```
/*
 *
 * magic8.c
 *
 * Author:      Michael Kepple
 * Date:        24 Apr 2013
 * Description: Implementation code for magic8 device driver.
 *
*/
#include <minix/drivers.h>
#include <minix/chardriver.h>
#include <stdio.h>
#include <stdlib.h>
#include <minix/ds.h>
#include <ctype.h>
#include <stdlib.h>
#include <time.h>
#include "magic8.h"

static int magic_open(message *m);
static int magic_close(message *m);
static int magic_ioctl(message *m);
static struct device * magic_prepare(dev_t device);
static int magic_transfer(endpoint_t endpt, int opcode, u64_t position,
        iovec_t *iov, unsigned int nr_req, endpoint_t user_endpt, unsigned int
        flags);
static void sef_local_startup(void);
static int sef_cb_init(int type, sef_init_info_t *info);
static struct chardriver magic_tab =
{
    magic_open,
    magic_close,
    magic_ioctl,
    magic_prepare,
    magic_transfer,
    nop_cleanup,
    nop_alarm,
    nop_cancel,
    nop_select,
    NULL
};
static struct device magic_device;

static int open_counter;

static int magic_open(message *UNUSED(m))
{
    int rand = random() % NUM_RESPONSES;
    char *message;
    if (SEQ == SEQUENTIAL)
        message = answers[open_counter%NUM_RESPONSES];
    else
        message = answers[rand];
    int i=0;
    switch (CASE)
    {
        case ORIGINAL:
            while (message[i] != '\0')
                printf("%c", message[i++]);
            break;
        case UPPER:
            while (message[i] != '\0')
                printf("%c", toupper(message[i++]));
```

```
                break;
            case LOWER:
                 while (message[i] != '\0')
                     printf("%c", tolower(message[i++]));
        }
        if (STOKE)
            printf("!!!");
        printf("\n");
        open_counter++;
        return OK;
    }

    static int magic_close(message *UNUSED(m))
    {
        return OK;
    }

    static int magic_ioctl(message *m)
    {
        switch (m->TTY_REQUEST)
        {
            case SET_ORIG:
                CASE = ORIGINAL;
                break;
            case SET_UPPER:
                CASE = UPPER;
                break;
            case SET_LOWER:
                CASE = LOWER;
                break;
            case SET_SEQUENTIAL:
                SEQ = SEQUENTIAL;
                break;
            case SET_RANDOM:
                SEQ = RANDOM;
                break;
            case SET_STOKED:
                STOKE = STOKED;
                break;
            case SET_NOT_STOKED:
                STOKE = NOT_STOKED;
                break;
        }
        return OK;
    }

    static struct device * magic_prepare(dev_t UNUSED(dev))
    {
        magic_device.dv_base = make64(0, 0);
        return &magic_device;
    }

    static int magic_transfer(endpoint_t endpt, int opcode, u64_t position,
        iovec_t *iov, unsigned nr_req, endpoint_t UNUSED(user_endpt),
        unsigned int UNUSED(flags))
    {
        return OK;
    }

    static void sef_local_startup()
    {
        sef_setcb_init_fresh(sef_cb_init);
        sef_setcb_init_lu(sef_cb_init);
```

```
        sef_setcb_init_restart(sef_cb_init);
        sef_setcb_lu_prepare(sef_cb_lu_prepare_always_ready);
        sef_setcb_lu_state_isvalid(sef_cb_lu_state_isvalid_standard);
        sef_startup();
    }

    static int sef_cb_init(int type, sef_init_info_t *UNUSED(info))
    {
        CASE = ORIGINAL;
        SEQ = SEQUENTIAL;
        STOKE = NOT_STOKED;
        srandom(time(NULL));
        open_counter = 0;
        return OK;
    }

    int main(void)
    {
        sef_local_startup();
        chardriver_task(&magic_tab, CHARDRIVER_SYNC);
        return OK;
    }
    /*
     * File:   magic8.h
     * Author: Michael Kepple
     * Date:   24 Apr 2013
    */
    #ifndef __MAGIC8_H
    #define __MAGIC8_H
    #define NUM_RESPONSES 5
    #define SET_SEQUENTIAL 0
    #define SET_RANDOM 1
    #define SET_ORIG 2
    #define SET_UPPER 3
    #define SET_LOWER 4
    #define SET_STOKED 5
    #define SET_NOT_STOKED 6
    #define SEQUENTIAL 0
    #define RANDOM 1
    #define ORIGINAL 0
    #define UPPER 1
    #define LOWER 2
    #define NOT_STOKED 0
    #define STOKED 1
    int CASE;
    int SEQ;
    int STOKE;
    char *answers[5] = {"0 No", "1 Yes", "2 Maybe", "3 Not a chance in hell",
        "4 Ab-so-lutely"};
    #endif
    #
    # Makefile for magic8 driver
    # Author: Michael Kepple
    # Date:   24 Apr 2013
    #
    PROG=   magic8
    SRCS=   magic8.c

    DPADD+= ${LIBCHARDRIVER} ${LIBSYS}
    LDADD+= -lchardriver -lsys

    MAN=
```

```
    BINDIR?= /usr/sbin

    .include <minix.service.mk>
    # Results file for Magic8 driver testing
    # Author: Michael Kepple
    # Date:   24 Apr 2013

    Testing Sequential/Original Case:
    0 No
    1 Yes
    2 Maybe
    3 Not a chance in hell
    4 Ab-so-lutely
    Testing Random/Uppercase:
    1 YES
    2 YES
    4 AB-SO-LUTELY
    2 MAYBE
    0 NO
    Testing Random/Lowercare/Stoked:
    3 no a chance in hell!!!
    4 ab-so-lutely!!!
    0 no!!!
    4 ab-so-lutely!!!
    1 yes!!!
    /*
     *
     * test.c
     *
     * Author:      Michael Kepple
     * Date:        24 Apr 2013
     * Description: test file for magic8 driver functionality.
     *
    */
    #include <stdio.h>
    #include <stdlib.h>
    #include <sys/ioctl.h>
    #include <sys/types.h>
    #include <fcntl.h>
    #include <ctype.h>
    #include "magic8.h"

    int main()
    {
        int fd;
        printf("Testing Sequential/Original Case:\n");
        fd = open("/dev/magic8", 0);
        system("cat /dev/magic8");
        system("cat /dev/magic8");
        system("cat /dev/magic8");
        system("cat /dev/magic8");
        printf("Testing Random/Uppercase:\n");
        ioctl(fd, SET_RANDOM, NULL);
        ioctl(fd, SET_UPPER, NULL);
        system("cat /dev/magic8");
        system("cat /dev/magic8");
        system("cat /dev/magic8");
        system("cat /dev/magic8");
        system("cat /dev/magic8");
        printf("Testing Random/Lowercase/Stoked:\n");
        ioctl(fd, SET_LOWER, NULL);
        ioctl(fd, SET_STOKED, NULL);
        system("cat /dev/magic8");
```

```
        system("cat /dev/magic8");
        system("cat /dev/magic8");
        system("cat /dev/magic8");
        system("cat /dev/magic8");
        ioctl(fd, SET_ORIG, NULL);
        ioctl(fd, SET_SEQUENTIAL, NULL);
        ioctl(fd, SET_NOT_STOKED, NULL);
    }
```