**PONTIFICIA UNIVERSIDAD CATOLICA MADRE Y MAESTRA**

****

**Nombre:**

Dhamar Martinez 2014 - 0047

**Materia:**

ST-ISC-364-T-001 - Sistemas Operativos 1

**Profesor:**

Alvaro A. Reyes P.

**Practica sobre:**

Práctica 3

**Fecha de Entrega:**

viernes, 23 de febrero de 2018

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include </home/felixal/Documents/Sistemas Operativos/SistemasOperativos1/Tarea 4/ext2\_fs.h>

#define BASE\_OFFSET 1024

#define USB\_DEVICE "/dev/sdc1"

#define BLOCK\_OFFSET(block) (BASE\_OFFSET + (block - 1) \* block\_size)

#define FILTER\_BIT(byte, i) ((byte >> i) & 0x01)

static unsigned int block\_size = 0;

/\*

\* Please IMPLEMENT THIS! This function will print out a portion of the

\* superblock to the console. It is called in main after reading in the super

\* block from the disk.

\*/

void print\_super\_block(struct ext2\_super\_block\* super);

/\*

\* The actual function to open and read raw bytes off the USB stick.

\* Takes a pointer to a superblock structure, fills it with the bytes

\* from the USB drive and returns the file descriptor (which is actually

\* just an int)

\*/

int open\_usb(struct ext2\_super\_block\* super);

int main(void){

struct ext2\_super\_block usb\_block;

int file\_descriptor;

file\_descriptor = open\_usb(&usb\_block);

print\_super\_block(&usb\_block);

return 0;

}

/\*

\* Implement this method!

\*/

void print\_super\_block(struct ext2\_super\_block\* super){

printf("inode count: %d\n",super -> s\_inodes\_count);

printf("blocks count: %d\n",super -> s\_blocks\_count);

printf("free blocks count: %d\n ",super -> s\_free\_blocks\_count);

printf("free inodes count: %d\n",super -> s\_free\_inodes\_count);

printf("first data block: %d\n",super -> s\_first\_data\_block);

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

printf("blocks per group: %d\n",super -> s\_blocks\_per\_group);

printf("inode per group: %d\n",super -> s\_inodes\_per\_group);

printf("size of the inode structure: %d\n",super -> s\_inode\_size);

/\*

\* Print out the inodes count, blocks count, free blocks count, free inodes

\* count, the first data block, block size, blocks per group, inodes per

\* group, and the size of the inode structure. The output should be

\* "<field>\t:<value>\n".

\*/

}

int open\_usb(struct ext2\_super\_block\* super){

int fd;

struct ext2\_group\_desc\* group = malloc(sizeof(struct ext2\_group\_desc));

/\* open USB device \*/

fd = open(USB\_DEVICE, O\_RDONLY); //opening the device for reading

if(fd < 0){ //some kind of error occurred

perror(USB\_DEVICE);

exit(1); //we give up at this point

}

/\* Now we read in Mr. Superblock \*/

/\* seeking across the 'disk' to the superblock location \*/

lseek(fd, BASE\_OFFSET, SEEK\_SET);

/\*actually reading in the bytes \*/

read(fd, super, sizeof(struct ext2\_super\_block));

/\* Some sanity checks \*/

/\* Make sure we're reading an EXT2 filesystem \*/

if(super->s\_magic != EXT2\_SUPER\_MAGIC){

fprintf(stderr, "Not an Ext2 filesystem!\n");

exit(1);

}

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

/\* closing the USB device \*/

close(fd);

return fd;

}

**EXT\_Fs.h**

/\*

\* linux/include/linux/ext2\_fs.h

\*

\* Copyright (C) 1992, 1993, 1994, 1995

\* Remy Card (card@masi.ibp.fr)

\* Laboratoire MASI - Institut Blaise Pascal

\* Universite Pierre et Marie Curie (Paris VI)

\*

\* from

\*

\* linux/include/linux/minix\_fs.h

\*

\* Copyright (C) 1991, 1992 Linus Torvalds

\*/

#ifndef \_LINUX\_EXT2\_FS\_H

#define \_LINUX\_EXT2\_FS\_H

#include <linux/types.h>

/\*

\* The second extended filesystem constants/structures

\*/

/\*

\* Define EXT2FS\_DEBUG to produce debug messages

\*/

#undef EXT2FS\_DEBUG

/\*

\* Define EXT2\_PREALLOCATE to preallocate data blocks for expanding files

\*/

#define EXT2\_PREALLOCATE

/\*

\* The second extended file system version

\*/

#define EXT2FS\_DATE     "95/08/09"

