Multi Indirect & Symbolic Link & Sync

xv6의 file system에 관련된 기능들을 구현한다.

Multi Indirect

xv6의 기본 inode는 direct 포인터 12개 + indirect 포인터 1개로 구성되어 있다. 이것에 double indirect와 triple indirect를 추가한다.

```
// file.h

// in-memory copy of an inode
struct inode {
  uint dev;           // Device number
  uint inum;          // Inode number
  int ref;            // Reference count
  struct sleeplock lock; // protects everything below here
  int valid;          // inode has been read from disk?

  short type;           // File type
  short major;          // Major device number (T_DEV only)
  short minor;          // Minor device number (T_DEV only)
  short nlink;          // Number of links to inode in file s
  uint size;            // Size of file (bytes)
  uint addrs[NDIRECT+3];   // Data block addresses
  char path[40];        // symbolic link path
};
```

구조체 크기 유지를 위해 NDIRECT는 0으로 설정한다.

addrs[0] → single indirect

addrs[1] → double indirect

addrs[2]→ triple indirect로 설정한다.

```
// fs.c

static uint
bmap(struct inode *ip, uint bn)
{
  uint addr, *a, idx;
  struct buf *bp, *bp1, *bp2;

  if(bn < NDIRECT){
    if((addr = ip->addrs[bn]) == 0)
      ip->addrs[bn] = addr = balloc(ip->dev);
    return addr;
  }
  bn -= NDIRECT;

  if(bn < NINDIRECT){
    // Load indirect block, allocating if necessary.
    if((addr = ip->addrs[NDIRECT]) == 0)
      ip->addrs[NDIRECT] = addr = balloc(ip->dev);
    bp = bread(ip->dev, addr);
    a = (uint*)bp->data;

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

  bn -= NINDIRECT;

  if(bn < NDADDR){
    // Load indirect block, allocating if necessary.
    if((addr = ip->addrs[NDIRECT+1]) == 0)
      ip->addrs[NDIRECT+1] = addr = balloc(ip->dev);
```

```
    bp = bread(ip->dev, addr);
    a = (uint*)bp->data;

    idx = bn / NINDIRECT;
    bn = bn % NINDIRECT;
    if((addr = a[idx]) == 0)
      a[idx] = addr = balloc(ip->dev);
    bp1 = bread(ip->dev, addr);
    a = (uint*)bp1->data;

    if((addr = a[bn]) == 0){
      a[bn] = addr = balloc(ip->dev);
      log_write(bp1);
      log_write(bp);
    }
    brelse(bp1);
    brelse(bp);

    return addr;
  }
  bn -= NDADDR;

  if(bn < NTADDR){
    // Load indirect block, allocating if necessary.
    if((addr = ip->addrs[NDIRECT+2]) == 0)
      ip->addrs[NDIRECT+2] = addr = balloc(ip->dev);
    bp = bread(ip->dev, addr);
    a = (uint*)bp->data;

    idx = bn / NDADDR;
    bn = bn % NDADDR;
    if((addr = a[idx]) == 0)
      a[idx] = addr = balloc(ip->dev);
    bp1 = bread(ip->dev, addr);
    a = (uint*)bp1->data;

    idx = bn / NINDIRECT;
    bn = bn % NINDIRECT;
```

```
    if((addr = a[idx]) == 0)
      a[idx] = addr = balloc(ip->dev);
    bp2 = bread(ip->dev, addr);
    a = (uint*)bp2->data;

    if((addr = a[bn]) == 0){
      a[bn] = addr = balloc(ip->dev);
      log_write(bp2);
      log_write(bp1);
      log_write(bp);
    }
    brelse(bp2);
    brelse(bp1);
    brelse(bp);

    return addr;
  }

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

기존의 indirect시에 동작방식을 참고하여 double, triple을 만든다.

Symbolic Link

ln.c 에는 기본적으로 하드 링크가 구현되어 있다. symbolic link를 추가한다.

```
// ln.c

int
main(int argc, char *argv[])
{
  if(argc != 4 || argv[1][0] != '-' || argv[1][2] != 0){
    printf(2, "Usage: ln linktype old new\n");
    exit();
  }
  if (argv[1][1] == 'h'){
```

```
    if(link(argv[2], argv[3]) < 0)
      printf(2, "link %s %s: failed\n", argv[1], argv[2]);
  } else if (argv[1][1] == 's'){
    if(open(argv[3], O_CREATE) < 0 || slink(argv[2], argv[3])
      printf(2, "symbolic link %s %s: failed\n", argv[1], arg
  }
  exit();
}
```

open함수로 파일 엔트리와 inode를 만든 후, slink system call을 추가하여 호출한다.

```
// sysfile.c

int
sys_slink(void){
  char *new, *old;
  struct inode *ip;
  int i;

  if(argstr(0, &old) < 0 || argstr(1, &new) < 0)
    return -1;

  begin_op();

  if((ip = namei(new)) == 0){
      end_op();
      return -1;
  }
  ilock(ip);

  for (i = 0; old[i] != 0 && i < 40; i++){
    ip->path[i] = old[i];
  }
  iunlockput(ip);

  end_op();
```

```
  return 0;
}
```

아까 만들어두었던 path 변수에 원래 파일의 주소를 집어넣는다.

```
// sysfile.c sys_open()

// ...
if (ip->path[0] != 0){
    if ((dp = namei(ip->path)) == 0){
      iunlockput(ip);
      end_op();
      return -1;
    }
    iunlockput(ip);
  } else dp = ip;
// ...
```

open시 path 변수에 값이 들어있으면 symbolic으로 연결되었다는 뜻이므로 값을 불러와 해당 위치의 inode를 연다.