# Project03 wiki

컴퓨터소프트웨어학부

2016025141

고세진

---

# Specification

기존 xv6에서는 12개의 direct block, 1개의  indirect block을 지원하므로 최대 140 block 크기의 파일만 이용 가능하다. xv6가 더 큰 파일을 다룰 수 있게 하기 위해서 inode에 double indirect block을 구현한다. 기존 inode의 블록 구성을 11개의 direct block,  1개의 indirect block, 1개의 double indirect block으로 바꾼다.

inode에 double indirect를 구현하여 더 큰 파일을 다룰 수 있게한다.

---

# Design

1. Constant
    - FSSIZE 를 20,000으로 설정
    - direct block의 개수에 대한 상수인 NDIRECT를 11로 변경
    - double indirect block이 취급할 수 있는 데이터 블록의 개수 NDOULBEINDIRECT 추가
2. inode, dinode 구조체가 데이터 블록 배열 선언을 위해 크기를 (NDIRECT 상수 + 1)를 이용하므로 (NDIRECT + 2)를 이용하도록 수정한다.
3. bmap(struct inode *ip, uint bn) 함수 수정
    - 2-level indirect block이 가리키는 데이터의 block number를 return하도록 수정

        - 1-level indirect block의 각 데이터 블록을 나타내기 위해 bn / NINDIRECT 이용
        - 1-level indirect block의 각 데이터 블록을 나타내기 위해 bn % NINDIRECT 이용
    - 2-level indirect block이 가리키는 데이터 block이 없을경우, balloc을 통해 새로 할당하고 log_write를 이용하여 업데이트 정보를 반영함.
4. itrunc()
    - 2-level indirect block이 가리키는 데이터의 block도 deallocates 할 수 있도록 수정
        - data block, 2-level indirect block, 1-level indirect block 순서대로 free 진행

---

# Implementation

## constants

```
#define FSSIZE       20000  // size of file system in blocks
```

```
#define NDIRECT 11 //direct data block의 개수는 11개로 감소
#define NINDIRECT (BSIZE / sizeof(uint))
//double indirect block이 다루는 데이터의 개수 : NINDIRECT^2
#define NDOUBLEINDIRECT (NINDIRECT * NINDIRECT)
//파일의 최대 데이터 블록 수에도 반영
#define MAXFILE (NDIRECT + NINDIRECT + NDOUBLEINDIRECT)
```

---

## inode, dinode

```
struct inode {
  ...
  uint addrs[NDIRECT+1+1];
};
```

```
struct dinode {
  ...
  uint addrs[NDIRECT+1+1];   // Data block addresses
};
```