#define EXT2FS\_VERSION      "0.5b"

/\*

\* Debug code

\*/

#ifdef EXT2FS\_DEBUG

#   define ext2\_debug(f, a...)  { \

                    printk ("EXT2-fs DEBUG (%s, %d): %s:", \

                        \_\_FILE\_\_, \_\_LINE\_\_, \_\_FUNCTION\_\_); \

                   printk (f, ## a); \

                    }

#else

#   define ext2\_debug(f, a...)  /\*\*/

#endif

/\*

\* Special inodes numbers

\*/

#define EXT2\_BAD\_INO         1  /\* Bad blocks inode \*/

#define EXT2\_ROOT\_INO        2  /\* Root inode \*/

#define EXT2\_ACL\_IDX\_INO     3  /\* ACL inode \*/

#define EXT2\_ACL\_DATA\_INO    4  /\* ACL inode \*/

#define EXT2\_BOOT\_LOADER\_INO     5  /\* Boot loader inode \*/

#define EXT2\_UNDEL\_DIR\_INO   6  /\* Undelete directory inode \*/

/\* First non-reserved inode for old ext2 filesystems \*/

#define EXT2\_GOOD\_OLD\_FIRST\_INO 11

/\*

\* The second extended file system magic number

\*/

#define EXT2\_SUPER\_MAGIC    0xEF53

/\*

\* Maximal count of links to a file

\*/

#define EXT2\_LINK\_MAX       32000

/\*

\* Macro-instructions used to manage several block sizes

\*/

#define EXT2\_MIN\_BLOCK\_SIZE     1024

#define EXT2\_MAX\_BLOCK\_SIZE     4096

#define EXT2\_MIN\_BLOCK\_LOG\_SIZE      10

#ifdef \_\_KERNEL\_\_

# define EXT2\_BLOCK\_SIZE(s)     ((s)->s\_blocksize)

#else

# define EXT2\_BLOCK\_SIZE(s)     (EXT2\_MIN\_BLOCK\_SIZE << (s)->s\_log\_block\_size)

#endif

#define EXT2\_ACLE\_PER\_BLOCK(s)      (EXT2\_BLOCK\_SIZE(s) / sizeof (struct ext2\_acl\_entry))

#define EXT2\_ADDR\_PER\_BLOCK(s)      (EXT2\_BLOCK\_SIZE(s) / sizeof (\_\_u32))

#ifdef \_\_KERNEL\_\_

# define EXT2\_BLOCK\_SIZE\_BITS(s)    ((s)->s\_blocksize\_bits)

#else

# define EXT2\_BLOCK\_SIZE\_BITS(s)    ((s)->s\_log\_block\_size + 10)

#endif

#ifdef \_\_KERNEL\_\_

#define EXT2\_ADDR\_PER\_BLOCK\_BITS(s) ((s)->u.ext2\_sb.s\_addr\_per\_block\_bits)

#define EXT2\_INODE\_SIZE(s)      ((s)->u.ext2\_sb.s\_inode\_size)

#define EXT2\_FIRST\_INO(s)       ((s)->u.ext2\_sb.s\_first\_ino)

#else

#define EXT2\_INODE\_SIZE(s)  (((s)->s\_rev\_level == EXT2\_GOOD\_OLD\_REV) ? \

                 EXT2\_GOOD\_OLD\_INODE\_SIZE : \

                 (s)->s\_inode\_size)

#define EXT2\_FIRST\_INO(s)   (((s)->s\_rev\_level == EXT2\_GOOD\_OLD\_REV) ? \

                 EXT2\_GOOD\_OLD\_FIRST\_INO : \

                 (s)->s\_first\_ino)

