OS project-5

File extension & Block Group

Department of Software

2016312568

Jung Hee Yoon

이번 과제는 파일 시스템에서 Max file size를 Double Indirect를 통해 늘리고 BlockGroup을 구현하는 과제이다. 이를 구현하기 위해서는 fs.c, fs.h, param.h, mkfs.c을 수정해 주어야한다. 또한 Makefile의 NCPU가 2일경우 테스트하는데 오래걸려 NCPU를 1로 수정한 뒤에 코딩을 진행하였다. 또한 원래 Block Group (FFS)를 구현하는데에 있어 균등분배와 루트에서 할당할때에 minimum bitmap checked인곳을 찾고 넣어야 한다고 생각하여 구현하였으나 필요하지 않다고 하여 뺐습니다. 다음 페이지부터 File extension, BlockGroup 각각설명하도록 하겠습니다.

A) Max File size extension

이를 구현하기 위해서는 먼저 fs.h에서 설정되어있는 구조를 수정하여준다. 원래는 NDIRECT와 NINDIRECT로만 이루어졌지만 다음과 같이 수정하여준다.

```
#define NDIRECT 11
#define NINDIRECT (BSIZE / sizeof(uint))
#define N2INDIRECT (BSIZE / sizeof(uint)) * (BSIZE / sizeof(uint))
#define MAXFILE (NDIRECT + NINDIRECT + N2INDIRECT)
```

그런 다음, 이 구조로 Mapping을 하기위하여 fs.c의 bmap함수에 다음 부분을 추가하여 준다.

```
468   if(bn < N2INDIRECT){
469       if((addr = ip->addrs[NDIRECT+1]) == 0)
470         ip->addrs[NDIRECT+1] = addr = balloc(ip->dev);
471
472       bp = bread(ip->dev, addr);
473       a = (uint*)bp->data;
474       if((addr = a[bn/NINDIRECT]) == 0){
475           a[bn/NINDIRECT] = addr = balloc(ip->dev);
476           log_write(bp);
477       }
478       brelse(bp);
479
480       bp = bread(ip->dev, addr);
481       a = (uint*)bp->data;
482       if((addr = a[bn%NINDIRECT]) == 0){
483           a[bn%NINDIRECT] = addr = balloc(ip->dev);
484           log_write(bp);
485       }
486       brelse(bp);
487       return addr;
488   }
```

이렇게 하면 file max size가 증가되어 Mapping 된다.

하지만 지우는 것 또한 이 구조를 따라야 하기에 itrunc에 다음과 같은 부분을 추가하여 준다.

```
525
526   if(ip->addrs[NDIRECT+1]){
527     bp = bread(ip->dev, ip->addrs[NDIRECT+1]);
528     a = (uint*)bp->data;
529     for(j = 0; j < NINDIRECT; j++){
530       if(a[j]){
531           ck = bread(ip->dev,a[j]);
532           b = (uint*)ck->data;
533           for(i = 0; i< NINDIRECT; i++){
534                 if(b[i])
535                     bfree(ip->dev,b[i]);
536           }
537           brelse(ck);
538           bfree(ip->dev, a[j]);
539       }
540     }
541
542     brelse(bp);
543     bfree(ip->dev, ip->addrs[NDIRECT+1]);
544     ip->addrs[NDIRECT+1] = 0;
545   }
546
547   ip->size = 0;
548   iupdate(ip);
549 }
```

이렇게 하면 max File size를 Double Indirect를 추가하여 수정할 수 있다.

B) Block Group

먼저, 내가 구현한 File System의 구조를 보면 다음과 같다.

위의 구조에서 Boot Block, Super Block, Log Block, Swap area블록이 nmeta에 들어가며 이후에 남은 공간이 32개의 Block Group을 구성한다. Block Group에 대한 정보들은 Super block에 들어가며 이는 mkfs.c의 main에서 실행된다. 따라서 fs.h에서 Super block의 구조를 수정한뒤에 mkfs.c의 main에서 다음과 같이 값을 넣어준다.

```
struct superblock {
  uint size;
  uint nblocks;
  uint ninodes;
  uint nlog;
  uint logstart;
  uint BGsize;
  uint nmeta;
  uint nswap;
  uint nBG;
  uint nbmapBG;
  uint ninodeBG;
  uint metaBG;
};
```

```
// 1 fs block = 1 disk sector
int usable = FSSIZE-2-LOGSIZE;
int BGsize = usable/32;
if(BGsize<4096) BGsize=4096;
int nBG = usable/BGsize;
int nbmapBG;
if(BGsize%4096==0) nbmapBG=BGsize/4096;
else nbmapBG=BGsize/4096+1;
int ninodeBG=BGsize/32;
int metaBG = ninodeBG+nbmapBG;
int nswap = usable%BGsize;
nmeta = 2+nlog+nswap;
nblocks = FSSIZE-nmeta;
sb.size = xint(FSSIZE);

sb.nblocks = xint(nblocks);
sb.ninodes = xint(ninodeBG*nBG);
sb.nswap = xint(nswap);
sb.nlog = xint(nlog);
sb.nmeta = xint(nmeta);
sb.logstart = xint(2);

sb.nBG = xint(nBG);
sb.BGsize = xint(BGsize);
sb.ninodeBG = xint(ninodeBG);
sb.nbmapBG = xint(nbmapBG);
sb.metaBG = xint(metaBG);
```

