```
/*
 * linux/drivers/char/misc.c
 *
 * Generic misc open routine by Johan Myreen
 *
 * Based on code from Linus
 *
 * Teemu Rantanen's Microsoft Busmouse support and Derrick Cole's
 *   changes incorporated into 0.97pl4
 *   by Peter Cervasio (pete%q106fm.uucp@wupost.wustl.edu) (08SEP92)
 *   See busmouse.c for particulars.
 *
 * Made things a lot mode modular - easy to compile in just one or two
 * of the misc drivers, as they are now completely independent. Linus.
 *
 * Support for loadable modules. 8-Sep-95 Philip Blundell <pjb27@cam.ac.uk>
 *
 * Fixed a failing symbol register to free the device registration
 *              Alan Cox <alan@lxorguk.ukuu.org.uk> 21-Jan-96
 *
 * Dynamic minors and /proc/mice by Alessandro Rubini. 26-Mar-96
 *
 * Renamed to misc and miscdevice to be more accurate. Alan Cox 26-Mar-96
 *
 * Handling of mouse minor numbers for kerneld:
 *  Idea by Jacques Gelinas <jack@solucorp.qc.ca>,
 *  adapted by Bjorn Ekwall <bj0rn@blox.se>
 *  corrected by Alan Cox <alan@lxorguk.ukuu.org.uk>
 */

#include <linux/config.h>
#include <linux/module.h>

#include <linux/fs.h>
#include <linux/errno.h>
#include <linux/miscdevice.h>
#include <linux/kernel.h>
#include <linux/major.h>
#include <linux/malloc.h>
#include <linux/proc_fs.h>
#include <linux/stat.h>
#ifdef CONFIG_APM
#include <linux/apm_bios.h>
#endif

#include <linux/tty.h> /* needed by selection.h */
#include "selection.h" /* export its symbols */
#ifdef CONFIG_KERNELD
#include <linux/kerneld.h>
#endif

/*
 * Head entry for the doubly linked miscdevice list
 */
static struct miscdevice misc_list = { 0, "head", NULL,
                                       &misc_list, &misc_list };

/*
 * Assigned numbers, used for dynamic minors
 */
```

```
#define DYNAMIC_MINORS 64 /* like dynamic majors */
static unsigned char misc_minors[DYNAMIC_MINORS / 8];

#ifndef MODULE
extern int bus_mouse_init(void);
extern int psaux_init(void);
extern int ms_bus_mouse_init(void);
extern int atixl_busmouse_init(void);
extern int sun_mouse_init(void);
extern void watchdog_init(void);
extern void pcwatchdog_init(void);
extern int rtc_init(void);

#ifdef CONFIG_PROC_FS
static int proc_misc_read(char *buf, char **start, off_t offset,
                          int len, int unused)
{
	struct miscdevice *p;

	len=0;
	for (p = misc_list.next; p != &misc_list; p = p->next)
		len += sprintf(buf+len, "%3i %s\n",p->minor, p->name ?: "");
	return len;
}

#endif /* PROC_FS */
#endif /* !MODULE */

static int misc_open(struct inode * inode, struct file * file)
{
	int minor = MINOR(inode->i_rdev);
	struct miscdevice *c = misc_list.next;
	file->f_op = NULL;

	while ((c != &misc_list) && (c->minor != minor))
		c = c->next;
	if (c == &misc_list) {
#ifdef CONFIG_KERNELD
		char modname[20];
		sprintf(modname, "char-major-%d-%d", MISC_MAJOR, minor);
		request_module(modname);
		c = misc_list.next;
		while ((c != &misc_list) && (c->minor != minor))
			c = c->next;
		if (c == &misc_list)
#endif
			return -ENODEV;
	}

	if ((file->f_op = c->fops))
		return file->f_op->open(inode,file);
	else
		return -ENODEV;
}

static struct file_operations misc_fops = {
	NULL,		/* seek */
	NULL,		/* read */
	NULL,		/* write */
	NULL,		/* readdir */
```

```