#endif

/\*

\* Macro-instructions used to manage fragments

\*/

#define EXT2\_MIN\_FRAG\_SIZE      1024

#define EXT2\_MAX\_FRAG\_SIZE      4096

#define EXT2\_MIN\_FRAG\_LOG\_SIZE       10

#ifdef \_\_KERNEL\_\_

# define EXT2\_FRAG\_SIZE(s)      ((s)->u.ext2\_sb.s\_frag\_size)

# define EXT2\_FRAGS\_PER\_BLOCK(s)    ((s)->u.ext2\_sb.s\_frags\_per\_block)

#else

# define EXT2\_FRAG\_SIZE(s)      (EXT2\_MIN\_FRAG\_SIZE << (s)->s\_log\_frag\_size)

# define EXT2\_FRAGS\_PER\_BLOCK(s)    (EXT2\_BLOCK\_SIZE(s) / EXT2\_FRAG\_SIZE(s))

#endif

/\*

\* ACL structures

\*/

struct ext2\_acl\_header  /\* Header of Access Control Lists \*/

{

    \_\_u32   aclh\_size;

    \_\_u32   aclh\_file\_count;

    \_\_u32   aclh\_acle\_count;

    \_\_u32   aclh\_first\_acle;

};

struct ext2\_acl\_entry   /\* Access Control List Entry \*/

{

    \_\_u32   acle\_size;

    \_\_u16   acle\_perms; /\* Access permissions \*/

    \_\_u16   acle\_type;  /\* Type of entry \*/

    \_\_u16   acle\_tag;   /\* User or group identity \*/

    \_\_u16   acle\_pad1;

    \_\_u32   acle\_next;  /\* Pointer on next entry for the \*/

                    /\* same inode or on next free entry \*/

};

/\*

\* Structure of a blocks group descriptor

\*/

struct ext2\_group\_desc

{

    \_\_u32   bg\_block\_bitmap;        /\* Blocks bitmap block \*/

    \_\_u32   bg\_inode\_bitmap;        /\* Inodes bitmap block \*/

    \_\_u32   bg\_inode\_table;     /\* Inodes table block \*/

    \_\_u16   bg\_free\_blocks\_count;   /\* Free blocks count \*/

    \_\_u16   bg\_free\_inodes\_count;   /\* Free inodes count \*/

    \_\_u16   bg\_used\_dirs\_count; /\* Directories count \*/

    \_\_u16   bg\_pad;

    \_\_u32   bg\_reserved[3];

};

/\*

\* Macro-instructions used to manage group descriptors

\*/

#ifdef \_\_KERNEL\_\_

# define EXT2\_BLOCKS\_PER\_GROUP(s)   ((s)->u.ext2\_sb.s\_blocks\_per\_group)

# define EXT2\_DESC\_PER\_BLOCK(s)     ((s)->u.ext2\_sb.s\_desc\_per\_block)

# define EXT2\_INODES\_PER\_GROUP(s)   ((s)->u.ext2\_sb.s\_inodes\_per\_group)

# define EXT2\_DESC\_PER\_BLOCK\_BITS(s)    ((s)->u.ext2\_sb.s\_desc\_per\_block\_bits)

#else

# define EXT2\_BLOCKS\_PER\_GROUP(s)   ((s)->s\_blocks\_per\_group)

# define EXT2\_DESC\_PER\_BLOCK(s)     (EXT2\_BLOCK\_SIZE(s) / sizeof (struct ext2\_group\_desc))

# define EXT2\_INODES\_PER\_GROUP(s)   ((s)->s\_inodes\_per\_group)

#endif

/\*

\* Constants relative to the data blocks

\*/

#define EXT2\_NDIR\_BLOCKS        12

#define EXT2\_IND\_BLOCK          EXT2\_NDIR\_BLOCKS

#define EXT2\_DIND\_BLOCK         (EXT2\_IND\_BLOCK + 1)

#define EXT2\_TIND\_BLOCK         (EXT2\_DIND\_BLOCK + 1)

#define EXT2\_N\_BLOCKS           (EXT2\_TIND\_BLOCK + 1)

/\*

\* Inode flags

\*/

#define EXT2\_SECRM\_FL           0x00000001 /\* Secure deletion \*/

#define EXT2\_UNRM\_FL            0x00000002 /\* Undelete \*/

#define EXT2\_COMPR\_FL           0x00000004 /\* Compress file \*/

#define EXT2\_SYNC\_FL            0x00000008 /\* Synchronous updates \*/

#define EXT2\_IMMUTABLE\_FL       0x00000010 /\* Immutable file \*/

#define EXT2\_APPEND\_FL          0x00000020 /\* writes to file may only append \*/