이처럼 하면 Super block에 Block Group에 대한 정보가 들어간다. 이제 이 구조로 allocate를 해보자. 이를 하기 앞서 mkfs.c에서 main을 Block Group 구조로 바꾼것에 맞게 바꾸어준다. mkfs.c의 main, balloc과 iappend에서 약간의 수정을 하였다. 또한 원래 구조에 맞는 매크로함수들 (IBLOCK, BBLOCK)은 Block Group구조에 맞게 바꾸어주고 자주 사용할 만한 함수들을 매크로로 정의해 놓는다.

```
108 static uint balloc(uint dev)
109 {
110     int m;
111     struct buf *bp;
112 
113     for(int i=0;i<sb.nBG;i++)
114     {
115         int firstblock = BBLOCKGROUPSTART(i,sb);
116 
117         for(int j=0;j<sb.nbmapBG;j++)
118         {
119             bp=bread(dev,firstblock+sb.ninodeBG+j);
120             for(int k=0;k<BPB;k++)
121             {
122                 int alloc = firstblock+j*BPB+k;
123                 if((j*BPB)+k < sb.metaBG) {
124                     k++;
125                     continue;
126                 }
127                 if(alloc>=firstblock+sb.BGsize) break;
128 
129                 m = 1 <<(k%8);
130                 if((bp->data[k/8] & m)==0) {
131                     bp->data[k/8] |=m;
132                     log_write(bp);
133                     brelse(bp);
134                     bzero(dev,alloc);
135                     return alloc;
136                 }
137             }
138             brelse(bp);
139         }
140     }
141     panic("balloc: out of blocks");
142 }
```

그런다음 fs.c에 balloc함수를 다음과 같이 바꾸어 원하는 구조에 맞게 할당한다.
또한 할당한것을 지워주는 것도 수정하여야 하기에 다음과 같이 수정하여준다

```
145 bfree(int dev, uint b)
146 {
147   struct buf *bp;
148   int  m, bi;
149   readsb(dev, &sb);
150   bp = bread(dev, BBLOCK(b, sb));
151   bi = ((b-sb.nmeta)%sb.BGsize)%BPB;
152   m = 1 <<(bi%8);
153   if((bp->data[bi/8] & m) == 0) panic("freeing free block\n");
154   bp->data[bi/8] &= ~m;
155   log_write(bp);
156   brelse(bp);
157 }
```

이렇게 하면 file size extension과 Block Group화를 모두 마쳤다.

다음사진은 Block Group의 내용을 보여준다.

```
==========================================================
=======================Printed============================
==========================================================
sb==> FSsize : 20000, nblocks(FSSIZE - nmeta) : 16384, ninodes : 512
sb==> nlog : 30, nswap : 3584, logstart : 2
sb==> Block Group size : 4096, nmeta : 3616, number of Block Group : 4
sb==> bitmap block per Block Group : 1, inode block per Block Group : 128
sb==> meta blocks per Block Group : 129, data blocks per Block Group: 3967
==========================================================
==========================================================
```

또한 바꾼 구조로 test.c를 돌려보면 다음과 같이나온다.

```
==========================================================
=======================Printed============================
==========================================================
sb==> FSsize : 20000, nblocks(FSSIZE - nmeta) : 16384, ninodes : 512
sb==> nlog : 30, nswap : 3584, logstart : 2
sb==> Block Group size : 4096, nmeta : 3616, number of Block Group : 4
sb==> bitmap block per Block Group : 1, inode block per Block Group : 128
sb==> meta blocks per Block Group : 129, data blocks per Block Group: 3967
==========================================================
==========================================================
init: starting sh
$ test
..........................................................................................................................................................
.lapicid 0: panic: balloc: out of blocks
 8010122e 80101453 80101dbf 80101098 80104fe2 80104c89 80105d05 80105b1f 0 0QEMU: Terminated
```

여기서 out of blocks라는 패닉이 뜨게 되는데 이유는 FSsize를 20000으로 하면 잉여 데이터 블럭보다 filesize가 크기에 읽을수 없어 나는 에러이다. 이는 스왑영역이 넓어지면서 원래 파일사이즈 대로하면 쓸수 있어야하는 데이터가 스왑영역에 할당되면서 나는 에러이다. 이는 FSSIZE를 늘리면 다음과 같이 해결 가능하다.

```
==========================================================
=======================Printed============================
==========================================================
sb==> FSsize : 25000, nblocks(FSSIZE - nmeta) : 24576, ninodes : 768
sb==> nlog : 30, nswap : 392, logstart : 2
sb==> Block Group size : 4096, nmeta : 424, number of Block Group : 6
sb==> bitmap block per Block Group : 1, inode block per Block Group : 128
sb==> meta blocks per Block Group : 129, data blocks per Block Group: 3967
==========================================================
==========================================================
init: starting sh
$ ls
.              1 1 512
..             1 1 512
README         2 2 2327
cat            2 3 13684
echo           2 4 12696
forktest       2 5 8124
grep           2 6 15560
init           2 7 13276
kill           2 8 12748
ln             2 9 12644
ls             2 10 14832
mkdir          2 11 12828
rm             2 12 12804
sh             2 13 23288
stressfs       2 14 13476
usertests      2 15 56420
wc             2 16 14224
zombie         2 17 12468
test           2 18 13672
console        3 19 0
$ test
..........................................................................................................................................................
..............
wrote 16523 sectors
done; ok
$
```