ELE3021_project03_12299_2018009216

구현 목표

- Multi Indirect
- Symbolic Link
- Buffered I/O

실행방법

1. make
2. make fs.img
3. qemu-system-i386 -nographic -serial mon:stdio -hdb fs.img xv6.img -smp 1 -m 512

Multi Indirect

DESIGN

기존의 direct + single indirect ⇒ direct + single indirect + double indirect + triple indirect로 바꿔주면 됩니다.

이를 위해서 기존의 fs.c에 있는 bmap함수를 수정해주었습니다.

기존의 코드를 single indirect코드를 참조하여 double, triple indirect를 만들었습니다.

그리고 param.h에 있는 FSSIZE도 충분히 큰 50000으로 바꿔주었습니다.

또한 fs.h에 있는 MAXFILE도 바꿔주었습니다.

IMPLEMENT

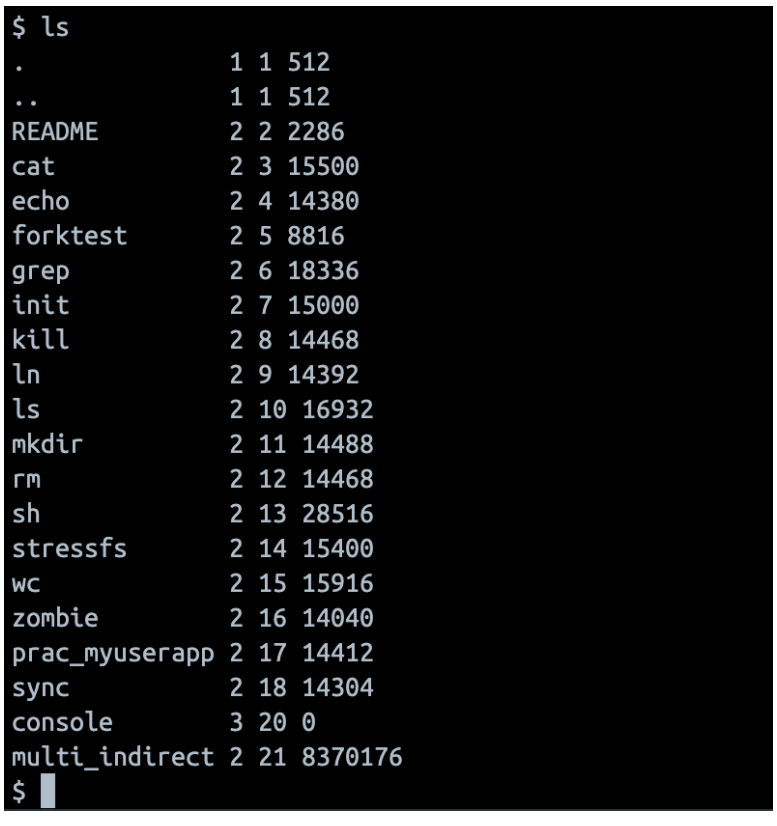
```
//fs.c

//bmap

static uint
bmap(struct inode *ip, uint bn) // bn : block number : 접근하고자 하는 block
{
  uint addr, *a;
  struct buf *bp; // buffer pointer

  if(bn < NDIRECT){ // inode는 13개인데...
    if((addr = ip->addrs[bn]) == 0) // 할당이 안된경우 할당해줌
      ip->addrs[bn] = addr = balloc(ip->dev);
    return addr;
  }
  bn -= NDIRECT;
  // #define NINDIRECT (BSIZE / sizeof(uint)), BSIZE = 512

  // single indirect
  if(bn < NINDIRECT){
    // Load indirect block, allocating if necessary.
    if((addr = ip->addrs[NDIRECT]) == 0) // 해당블록이 없는 경우
      ip->addrs[NDIRECT] = addr = balloc(ip->dev); //
    bp = bread(ip->dev, addr); // block table의 주소, 가 저장된 디바이스의 번호, 블록번호
    a = (uint*)bp->data; // 여러개의 block table
    if((addr = a[bn]) == 0){ // 해당블록에 주소할당x
      a[bn] = addr = balloc(ip->dev);
      log_write(bp); // log를 남김 왜 남기는 거???
```

```
    }
    brelse(bp); // buffer cache 반환
    return addr;
  }

  bn -= NINDIRECT;

  // double indirect
  if(bn < NINDIRECT * NINDIRECT){
    // 첫번째 indirect table
    if((addr = ip->addrs[NDIRECT + 1]) == 0)
      ip->addrs[NDIRECT + 1] = addr = balloc(ip->dev);
    bp = bread(ip->dev, addr); //첫번째 block table
    a = (uint*)bp->data; // 여러개의 block table

    // 두번째 indirect table
    if((addr = a[bn / NINDIRECT]) == 0){
      a[bn / NINDIRECT] = addr = balloc(ip->dev);
      log_write(bp);
    }
    brelse(bp);

    bp = bread(ip->dev, addr); // addr : 2번쨔 block table
    a = (uint*)bp->data;

    // data를 넣어줄 block
    if((addr = a[bn % NINDIRECT]) == 0){
      a[bn % NINDIRECT] = addr = balloc(ip->dev);
      log_write(bp);
    }
    brelse(bp);
    return addr;
  }


  // triple indirect
  bn -= (NINDIRECT * NINDIRECT);

  if(bn < (NINDIRECT * NINDIRECT * NINDIRECT)){
    if((addr = ip->addrs[NDIRECT + 2]) == 0) // 첫번째 table
      ip->addrs[NDIRECT + 2] = addr = balloc(ip->dev);
    bp = bread(ip->dev, addr);
    a = (uint*)bp->data;

    // 두번쨔 indirect table
    if((addr = a[bn/(NINDIRECT * NINDIRECT)]) == 0){
      a[bn/(NINDIRECT * NINDIRECT)] = addr = balloc(ip->dev);
      log_write(bp);
    }
    brelse(bp);

    // 세번째 indirect table // 여기서 문제
    bp = bread(ip->dev, addr); // addr : 2번쨔 block table
    a = (uint*)bp->data;

    if((addr = a[(bn % (NINDIRECT * NINDIRECT)) / NINDIRECT]) == 0){
      a[(bn % (NINDIRECT * NINDIRECT)) / NINDIRECT] = addr = balloc(ip->dev);
      log_write(bp);
```

```
    }
    brelse(bp);

    // 세번쨰 indirect table과 연결된 data block
    bp = bread(ip->dev, addr); // addr : 2번쨰 block table
    a = (uint*)bp->data;

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

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

RESULT

8MB까지 들어가는 걸 확인했다. ⇒ double과 triple이 작동한다.



Symbolic Link

GOAL

xv6는 Hard Link를 지원하지만 Symbolic Link는 지원하지 않음

xv6에 Symbolic Link를 구현하는 것이 목표

DESIGN

file구조체에서 addrs가 똑같으면 똑같은 데이터를 공유하는 것임

따라서 addrs를 복사하여 같은 파일을 가리키도록 디자인

기존의 hard link는 sysfile.c, fs.c, ln.c 3개의 파일에 구현되어있다. 따라서 이 파일에 구현하겠다.

ln.c/main ⇒ sysfile.c/sys_link ⇒ fs.c/dirlink 로 불린다.

IMPLEMENT