#define EXT2\_NODUMP\_FL          0x00000040 /\* do not dump file \*/

#define EXT2\_RESERVED\_FL        0x80000000 /\* reserved for ext2 lib \*/

/\*

\* ioctl commands

\*/

#define EXT2\_IOC\_GETFLAGS       \_IOR('f', 1, long)

#define EXT2\_IOC\_SETFLAGS       \_IOW('f', 2, long)

#define EXT2\_IOC\_GETVERSION     \_IOR('v', 1, long)

#define EXT2\_IOC\_SETVERSION     \_IOW('v', 2, long)

/\*

\* Structure of an inode on the disk

\*/

struct ext2\_inode {

    \_\_u16   i\_mode;     /\* File mode \*/

    \_\_u16   i\_uid;      /\* Owner Uid \*/

    \_\_u32   i\_size;     /\* Size in bytes \*/

    \_\_u32   i\_atime;    /\* Access time \*/

    \_\_u32   i\_ctime;    /\* Creation time \*/

    \_\_u32   i\_mtime;    /\* Modification time \*/

    \_\_u32   i\_dtime;    /\* Deletion Time \*/

    \_\_u16   i\_gid;      /\* Group Id \*/

    \_\_u16   i\_links\_count;  /\* Links count \*/

    \_\_u32   i\_blocks;   /\* Blocks count \*/

    \_\_u32   i\_flags;    /\* File flags \*/

    union {

        struct {

            \_\_u32 l\_i\_reserved1;

        } linux1;

        struct {

            \_\_u32 h\_i\_translator;

        } hurd1;

        struct {

            \_\_u32 m\_i\_reserved1;

        } masix1;

    } osd1;             /\* OS dependent 1 \*/

    \_\_u32   i\_block[EXT2\_N\_BLOCKS];/\* Pointers to blocks \*/

    \_\_u32   i\_version;  /\* File version (for NFS) \*/

    \_\_u32   i\_file\_acl; /\* File ACL \*/

    \_\_u32   i\_dir\_acl;  /\* Directory ACL \*/

    \_\_u32   i\_faddr;    /\* Fragment address \*/

    union {

        struct {

            \_\_u8    l\_i\_frag;   /\* Fragment number \*/

            \_\_u8    l\_i\_fsize;  /\* Fragment size \*/

            \_\_u16   i\_pad1;

            \_\_u32   l\_i\_reserved2[2];

        } linux2;

        struct {

            \_\_u8    h\_i\_frag;   /\* Fragment number \*/

            \_\_u8    h\_i\_fsize;  /\* Fragment size \*/

            \_\_u16   h\_i\_mode\_high;

            \_\_u16   h\_i\_uid\_high;

            \_\_u16   h\_i\_gid\_high;

            \_\_u32   h\_i\_author;

        } hurd2;

        struct {

            \_\_u8    m\_i\_frag;   /\* Fragment number \*/

            \_\_u8    m\_i\_fsize;  /\* Fragment size \*/

            \_\_u16   m\_pad1;

            \_\_u32   m\_i\_reserved2[2];

        } masix2;

    } osd2;             /\* OS dependent 2 \*/

};

#if defined(\_\_KERNEL\_\_) || defined(\_\_linux\_\_)

#define i\_reserved1 osd1.linux1.l\_i\_reserved1

#define i\_frag      osd2.linux2.l\_i\_frag

#define i\_fsize     osd2.linux2.l\_i\_fsize

#define i\_reserved2 osd2.linux2.l\_i\_reserved2

#endif

#ifdef  \_\_hurd\_\_

#define i\_translator    osd1.hurd1.h\_i\_translator

#define i\_frag      osd2.hurd2.h\_i\_frag;

#define i\_fsize     osd2.hurd2.h\_i\_fsize;

