```
/*
 * blk.h
 *
 * Note: pared down for USENIX example source code
 */

#ifndef _BLK_H
#define _BLK_H

#include <linux/blkdev.h>
#include <linux/locks.h>
#include <linux/config.h>

/*
 * NR_REQUEST is the number of entries in the request-queue.
 * NOTE that writes may use only the low 2/3 of these: reads
 * take precedence.
 */
#define NR_REQUEST      64

/*
 * This is used in the elevator algorithm.  We don't prioritise reads
 * over writes any more --- although reads are more time-critical than
 * writes, by treating them equally we increase filesystem throughput.
 * This turns out to give better overall performance.  -- sct
 */
#define IN_ORDER(s1,s2) \
((s1)->rq_dev < (s2)->rq_dev || (((s1)->rq_dev == (s2)->rq_dev && \
(s1)->sector < (s2)->sector)))

/*
 * These will have to be changed to be aware of different buffer
 * sizes etc.. It actually needs a major cleanup.
 */
#if defined(IDE_DRIVER) || defined(MD_DRIVER)
#define SECTOR_MASK ((BLOCK_SIZE >> 9) - 1)
#else
#define SECTOR_MASK (blksize_size[MAJOR_NR] &&     \
        blksize_size[MAJOR_NR][MINOR(CURRENT->rq_dev)] ? \
        ((blksize_size[MAJOR_NR][MINOR(CURRENT->rq_dev)] >> 9) - 1) :  \
        ((BLOCK_SIZE >> 9)  -  1))
#endif /* IDE_DRIVER */

#define SUBSECTOR(block) (CURRENT->current_nr_sectors > 0)

#ifdef CONFIG_BLK_DEV_HD
extern int hd_init(void);
#endif
#ifdef CONFIG_BLK_DEV_IDE
extern int ide_init(void);
#endif
#ifdef CONFIG_BLK_DEV_XD
extern int xd_init(void);
#endif
#ifdef CONFIG_BLK_DEV_LOOP
extern int loop_init(void);
#endif
#ifdef CONFIG_BLK_DEV_MD
extern int md_init(void);
#endif CONFIG_BLK_DEV_MD
```

```
extern void set_device_ro(kdev_t dev,int flag);
void add_blkdev_randomness(int major);

extern int floppy_init(void);
extern void rd_load(void);
extern int rd_init(void);
extern int rd_doload;           /* 1 = load ramdisk, 0 = don't load */
extern int rd_prompt;           /* 1 = prompt for ramdisk, 0 = don't prompt
*/
extern int rd_image_start;      /* starting block # of image */

#if defined(MAJOR_NR) || defined(IDE_DRIVER)

/*
 * Add entries as needed.
 */

#ifdef IDE_DRIVER

#define DEVICE_NR(device) (MINOR(device) >> PARTN_BITS)
#define DEVICE_ON(device) /* nothing */
#define DEVICE_OFF(device)        /* nothing */

#elif (MAJOR_NR == RAMDISK_MAJOR)

/* ram disk */
#define DEVICE_NAME "ramdisk"
#define DEVICE_REQUEST rd_request
#define DEVICE_NR(device) (MINOR(device))
#define DEVICE_ON(device)
#define DEVICE_OFF(device)
#define DEVICE_NO_RANDOM

#elif (MAJOR_NR == FLOPPY_MAJOR)

static void floppy_off(unsigned int nr);

#define DEVICE_NAME "floppy"
#define DEVICE_INTR do_floppy
#define DEVICE_REQUEST do_fd_request
#define DEVICE_NR(device) ( (MINOR(device) & 3) | \
                            ((MINOR(device) & 0x80 ) >> 5 ))
#define DEVICE_ON(device)
#define DEVICE_OFF(device) floppy_off(DEVICE_NR(device))

#elif (MAJOR_NR == HD_MAJOR)

/* harddisk: timeout is 6 seconds.. */
#define DEVICE_NAME "harddisk"
#define DEVICE_INTR do_hd
#define DEVICE_TIMEOUT HD_TIMER
#define TIMEOUT_VALUE (6*HZ)
#define DEVICE_REQUEST do_hd_request
#define DEVICE_NR(device) (MINOR(device)>>6)
#define DEVICE_ON(device)
#define DEVICE_OFF(device)

#elif (MAJOR_NR == SCSI_DISK_MAJOR)
```

```
#define DEVICE_NAME "scsidisk"
#define DEVICE_INTR do_sd
#define TIMEOUT_VALUE (2*HZ)
#define DEVICE_REQUEST do_sd_request
#define DEVICE_NR(device) (MINOR(device) >> 4)
#define DEVICE_ON(device)
#define DEVICE_OFF(device)

#elif (MAJOR_NR == SCSI_TAPE_MAJOR)

#define DEVICE_NAME "scsitape"
#define DEVICE_INTR do_st
#define DEVICE_NR(device) (MINOR(device) & 0x7f)
#define DEVICE_ON(device)
#define DEVICE_OFF(device)

#elif (MAJOR_NR == SCSI_CDROM_MAJOR)

#define DEVICE_NAME "CD-ROM"
#define DEVICE_INTR do_sr
#define DEVICE_REQUEST do_sr_request