## bmap

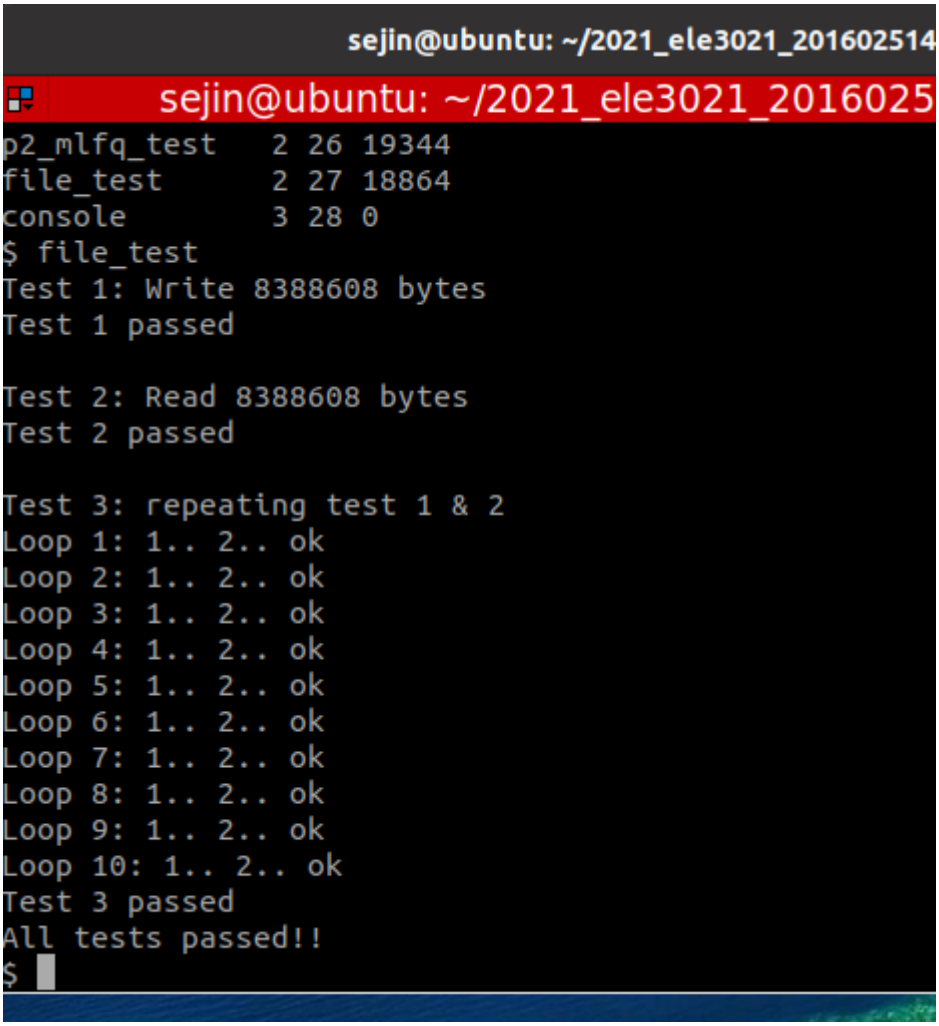
```
static uint
bmap(struct inode *ip, uint bn)
{
  ...
  bn -= NDIRECT;
  ...
  if(bn < NINDIRECT){
    ...
  }
  bn -= NINDIRECT;
  if(bn < NDOUBLEINDIRECT){
    // Load indirect block, allocating if necessary.
    if((addr = ip->addrs[NDIRECT + 1]) == 0)
      ip->addrs[NDIRECT + 1] = addr = balloc(ip->dev); // block alLOC

    // STEP1 bp is pointer to 1-level indirect block
    bp = bread(ip->dev, addr);
    a = (uint*)bp->data;

    int indexInLevel1 = bn / NINDIRECT;
    int indexInLevel2 = bn % NINDIRECT;

    if((addr = a[indexInLevel1]) == 0){
      a[indexInLevel1] = addr = balloc(ip->dev);
      log_write(bp);
    }
    brelse(bp);

    // STEP2 : bp is pointer to 2-level indirect block
    bp = bread(ip->dev, addr);
    a = (uint*)bp->data;

    if((addr = a[indexInLevel2]) == 0){
        a[indexInLevel2] = addr = balloc(ip->dev);
        log_write(bp);
    }
    brelse(bp);
    return addr;
  }

  panic("bmap: out of range");
}
```

### itrunc

```
static void
itrunc(struct inode *ip)
{
  int i, j, k;
  struct buf *bp;
  uint *a;
  uint *b;

  for(i = 0; i < NDIRECT; i++){
    ...
  }
  if(ip->addrs[NDIRECT]){
     ...
  }

  if(ip->addrs[NDIRECT+1]){
    //addr13
    bp = bread(ip->dev, ip->addrs[NDIRECT+1]);
    a = (uint*)bp->data;// block in 1 level

    for(j = 0; j < NINDIRECT; j++){
      if(a[j]){
        struct buf* bp2 = bread(ip->dev, a[j]); //
        b = (uint*)bp2->data; //block in level 2

        for(k = 0; k < NINDIRECT; k++){
          if(b[k]){
            bfree(ip->dev, b[k]);
          }
        }
        brelse(bp2);
        bfree(ip->dev, a[j]);
        a[j]=0;
      }
    }
    brelse(bp);
    bfree(ip->dev, ip->addrs[NDIRECT+1]);
    ip->addrs[NDIRECT+1] = 0;
  }

  ip->size = 0;
  iupdate(ip);
}
```

---

# Result


sejin@ubuntu: ~/2021_ele3021_201602514
sejin@ubuntu: ~/2021_ele3021_2016025
p2_mlfq_test 2 26 19344
file_test 2 27 18864
console 3 28 0
$ file_test
Test 1: Write 8388608 bytes
Test 1 passed
Test 2: Read 8388608 bytes
Test 2 passed
Test 3: repeating test 1 & 2
Loop 1: 1.. 2.. ok
Loop 2: 1.. 2.. ok
Loop 3: 1.. 2.. ok
Loop 4: 1.. 2.. ok
Loop 5: 1.. 2.. ok
Loop 6: 1.. 2.. ok
Loop 7: 1.. 2.. ok
Loop 8: 1.. 2.. ok
Loop 9: 1.. 2.. ok
Loop 10: 1.. 2.. ok
Test 3 passed
All tests passed!!
$