#define i\_uid\_high  osd2.hurd2.h\_i\_uid\_high

#define i\_gid\_high  osd2.hurd2.h\_i\_gid\_high

#define i\_author    osd2.hurd2.h\_i\_author

#endif

#ifdef  \_\_masix\_\_

#define i\_reserved1 osd1.masix1.m\_i\_reserved1

#define i\_frag      osd2.masix2.m\_i\_frag

#define i\_fsize     osd2.masix2.m\_i\_fsize

#define i\_reserved2 osd2.masix2.m\_i\_reserved2

#endif

/\*

\* File system states

\*/

#define EXT2\_VALID\_FS           0x0001  /\* Unmounted cleanly \*/

#define EXT2\_ERROR\_FS           0x0002  /\* Errors detected \*/

/\*

\* Mount flags

\*/

#define EXT2\_MOUNT\_CHECK\_NORMAL     0x0001  /\* Do some more checks \*/

#define EXT2\_MOUNT\_CHECK\_STRICT     0x0002  /\* Do again more checks \*/

#define EXT2\_MOUNT\_CHECK        (EXT2\_MOUNT\_CHECK\_NORMAL | \

                     EXT2\_MOUNT\_CHECK\_STRICT)

#define EXT2\_MOUNT\_GRPID        0x0004  /\* Create files with directory's group \*/

#define EXT2\_MOUNT\_DEBUG        0x0008  /\* Some debugging messages \*/

#define EXT2\_MOUNT\_ERRORS\_CONT      0x0010  /\* Continue on errors \*/

#define EXT2\_MOUNT\_ERRORS\_RO        0x0020  /\* Remount fs ro on errors \*/

#define EXT2\_MOUNT\_ERRORS\_PANIC     0x0040  /\* Panic on errors \*/

#define EXT2\_MOUNT\_MINIX\_DF     0x0080  /\* Mimics the Minix statfs \*/

#define clear\_opt(o, opt)       o &= ~EXT2\_MOUNT\_##opt

#define set\_opt(o, opt)         o |= EXT2\_MOUNT\_##opt

#define test\_opt(sb, opt)       ((sb)->u.ext2\_sb.s\_mount\_opt & \

                     EXT2\_MOUNT\_##opt)

/\*

\* Maximal mount counts between two filesystem checks

\*/

#define EXT2\_DFL\_MAX\_MNT\_COUNT      20  /\* Allow 20 mounts \*/

#define EXT2\_DFL\_CHECKINTERVAL      0   /\* Don't use interval check \*/

/\*

\* Behaviour when detecting errors

\*/

#define EXT2\_ERRORS\_CONTINUE        1   /\* Continue execution \*/

#define EXT2\_ERRORS\_RO          2   /\* Remount fs read-only \*/

#define EXT2\_ERRORS\_PANIC       3   /\* Panic \*/

#define EXT2\_ERRORS\_DEFAULT     EXT2\_ERRORS\_CONTINUE

/\*

\* Structure of the super block

\*/

struct ext2\_super\_block {

    \_\_u32   s\_inodes\_count;     /\* Inodes count \*/

    \_\_u32   s\_blocks\_count;     /\* Blocks count \*/

    \_\_u32   s\_r\_blocks\_count;   /\* Reserved blocks count \*/

    \_\_u32   s\_free\_blocks\_count;    /\* Free blocks count \*/

    \_\_u32   s\_free\_inodes\_count;    /\* Free inodes count \*/

    \_\_u32   s\_first\_data\_block; /\* First Data Block \*/

    \_\_u32   s\_log\_block\_size;   /\* Block size \*/

    \_\_s32   s\_log\_frag\_size;    /\* Fragment size \*/

    \_\_u32   s\_blocks\_per\_group; /\* # Blocks per group \*/

    \_\_u32   s\_frags\_per\_group;  /\* # Fragments per group \*/

    \_\_u32   s\_inodes\_per\_group; /\* # Inodes per group \*/

    \_\_u32   s\_mtime;        /\* Mount time \*/

    \_\_u32   s\_wtime;        /\* Write time \*/

    \_\_u16   s\_mnt\_count;        /\* Mount count \*/

    \_\_s16   s\_max\_mnt\_count;    /\* Maximal mount count \*/

    \_\_u16   s\_magic;        /\* Magic signature \*/

    \_\_u16   s\_state;        /\* File system state \*/

    \_\_u16   s\_errors;       /\* Behaviour when detecting errors \*/

    \_\_u16   s\_minor\_rev\_level;  /\* minor revision level \*/

    \_\_u32   s\_lastcheck;        /\* time of last check \*/

    \_\_u32   s\_checkinterval;    /\* max. time between checks \*/

    \_\_u32   s\_creator\_os;       /\* OS \*/

    \_\_u32   s\_rev\_level;        /\* Revision level \*/

    \_\_u16   s\_def\_resuid;       /\* Default uid for reserved blocks \*/

    \_\_u16   s\_def\_resgid;       /\* Default gid for reserved blocks \*/

    /\*

     \* These fields are for EXT2\_DYNAMIC\_REV superblocks only.