```
// ln.c //-h, -s 옵션을 받을 수 있게 인자를 1개 추가했다.

#include "types.h"
#include "stat.h"
#include "user.h"

int
main(int argc, char *argv[])
{
  if(argc != 4){
    printf(2, "Usage: ln -h/-s old new\n");
    exit();
  }
  if(link(argv[1], argv[2], argv[3]) < 0)
    printf(2, "link %s %s %s: failed\n", argv[1], argv[2], argv[3]);
  exit();
}
```

```
// sysfile.c/sys_link
int
sys_link(void)
{
    ...

    // 여기서 추가로 -h/ -s 옵션을 받으므로 &option을 추가함
  if(argstr(0, &option) < 0 || argstr(1, &old) < 0 || argstr(2, &new) < 0)
    return -1;

  // hard link인 경우 //////////////
  if(option[0] == '-' && option[1] == 'h'){
    cprintf("sysfile.c/sys_link hard link start\n");
    begin_op();
    if((ip = namei(old)) == 0){ // namei : 주어진 old의 inode를 가져옴
      end_op();
      return -1;
    }

    ilock(ip);
    if(ip->type == T_DIR){ // ip가 디렉토이면 오류
      iunlockput(ip);
      end_op();
      return -1;
    }

    //
    ip->nlink++; // ip가 디렉토리가 아닌경우
    iupdate(ip); // inode의 변경사항을 disk에 저장해줌
    iunlock(ip);

    if((dp = nameiparent(new, name)) == 0) // 부모 디렉토리의 inode를 가져옴 // dp : directory pointe
      goto bad;
    ilock(dp); // 부모 디렉토리의 inode lock
    // dirlink : dp에 새로운 디렉토리 엔트리를 추가
    if(dp->dev != ip->dev || dirlink(dp, name, ip->inum) < 0){
      iunlockput(dp);
```

