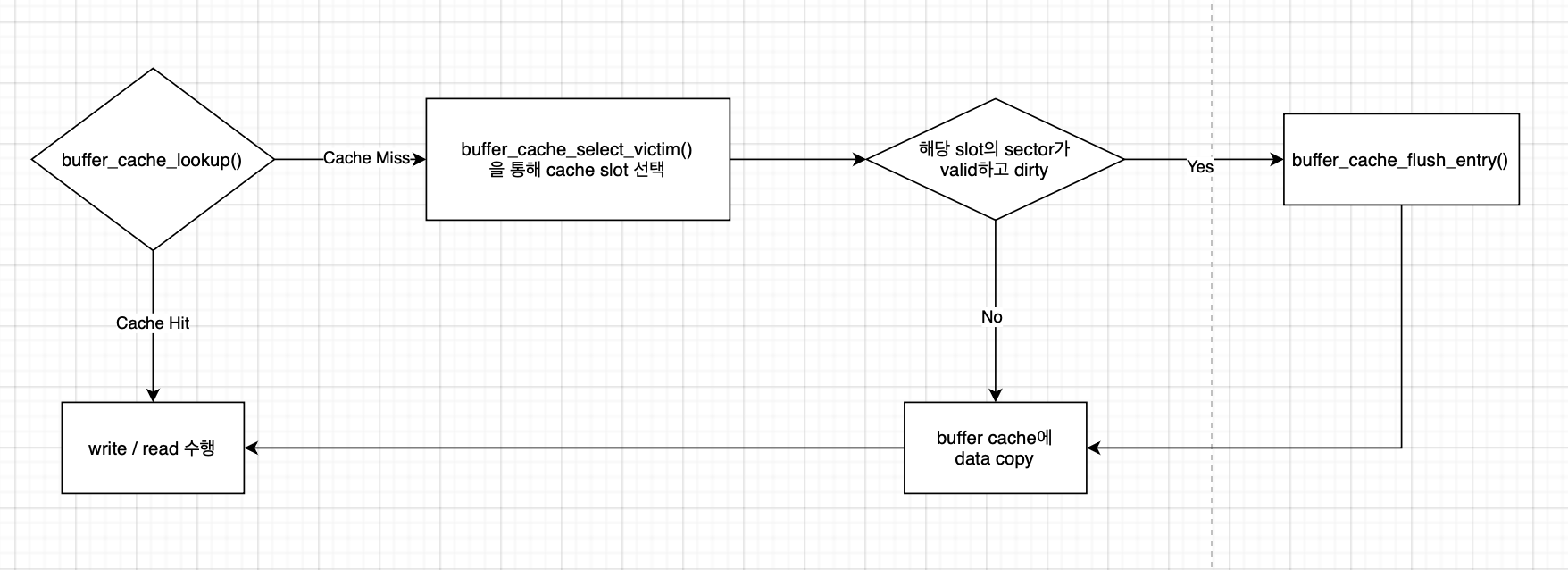
Extensible file을 위해 inode.c의 내용을 수정한다. 블록 할당 방식을 기존의 연속적인 할당 방식에서 direct, indirect 방식으로 수정한 후 이를 위한 함수를 구현한다.

1. **연구 결과**
   1. **Flow Chart**

* Extensible file & file growth



* Buffer cache



* 1. **제작 내용**
* Extensible file & file growth

enum direct\_t : 인덱스에 접근하는 방식을 정의하였다.

struct sector\_location : 해당 sector에 접근하는 방식과 이에 따른 인덱스를 담고 있다.

struct inode\_indirect\_block : indirect 방식을 위한 map table이다.

struct inode\_disk : direct와 indirect를 위한 table을 추가하였다.

locate\_byte() : file의 offset을 통해 block의 offset을 구하는 함수이다.

register\_sector() : 할당한 디스크 블록의 주소를 inode에 등록하는 함수이다.

inode\_update\_file\_length() : file의 크기를 수정하는 함수이다.

free\_inode\_sectors() : 파일에 할당되었던 디스크 블록들을 해제하는 함수이다.

byte\_to\_sector() : offset을 이용하여 디스크 블록 정보를 구하는 함수이다.

* Buffer cache

구조체 buffer\_cache\_entry를 선언하였다. valid\_bit, reference bit, dirty bit를 통해 해당 데이터가 유효한 지, 최근에 접근했는지(eviction을 위함), 수정이 되었는지(flush를 위함)를 확인한다.

buffer\_cache\_init() : buffer cache를 초기화한다.

buffer\_cache\_terminate() : file system을 종료하기 전, buffer cache의 내용을 flush한다.

buffer\_cache\_read() : lookup을 통해 cache에 해당 sector가 있는지 확인한다. 없다면 evict를 통해 sector를 cache에 올린다. 이후 buffer에 read를 진행한다.

buffer\_cache\_write() : lookup을 통해 cache에 해당 sector가 있는지 확인한다. 없다면 evict를 통해 sector를 cache에 올린다. 이후 buffer에 write를 진행한다.

buffer\_cache\_lookup() : sector가 cache에 있는지 확인한다.

buffer\_cache\_select\_victim() : clock 알고리즘을 통해 evict 될 cache slot을 선정한다.

buffer\_cache\_flush\_entry() : 하나의 cache slot이 dirty이면서 valid하다면 그 내용을 block\_write를 통해 디스크에 저장한다.

buffer\_cache\_flush\_all() : flush를 모든 cache slot에 대해 진행한다.

* 1. **시험 및 평가 내용**
* **Src/filesys make grade 수행결과를 캡처 하여 첨부.**

**텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명**