#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <linux/fs.h>

#include <linux/ext2\_fs.h>

#include <time.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

// Type defs

typedef struct ext2\_group\_desc GD;

typedef struct ext2\_super\_block SUPER;

typedef struct ext2\_inode INODE;

typedef struct ext2\_dir\_entry\_2 DIR;

// PTRS

GD \*gp;

SUPER \*sp;

INODE \*ip;

DIR \*dp;

int inode\_count,

block\_count,

inode\_size,

block\_size,

block\_bitmap,

inode\_bitmap,

inode\_table,

first\_block;

int get\_block(int fd, int blk, char \*buf)

{

lseek(fd, (long)(blk\*1024), 0);

read(fd, buf, 1024);

}

int gd (int fd)

{

char buf[1024];

get\_block(fd, 2, buf);

gp = (GD \*)buf;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* GrOuP dEsCrIpToRs \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("bg\_block\_bitmap \t=\t%d\n", gp->bg\_block\_bitmap);

printf("bg\_inode\_bitmap \t=\t%d\n", gp->bg\_inode\_bitmap);

printf("bg\_inode\_table \t\t=\t%d\n", gp->bg\_inode\_table);

printf("bg\_free\_blocks\_count \t=\t%d\n", gp->bg\_free\_blocks\_count);

printf("bg\_free\_inodes\_count \t=\t%d\n", gp->bg\_free\_inodes\_count);

printf("bg\_used\_dirs\_count \t=\t%d\n", gp->bg\_used\_dirs\_count);

block\_bitmap = gp->bg\_block\_bitmap;

inode\_bitmap = gp->bg\_inode\_bitmap;

inode\_table = gp->bg\_inode\_table;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

int super(int fd)

{

char buf[1024];

get\_block(fd, 1, buf);

sp = (SUPER \*)buf;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SuPeRbLoCk \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("s\_inodes\_count \t\t=\t%d\n", sp->s\_inodes\_count);

printf("s\_blocks\_count \t\t=\t%d\n", sp->s\_blocks\_count);

printf("s\_free\_inodes\_count \t=\t%d\n", sp->s\_free\_inodes\_count);

printf("s\_free\_blocks\_count \t=\t%d\n", sp->s\_free\_blocks\_count);

printf("s\_log\_block\_size \t=\t%d\n", sp -> s\_log\_block\_size);

printf("s\_blocks\_per\_group \t=\t%d\n", sp->s\_blocks\_per\_group);

printf("s\_inodes\_per\_group \t=\t%d\n", sp->s\_inodes\_per\_group);

printf("s\_mnt\_count \t\t=\t%d\n", sp->s\_mnt\_count);

printf("s\_max\_mnt\_count \t=\t%d\n", sp->s\_max\_mnt\_count);

printf("s\_magic \t\t=\t%x\n", sp->s\_magic);

printf("s\_mtime \t\t=\t%s", ctime(&(sp->s\_mtime)));

printf("s\_inode\_size \t\t=\t%d\n", sp->s\_inode\_size);

inode\_count = sp->s\_inodes\_count;

block\_count = sp->s\_blocks\_count;

block\_size = 1024 << sp -> s\_log\_block\_size;

inode\_size = sp->s\_inode\_size;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

int imap(int fd)

{

char buf[1024];

unsigned char bitmap[inode\_size];

int i, j, k;

get\_block(fd, inode\_bitmap, buf);

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* iMaP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("inode\_bitmap: %d\n", inode\_bitmap);

printf("inode\_size: %d\n", inode\_size);

for (i = 0; i <= inode\_count; i++)

{

j = bitmap[i]/8;

k = bitmap[i]%8;

if (bitmap[j] & (1<<k))

printf("1");

else

printf("0");

if (i > 0)

{

if ((i%8) == 0)

printf(" ");

if ((i%80) == 0)

printf("\n");

}

j++;

}

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

int inode(int fd)

{

int i = 0, j;

char buf[1024];

ip = (INODE \*)malloc(inode\_size);

lseek(fd,1024\*inode\_table+inode\_size,SEEK\_SET);

read(fd,ip,inode\_size);

printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* iNoDe \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("inode\_block=%d\n", inode\_count / 8);

printf("mode=%x\n", ip->i\_mode);

printf("uid=%d\n", ip->i\_uid);

printf("gid=%d\n", ip->i\_gid);

printf("size=%d\n", ip->i\_size);

printf("time=%s", ctime(&(ip->i\_ctime)));

printf("link=%d\n", ip->i\_links\_count);

printf("i\_block[0]=%d\n", ip->i\_block[0]);

first\_block = ip->i\_block[0];

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

int bmap(int fd)

{

unsigned char bitmap[block\_size];

int i, j, k;

get\_block(fd, block\_bitmap, bitmap);

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* bmap \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("block\_bitmap: %d\n", block\_bitmap);

printf("block\_size: %d\n", block\_size);

for (i = 0; i <= block\_count; i++)

{

j = bitmap[i]/8;

k = bitmap[i]%8;

if (bitmap[j] & (1<<k))

printf("1");

else

printf("0");

if (i > 0)

{

if ((i%8) == 0)

printf(" ");

if ((i%80) == 0)

printf("\n");

}

j++;

}

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

int dir(int fd)

{

unsigned int size = 0;

unsigned char buf[1024];

lseek(fd, 1024\*first\_block, SEEK\_SET);

read(fd, &buf, 1024);

dp = (DIR \*)buf;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* DIRS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

while (size < inode\_size)

{

printf("%d %d %d %s\n", dp->inode, dp->file\_type,

dp->name\_len, dp->name);

dp = (void \*)dp + dp->rec\_len;

size += dp->rec\_len;

}

}

char \*device = "mydisk";

main(int argc, char \*argv[])

{

int fd;

if (argc > 1)

device = argv[1];

fd = open(device, O\_RDONLY);

if (fd < 0){

printf("open %s failed\n", device);

exit(1);

}

super(fd);

gd(fd);

bmap(fd);

imap(fd);

inode(fd);

dir(fd);

}