#include <linux/kernel.h> // for kernel programming

#include <linux/module.h> // for kernel module struct.

#include <linux/fs.h> // struct file\_operations

#include <linux/uaccess.h>

#define RWBUF\_CLEAR 0x909090

#define RW\_CLEAR 123

#define DEVICE\_NAME "rwbuf"

#define rwbuf\_size 1024 // MAX size of buffer

static char rwbuf[rwbuf\_size] = "18130500001"; // the buffer keeping string

static int rwlen = 0; // length of string

static int rwbuf\_open ( struct inode \*inode, struct file \* filep );

static int rwbuf\_close( struct inode \*inode, struct file \* filep );

static ssize\_t rwbuf\_read( struct file\* filep, char \* buf, size\_t count, loff\_t\* ppos);

static long rwbuf\_ioctl ( struct file \* filep,

unsigned int cmd, unsigned long arg );

static ssize\_t rwbuf\_write ( struct file \* filep, const char \*buf,

size\_t count, loff\_t \* ppos );

static struct file\_operations rwbuf\_fops =

{

open: rwbuf\_open,

release: rwbuf\_close,

read: rwbuf\_read,

write: rwbuf\_write,

unlocked\_ioctl: rwbuf\_ioctl,

};

static ssize\_t rwbuf\_read( struct file\* filep, char \* buf, size\_t count, loff\_t\* ppos)

{

if(count > rwbuf\_size || count <0)

{

printk("error input");

return 0;

}

// 判断读取的长度是否有效

copy\_to\_user( buf, rwbuf, count); // 从内核空间复制到用户空间

// print some message by printk()

return count;

}

static long rwbuf\_ioctl ( struct file \* filep,

unsigned int cmd, unsigned long arg )

{

if ( cmd == RWBUF\_CLEAR ) {

rwlen = 0; // clear buf by set its len to zero

rwbuf[0] = '\0';

printk("rwbuf in kernel zero-ed\n");

};

return 0;

}

static ssize\_t rwbuf\_write ( struct file \* filep, const char \*buf,

size\_t count, loff\_t \* ppos )

{

// 判断写入的长度是否有效

copy\_from\_user(rwbuf, buf, count); // 从用户空间复制到内核空间

rwlen = count;

// print some message by printk();

return count;

}

static int inuse=0; // only one process permited at the same time

static int rwbuf\_open ( struct inode \*inode, struct file \* filep ) {

if (inuse == 1) return -1;

inuse = 1;

try\_module\_get(THIS\_MODULE);

return 0;

}

static int rwbuf\_close( struct inode \*inode, struct file \* filep ) {

inuse = 0;

module\_put(THIS\_MODULE);

return 0;

}

int init\_rwbuf(void)

{

printk("Hello world\n");

if ( register\_chrdev(60, DEVICE\_NAME, &rwbuf\_fops))

{ printk("register error\n");

return -1;

}

printk("register ok\n");

return 0;

}

void cleanup\_rwbuf(void)

{

unregister\_chrdev( 60, DEVICE\_NAME);

printk("bye\n");

}

MODULE\_LICENSE("GPL");

module\_init(init\_rwbuf);

module\_exit(cleanup\_rwbuf);