        NULL,           /* select */
        NULL,           /* ioctl */
        NULL,           /* mmap */
        misc_open,
        NULL            /* release */
};

int misc_register(struct miscdevice * misc)
{
        if (misc->next || misc->prev)
                return -EBUSY;
        if (misc->minor == MISC_DYNAMIC_MINOR) {
                int i = DYNAMIC_MINORS;
                while (--i >= 0)
                        if ( (misc_minors[i>>3] & (1 << (i&7))) == 0)
                                break;
                if (i<0) return -EBUSY;
                misc->minor = i;
        }
        if (misc->minor < DYNAMIC_MINORS)
                misc_minors[misc->minor >> 3] |= 1 << (misc->minor & 7);
        MOD_INC_USE_COUNT;
        misc->next = &misc_list;
        misc->prev = misc_list.prev;
        misc->prev->next = misc;
        misc->next->prev = misc;
        return 0;
}

int misc_deregister(struct miscdevice * misc)
{
        int i = misc->minor;
        if (!misc->next || !misc->prev)
                return -EINVAL;
        MOD_DEC_USE_COUNT;
        misc->prev->next = misc->next;
        misc->next->prev = misc->prev;
        misc->next = NULL;
        misc->prev = NULL;
        if (i < DYNAMIC_MINORS && i>0) {
                misc_minors[i>>3] &= ~(1 << (misc->minor & 7));
        }
        return 0;
}

#ifdef MODULE

#define misc_init init_module

void cleanup_module(void)
{
        unregister_chrdev(MISC_MAJOR, "misc");
}

#endif

static struct symbol_table misc_syms = {
/* Should this be surrounded with "#ifdef CONFIG_MODULES" ? */
#include <linux/symtab_begin.h>
        X(misc_register),
```

```
        X(misc_deregister),
#ifndef MODULE
        X(set_selection),   /* used by the kmouse module, can only */
        X(paste_selection), /* be exported if misc.c is in linked in */
#endif
#include <linux/symtab_end.h>
};

int misc_init(void)
{
#ifndef MODULE
#ifdef CONFIG_PROC_FS
        proc_register_dynamic(&proc_root, &(struct proc_dir_entry) {
                0, 4, "misc",
                S_IFREG | S_IRUGO, 1, 0, 0,
                0, NULL /* ops -- default to array */,
                &proc_misc_read /* get_info */,
        });
#endif /* PROC_FS */
#ifdef CONFIG_BUSMOUSE
        bus_mouse_init();
#endif
#if defined CONFIG_PSMOUSE
        psaux_init();
#endif
#ifdef CONFIG_MS_BUSMOUSE
        ms_bus_mouse_init();
#endif
#ifdef CONFIG_ATIXL_BUSMOUSE
        atixl_busmouse_init();
#endif
#ifdef CONFIG_AMIGAMOUSE
        amiga_mouse_init();
#endif
#ifdef CONFIG_ATARIMOUSE
        atari_mouse_init();
#endif
#ifdef CONFIG_SUN_MOUSE
        sun_mouse_init();
#endif
#ifdef CONFIG_SOFT_WATCHDOG
        watchdog_init();
#endif
#ifdef CONFIG_PCWATCHDOG
        pcwatchdog_init();
#endif
#ifdef CONFIG_APM
        apm_bios_init();
#endif
#ifdef CONFIG_RTC
        rtc_init();
#endif
#endif /* !MODULE */
        if (register_chrdev(MISC_MAJOR,"misc",&misc_fops)) {
          printk("unable to get major %d for misc devices\n",
                 MISC_MAJOR);
                return -EIO;
        }

        if(register_symtab(&misc_syms)!=0)
```

```
        {
                unregister_chrdev(MISC_MAJOR, "misc");
                return -EIO;
        }
        return 0;
}
```