```
      goto bad;
    }
    iunlockput(dp);
    iput(ip);

    end_op();

    return 0;
  }
  // symbolic link인 경우 -------------------------
  else if(option[0] == '-' && option[1] == 's'){
    cprintf("sysfile.c/sys_link symbolic link start\n");
    begin_op();
    if((ip = namei(old)) == 0){
      end_op();
      cprintf("sysfile.c/sys_link symbolic_link namei(old) error!\n");
      return -1;
    }

    ilock(ip);
    if(ip->type == T_DIR){ // ip가 디렉토이면 오류
      iunlockput(ip);
      end_op();
      return -1;
    }
    // 여기 까진 hard link와 같음
    // ip->nlink++; symbolic link이므로 => hard link와의 차이점
    iupdate(ip); // inode의 변경사항을 disk에 저장해줌
    iunlock(ip);

    if((dp = nameiparent(new, name)) == 0) // 부모 디렉토리의 inode를 가져옴 // dp : directory pointe
      goto bad;
    ilock(dp); // 부모 디렉토리의 inode lock

    // ip : old의 inode pointer
    // dp : new의 부모 디렉토리의 inode pointer

    // 여기서 dirlink를 고쳐야 할 듯
    if(dp->dev != ip->dev || dirlink(dp, name, ip->inum) < 0){
      iunlockput(dp);
      goto bad1;
    }
    //create()

    iunlockput(dp);
    iput(ip);

    end_op();

    return 0;
  }else{
    cprintf("sysfile.c/sys_link option error!\n");
    return -1;
  }


bad: // hard의 bad
  ilock(ip);
```

```
  ip->nlink--;
  iupdate(ip);
  iunlockput(ip);
  end_op();
  return -1;

bad1: // symbolic의 bad
  ilock(ip);
  iupdate(ip);
  iunlockput(ip);
  end_op();
  return -1;
}
```

RESULT

hard link test

```
$ echo "hard link test" > t1.txt
$ ln -h t1.txt hard.txt
sysfile.c/sys_link hard link start
$ cat t1.txt
"hard link test"
$ cat hard.txt
"hard link test"
$ rm t1.txt
$ ls
.              1 1 512
..             1 1 512
README         2 2 2286
cat            2 3 15500
echo           2 4 14380
forktest       2 5 8816
grep           2 6 18336
init           2 7 15000
kill           2 8 14468
ln             2 9 14392
ls             2 10 16932
mkdir          2 11 14488
rm             2 12 14468
sh             2 13 28516
stressfs       2 14 15400
wc             2 15 15916
zombie         2 16 14040
prac_myuserapp 2 17 14412
sync           2 18 14304
proj3_1_test   2 19 16784
console        3 20 0
hard.txt       2 21 17
$ cat hard.txt
"hard link test"
$
```

symbolic link test

```
$ echo "symbolic link test" > t2.txt
$ ln -s t2.txt symbolic.txt
sysfile.c/sys_link symbolic link start
$ cat t2.txt
"symbolic link test"
$ cat symbolic.txt
"symbolic link test"
$ rm t2.txt
$ ls
.              1 1 512
..             1 1 512
README         2 2 2286
cat            2 3 15500
echo           2 4 14380
forktest       2 5 8816
grep           2 6 18336
init           2 7 15000
kill           2 8 14468
ln             2 9 14392
ls             2 10 16932
mkdir          2 11 14488
rm             2 12 14468
sh             2 13 28516
stressfs       2 14 15400
wc             2 15 15916
zombie         2 16 14040
prac_myuserapp 2 17 14412
sync           2 18 14304
proj3_1_test   2 19 16784
console        3 20 0
hard.txt       2 21 17
symbolic.txt   0 22 0
$ cat symbolic.txt
$
```

Sync

GOAL

sync 함수를 호출될 때만 flush하는 Buffered IO구현

DESIGN

sync system call구현을 위해 살펴봐야할 것은 모두 log.c 파일에 있습니다.

1. log 구조체
2. logheader 구조체
3. void begin_op()
4. void end_op()
5. commit()
6. sync()

log 구조체를 참조하여 현재 buffer에 있는 block의 수를 return하기 위해 참조했습니다.

log 구조체에 있는 logheader의 n 값이 buffer에 있는 block의 수입니다.

begin_op()에서는 기존의 buffer사이즈가 넘어가면 sleep()을 호출합니다. 이를 강제로 flush시킬수 있게 바꾸어주었습니다.

end_op()의 경우 commit()부분을 sync() 로 따로 빼와서 구현하였습니다.

IMPLEMENT

```
// log.c
// 기존의 buffer가 가득차면 sleep하던 것을 sync()를 호출하여 강제로 flush하게 해줍니다.

void
begin_op(void)
{
```

```
  acquire(&log.lock);
  while(1){
    if

        ...