     \*

     \* Note: the difference between the compatible feature set and

     \* the incompatible feature set is that if there is a bit set

     \* in the incompatible feature set that the kernel doesn't

     \* know about, it should refuse to mount the filesystem.

     \*

     \* e2fsck's requirements are more strict; if it doesn't know

     \* about a feature in either the compatible or incompatible

     \* feature set, it must abort and not try to meddle with

     \* things it doesn't understand...

     \*/

    \_\_u32   s\_first\_ino;        /\* First non-reserved inode \*/

    \_\_u16 s\_inode\_size;       /\* size of inode structure \*/

    \_\_u16   s\_block\_group\_nr;   /\* block group # of this superblock \*/

    \_\_u32   s\_feature\_compat;   /\* compatible feature set \*/

    \_\_u32   s\_feature\_incompat;     /\* incompatible feature set \*/

    \_\_u32   s\_feature\_ro\_compat;    /\* readonly-compatible feature set \*/

    \_\_u32   s\_reserved[230];    /\* Padding to the end of the block \*/

};

/\*

\* Codes for operating systems

\*/

#define EXT2\_OS\_LINUX       0

#define EXT2\_OS\_HURD        1

#define EXT2\_OS\_MASIX       2

#define EXT2\_OS\_FREEBSD     3

#define EXT2\_OS\_LITES       4

/\*

\* Revision levels

\*/

#define EXT2\_GOOD\_OLD\_REV   0   /\* The good old (original) format \*/

#define EXT2\_DYNAMIC\_REV    1   /\* V2 format w/ dynamic inode sizes \*/

#define EXT2\_CURRENT\_REV    EXT2\_GOOD\_OLD\_REV

#define EXT2\_MAX\_SUPP\_REV   EXT2\_DYNAMIC\_REV

#define EXT2\_GOOD\_OLD\_INODE\_SIZE 128

/\*

\* Default values for user and/or group using reserved blocks

\*/

#define EXT2\_DEF\_RESUID     0

#define EXT2\_DEF\_RESGID     0

/\*

\* Structure of a directory entry

\*/

#define EXT2\_NAME\_LEN 255

struct ext2\_dir\_entry {

    \_\_u32   inode;          /\* Inode number \*/

    \_\_u16   rec\_len;        /\* Directory entry length \*/

    \_\_u16   name\_len;       /\* Name length \*/

    char    name[EXT2\_NAME\_LEN];    /\* File name \*/

};

/\*

\* EXT2\_DIR\_PAD defines the directory entries boundaries

\*

\* NOTE: It must be a multiple of 4

\*/

#define EXT2\_DIR\_PAD            4

#define EXT2\_DIR\_ROUND          (EXT2\_DIR\_PAD - 1)

#define EXT2\_DIR\_REC\_LEN(name\_len)  (((name\_len) + 8 + EXT2\_DIR\_ROUND) & \

                     ~EXT2\_DIR\_ROUND)

/\*

\* Feature set definitions --- none are defined as of now

\*/

#define EXT2\_FEATURE\_COMPAT\_SUPP    0

#define EXT2\_FEATURE\_INCOMPAT\_SUPP  0

#define EXT2\_FEATURE\_RO\_COMPAT\_SUPP 0

#ifdef \_\_KERNEL\_\_

/\*

\* Function prototypes

\*/

/\*

\* Ok, these declarations are also in <linux/kernel.h> but none of the

\* ext2 source programs needs to include it so they are duplicated here.