#define DEVICE_NR(device) (MINOR(device))
#define DEVICE_ON(device)
#define DEVICE_OFF(device)

#elif (MAJOR_NR == XT_DISK_MAJOR)

#define DEVICE_NAME "xt disk"
#define DEVICE_REQUEST do_xd_request
#define DEVICE_NR(device) (MINOR(device) >> 6)
#define DEVICE_ON(device)
#define DEVICE_OFF(device)

#endif /* MAJOR_NR == whatever */

#if (MAJOR_NR != SCSI_TAPE_MAJOR)
#if !defined(IDE_DRIVER)

#ifndef CURRENT
#define CURRENT (blk_dev[MAJOR_NR].current_request)
#endif

#define CURRENT_DEV DEVICE_NR(CURRENT->rq_dev)

#ifdef DEVICE_INTR
static void (*DEVICE_INTR)(void) = NULL;
#endif
#ifdef DEVICE_TIMEOUT

#define SET_TIMER \
((timer_table[DEVICE_TIMEOUT].expires = jiffies + TIMEOUT_VALUE), \
(timer_active |= 1<<DEVICE_TIMEOUT))

#define CLEAR_TIMER \
timer_active &= ~(1<<DEVICE_TIMEOUT)

#define SET_INTR(x) \
if ((DEVICE_INTR = (x)) != NULL) \
        SET_TIMER; \
else \
```

```
        CLEAR_TIMER;

#else

#define SET_INTR(x) (DEVICE_INTR = (x))

#endif /* DEVICE_TIMEOUT */

static void (DEVICE_REQUEST)(void);

#ifdef DEVICE_INTR
#define CLEAR_INTR SET_INTR(NULL)
#else
#define CLEAR_INTR
#endif

#define INIT_REQUEST \
        if (!CURRENT) {\
                CLEAR_INTR; \
                return; \
        } \
        if (MAJOR(CURRENT->rq_dev) != MAJOR_NR) \
                panic(DEVICE_NAME ": request list destroyed"); \
        if (CURRENT->bh) { \
                if (!buffer_locked(CURRENT->bh)) \
                        panic(DEVICE_NAME ": block not locked"); \
        }

#endif /* !defined(IDE_DRIVER) */

/* end_request() - SCSI devices have their own version */
/*               - IDE drivers have their own copy too */

#if ! SCSI_BLK_MAJOR(MAJOR_NR)

/* shared copy for IDE modules */
#if defined(IDE_DRIVER) && !defined(_IDE_C)
void ide_end_request(byte uptodate, ide_hwgroup_t *hwgroup);
#else

#ifdef IDE_DRIVER
void ide_end_request(byte uptodate, ide_hwgroup_t *hwgroup) {
        struct request *req = hwgroup->rq;
#else
static void end_request(int uptodate) {
        struct request *req = CURRENT;
#endif /* IDE_DRIVER */
        struct buffer_head * bh;

        req->errors = 0;
        if (!uptodate) {
                printk("end_request: I/O error, dev %s, sector %lu\n",
                        kdevname(req->rq_dev), req->sector);
                req->nr_sectors--;
                req->nr_sectors &= ~SECTOR_MASK;
                req->sector += (BLOCK_SIZE / 512);
                req->sector &= ~SECTOR_MASK;
        }

        if ((bh = req->bh) != NULL) {
```

```
                req->bh = bh->b_reqnext;
                bh->b_reqnext = NULL;
                mark_buffer_uptodate(bh, uptodate);
                unlock_buffer(bh);
                if ((bh = req->bh) != NULL) {
                        req->current_nr_sectors = bh->b_size >> 9;
                        if (req->nr_sectors < req->current_nr_sectors) {
                                req->nr_sectors = req->current_nr_sectors;
                                printk("end_request: buffer-list
destroyed\n");
                        }
                        req->buffer = bh->b_data;
                        return;
                }
        }
#ifndef DEVICE_NO_RANDOM
        add_blkdev_randomness(MAJOR(req->rq_dev));
#endif
#ifdef IDE_DRIVER
        blk_dev[MAJOR(req->rq_dev)].current_request = req->next;
        hwgroup->rq = NULL;
#else
        DEVICE_OFF(req->rq_dev);
        CURRENT = req->next;
#endif /* IDE_DRIVER */
        if (req->sem != NULL)
                up(req->sem);
        req->rq_status = RQ_INACTIVE;
        wake_up(&wait_for_request);
}
#endif /* defined(IDE_DRIVER) && !defined(_IDE_C) */
#endif /* ! SCSI_BLK_MAJOR(MAJOR_NR) */
#endif /* (MAJOR_NR != SCSI_TAPE_MAJOR) */

#endif /* defined(MAJOR_NR) || defined(IDE_DRIVER) */

#endif /* _BLK_H */
```