    } else if(log.lh.n + (log.outstanding+1)*MAXOPBLOCKS > LOGSIZE){
      cprintf("force buffer flush\n");
      sync();
    }

        ...
```

```
// log.c
// 기존에 있는 commit을 하는 부분을 전부 삭제했다.

void
end_op(void) // commit부분 빼야함
{
  acquire(&log.lock);
  log.outstanding -= 1; // 현재 진행중인 작업의 수
  if(log.committing) // 이미 커밋중인 경우 예외 발생
    panic("log.committing");

  if(log.outstanding != 0){
    wakeup(&log);
  }
  release(&log.lock);
}
```

```
// log.c
// flush해주는 sync

int
sync()
{
  cprintf("log.c/sync start\n");

  int block_num = 0; // 성공시 flush된 block의 수를 return 하기 위해
  int do_commit = 0;

  acquire(&log.lock);
  if(log.committing){ // 이미 커밋중인 경우 예외 발생
    cprintf("log.committing");
    return -1;
  }
  if(log.outstanding == 0){ // 마지막 작업을 하는 중이면
    do_commit = 1;          // do_commit을 지정함
    log.committing = 1;     // 커밋이 진행중임을 나타냄
  } else {
    // begin_op() may be waiting for log space,
    // and decrementing log.outstanding has decreased
    // the amount of reserved space.
    wakeup(&log); //
  }
```

```
  release(&log.lock);

  if(do_commit){ //
    block_num = log.lh.n;
    //cprintf("block num : %d %d %d\n",log.size, log.lh.n, block_num);
    // call commit w/o holding locks, since not allowed
    // to sleep with locks.
    commit(); // log에 기록된 변경내용을 디스크에 커밋함
    acquire(&log.lock);
    log.committing = 0;
    wakeup(&log);
    release(&log.lock);
  }
  return block_num; // flush된 block의 수
}
```

```
// log.c
// sync system call을 위한 wrapper function

int
sys_sync(void)
{
  int num = 0;
  num = sync();

  return num;
}
```

```
// myapp.c
// sync 테스트를 위한 파일입니다.

#include "types.h" //stick order!
#include "stat.h"
#include "user.h"

int main(int argc, char* argv[])
{
    int block_num = 0;
    printf(1,"proj3_test start\n");
    block_num = sync();
    printf(1,"flushed buffer num : %d\n",block_num);
    exit();
};
```

RESULT

1. t1.txt와 t2.txt만 myapp에 있는 sync를 호출하여 flush해주고 t3.txt와 t4.txt의 경우는 flush하지 않았습니다. ⇒ 다시 myapp 을 sync로 바꿔주었습니다. sync를 호출하면 sync syscall이 호출됩니다.