\*/

# define NORET\_TYPE /\*\*/

# define ATTRIB\_NORET \_\_attribute\_\_((noreturn))

# define NORET\_AND noreturn,

/\* acl.c \*/

extern int ext2\_permission (struct inode \*, int);

/\* balloc.c \*/

extern int ext2\_new\_block (const struct inode \*, unsigned long,

             \_\_u32 \*, \_\_u32 \*, int \*);

extern void ext2\_free\_blocks (const struct inode \*, unsigned long,

             unsigned long);

extern unsigned long ext2\_count\_free\_blocks (struct super\_block \*);

extern void ext2\_check\_blocks\_bitmap (struct super\_block \*);

/\* bitmap.c \*/

extern unsigned long ext2\_count\_free (struct buffer\_head \*, unsigned);

/\* dir.c \*/

extern int ext2\_check\_dir\_entry (const char \*, struct inode \*,

                 struct ext2\_dir\_entry \*, struct buffer\_head \*,

                 unsigned long);

/\* file.c \*/

extern int ext2\_read (struct inode \*, struct file \*, char \*, int);

extern int ext2\_write (struct inode \*, struct file \*, char \*, int);

/\* fsync.c \*/

extern int ext2\_sync\_file (struct inode \*, struct file \*);

/\* ialloc.c \*/

extern struct inode \* ext2\_new\_inode (const struct inode \*, int, int \*);

extern void ext2\_free\_inode (struct inode \*);

extern unsigned long ext2\_count\_free\_inodes (struct super\_block \*);

extern void ext2\_check\_inodes\_bitmap (struct super\_block \*);

/\* inode.c \*/

extern int ext2\_bmap (struct inode \*, int);

extern struct buffer\_head \* ext2\_getblk (struct inode \*, long, int, int \*);

extern struct buffer\_head \* ext2\_bread (struct inode \*, int, int, int \*);

extern int ext2\_getcluster (struct inode \* inode, long block);

extern void ext2\_read\_inode (struct inode \*);

extern void ext2\_write\_inode (struct inode \*);

extern void ext2\_put\_inode (struct inode \*);

extern int ext2\_sync\_inode (struct inode \*);

extern void ext2\_discard\_prealloc (struct inode \*);

/\* ioctl.c \*/

extern int ext2\_ioctl (struct inode \*, struct file \*, unsigned int,

         unsigned long);

/\* namei.c \*/

extern void ext2\_release (struct inode \*, struct file \*);

extern int ext2\_lookup (struct inode \*,const char \*, int, struct inode \*\*);

extern int ext2\_create (struct inode \*,const char \*, int, int,

            struct inode \*\*);

extern int ext2\_mkdir (struct inode \*, const char \*, int, int);

extern int ext2\_rmdir (struct inode \*, const char \*, int);

extern int ext2\_unlink (struct inode \*, const char \*, int);

extern int ext2\_symlink (struct inode \*, const char \*, int, const char \*);

extern int ext2\_link (struct inode \*, struct inode \*, const char \*, int);

extern int ext2\_mknod (struct inode \*, const char \*, int, int, int);

extern int ext2\_rename (struct inode \*, const char \*, int,

            struct inode \*, const char \*, int, int);

/\* super.c \*/

extern void ext2\_error (struct super\_block \*, const char \*, const char \*, ...)

    \_\_attribute\_\_ ((format (printf, 3, 4)));

extern NORET\_TYPE void ext2\_panic (struct super\_block \*, const char \*,

                 const char \*, ...)

    \_\_attribute\_\_ ((NORET\_AND format (printf, 3, 4)));

extern void ext2\_warning (struct super\_block \*, const char \*, const char \*, ...)

    \_\_attribute\_\_ ((format (printf, 3, 4)));

extern void ext2\_put\_super (struct super\_block \*);

extern void ext2\_write\_super (struct super\_block \*);

extern int ext2\_remount (struct super\_block \*, int \*, char \*);

extern struct super\_block \* ext2\_read\_super (struct super\_block \*,void \*,int);

extern int init\_ext2\_fs(void);

extern void ext2\_statfs (struct super\_block \*, struct statfs \*, int);

/\* truncate.c \*/

extern void ext2\_truncate (struct inode \*);

/\*

\* Inodes and files operations

\*/

/\* dir.c \*/

extern struct inode\_operations ext2\_dir\_inode\_operations;

/\* file.c \*/

extern struct inode\_operations ext2\_file\_inode\_operations;

/\* symlink.c \*/

extern struct inode\_operations ext2\_symlink\_inode\_operations;

#endif  /\* \_\_KERNEL\_\_ \*/

#endif  /\* \_LINUX\_EXT2\_FS\_H \*/