```c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/kern_levels.h>
#include <linux/uaccess.h>
#include <linux/fs.h>
#include <linux/ioctl.h>
#include <linux/random.h>


MODULE_LICENSE("GPL");

#define ECE_BUF_SIZE 256
#define WORD_SIZE 5
#define IOCTL_RESET_GAME _IO('W', 1)

static const char *word_list[] = {
    "apple", "grape", "berry", "lemon", "melon",
    "brick", "chair", "cloud", "dance", "eagle",
    "flame", "giant", "horse", "index", "jelly",
    "knife", "light", "magic", "noble", "ocean",
    "plant", "queen", "rider", "smile", "table",
    "ultra", "vivid", "wheat", "xenon", "yield",
    "zebra", "blush", "crisp", "drink", "elite",
    "frost", "globe", "honey", "ivory", "jolly",
    "karma", "lunar", "mango", "nerdy", "orbit",
    "pearl", "quake", "reign", "spice", "toast",
    "unite", "vapor", "whirl", "stage", "young",
    "zesty", "actor", "beach", "cabin", "dream",
    "early", "fable", "grind", "haste", "ideal",
    "jewel", "koala", "latch", "birth", "nudge",
    "optic", "plush", "quart", "risky", "skate",
    "trick", "uncle", "vigor", "wound", "field",
    "yacht", "float", "alien", "blaze", "crown",
    "dizzy", "exile", "frown", "gleam", "haunt",
    "inbox", "jumps", "kneel", "liver", "manor",
    "ninth", "oxide", "piano", "quiet", "rural"
};

#define WORD_LIST_SIZE (sizeof(word_list) / sizeof(word_list[0]))


static char ece_buffer[ECE_BUF_SIZE];
static char target_word[WORD_SIZE + 1] = "apple";
int isReg;
int major;
int ece_offset_w;
int ece_offset_r;
int ece_size;

int ece_init(void);
void ece_end(void);
static ssize_t ece_write(struct file*, const char*, size_t, loff_t*);
static ssize_t ece_read(struct file*, char*, size_t, loff_t*);
static long ece_ioctl(struct file *file, unsigned int cmd, unsigned long arg);

static struct file_operations ece_fops =
{
    .read = ece_read,
    .write = ece_write,
    .unlocked_ioctl = ece_ioctl,
};

int ece_init(void)
{
    major = register_chrdev(0, "Seminar5", &ece_fops);
    ece_offset_w = 0;
    ece_offset_r = 0;
    ece_size = 0;

    if (major < 0)
    {
        isReg = 0;
        printk(KERN_INFO "ECE4310: Start FAIL \n");
    } else
    {
        isReg = 1;
        printk(KERN_INFO "ECE4310: Start here \n");
        printk(KERN_INFO "ECE4310: Major number = %d\n", major);
    }
    return 0;
}

void ece_end(void)
{
    if (isReg)
    {
        unregister_chrdev(major, "Seminar5");
    }
    printk(KERN_INFO "ECE4310: End here \n");
}

static ssize_t ece_write(struct file *fp, const char *buf, size_t count, loff_t
*op)
{
    char guess[WORD_SIZE + 1] = {0};
    char hint[WORD_SIZE + 1] = {0};
    int matched[WORD_SIZE] = {0};
    int i, j;

    if (copy_from_user(guess, buf, WORD_SIZE))
        return -1;

    for (i = 0; i < WORD_SIZE; i++)
    {
        if (guess[i] == target_word[i])
        {
            hint[i] = guess[i];
            matched[i] = 1;
        } else
        {
            hint[i] = '_';
        }
    }

    for (i = 0; i < WORD_SIZE; i++)
    {
        if (hint[i] == '_')
        {
            for (j = 0; j < WORD_SIZE; j++)
            {
                if (!matched[j] && guess[i] == target_word[j])
                {
                    hint[i] = '?';
                    matched[j] = 1;
                    break;
                }
            }
        }
    }

    memset(ece_buffer, 0, ECE_BUF_SIZE);
    snprintf(ece_buffer, ECE_BUF_SIZE, "%s", hint);
    ece_offset_r = 0;

    return count;
}

static ssize_t ece_read(struct file *fp, char *buf, size_t count, loff_t *offset)
{
    int bytes_left = strlen(ece_buffer) - ece_offset_r;
    int to_copy = min((int)count, bytes_left);

    if (to_copy <= 0)
        return 0;

    printk(KERN_INFO "ECE4310: Copy to user for Wordle hint.\n");

    if (copy_to_user(buf, ece_buffer + ece_offset_r, to_copy))
        return -1;

    ece_offset_r += to_copy;
    return to_copy;
}

static long ece_ioctl(struct file *fp, unsigned int cmd, unsigned long arg)
{
    switch (cmd)
    {
        case IOCTL_RESET_GAME:
        {
            unsigned int rand_index;
            get_random_bytes(&rand_index, sizeof(rand_index));
            rand_index = rand_index % WORD_LIST_SIZE;

            strncpy(target_word, word_list[rand_index], WORD_SIZE);
            printk(KERN_INFO "ECE4310: Game reset to word: %s\n", target_word);
            return 0;
        }
        default:
            return -1;
    }
}

module_init(ece_init);
module_exit(ece_end);
```