```
$ echo "test1" > t1.txt
$ echo "test2" > t2.txt
$ myapp
proj3_test start
log.c/sync start
block num : 30 5 5
flushed buffer num : 5
$
exec test: fail
exec  failed
$ echo "test3" > t3.txt
$ echo "test4" > t4.txt
$
```

2. 종료후 재시작해였습니다.

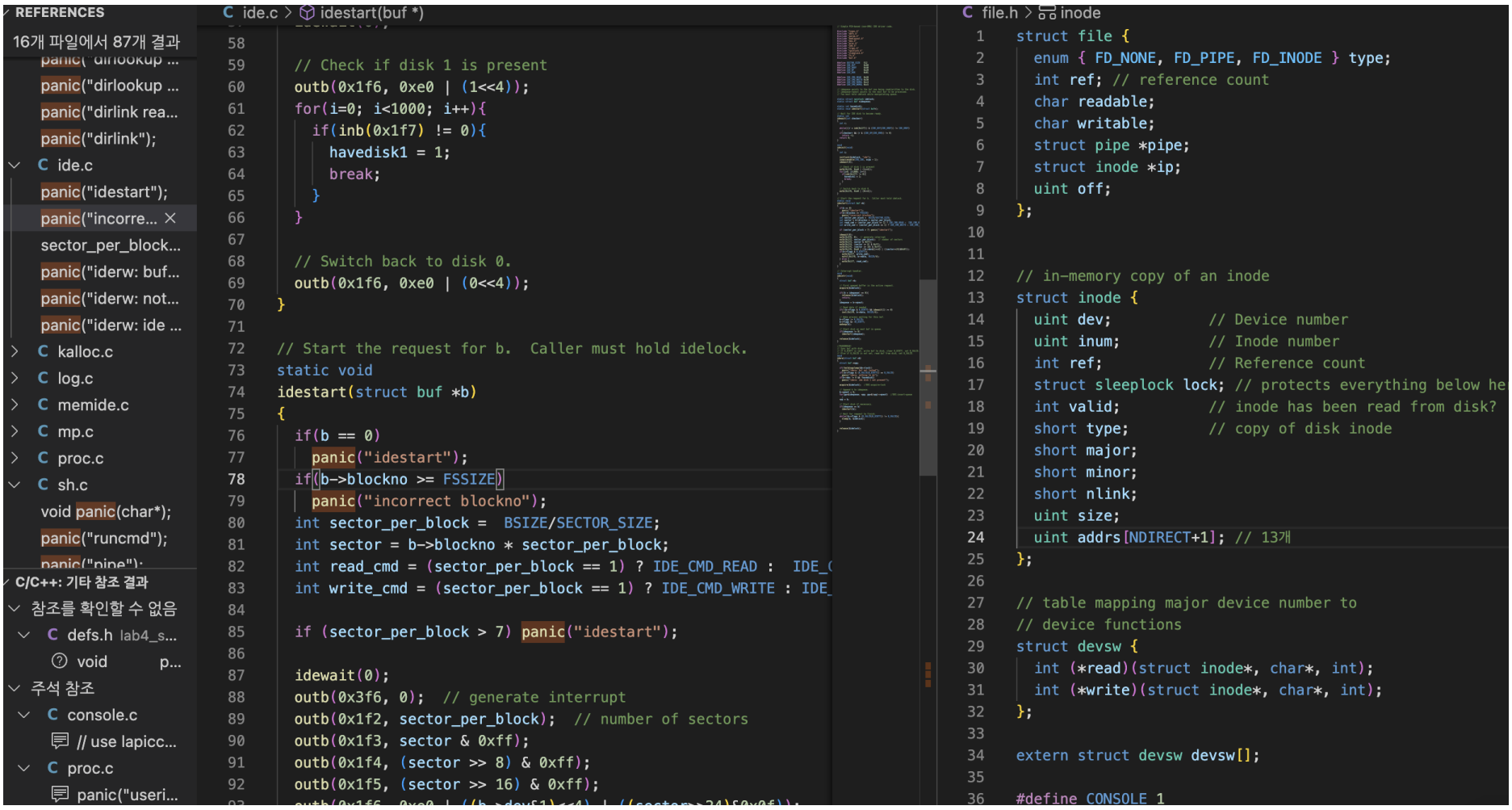
```
$ QEMU: Terminated
# qemu-system-i386 -nographic -serial mon:stdio -hdb fs.img xv6.img -smp 1 -m 512
```

3. ls로 확인결과 myapp을 통해 sync를 호출하기 이전의 파일들만 저장된것을 확인 할 수 있습니다.

```
Booting from Hard Disk...
xv6...
cpu0: starting 0
sb: size 50000 nblocks 49929 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ ls
.              1 1 512
..             1 1 512
README         2 2 2286
cat            2 3 15500
echo           2 4 14380
forktest       2 5 8816
grep           2 6 18336
init           2 7 15000
kill           2 8 14468
ln             2 9 14392
ls             2 10 16932
mkdir          2 11 14488
rm             2 12 14468
sh             2 13 28516
stressfs       2 14 15400
wc             2 15 15916
zombie         2 16 14040
prac_myuserapp 2 17 14412
myapp          2 18 14360
console        3 19 0
t1.txt         2 20 8
t2.txt         2 21 8
$
```

Trouble shooting

계속 incorrect blockno이라는 에러가 나타났습니다.



코드 확인결과 FSSIZE의 문제임을 확인하고 FSSIZE의 크기를 늘려주었습니다.

git에 에러가 발생하여 계속 push가 안되고 branch가 하나지만 branch가 다르다는 error가 계속 발생하였습니다.
계속된 구글링과 시도끝에 문제를 해결하였습니다.