```
#include <linux/module.h>
#include <linux/fs.h>
#include <linux/proc_fs.h>
#include <linux/uaccess.h>

#define DEVICE_NAME "sys_metrics"
#define BUF_LEN 1024

static int device_open(struct inode *, struct file *);
static int device_release(struct inode *, struct file *);
static ssize_t device_read(struct file *, char *, size_t, loff_t *);

static int major;
static char msg[BUF_LEN];
static int msg_len;

static struct file_operations fops = {
    .read = device_read,
    .open = device_open,
    .release = device_release
};

static int __init sys_metrics_init(void) {
    major = register_chrdev(0, DEVICE_NAME, &fops);
    if (major < 0) {
        printk(KERN_ALERT "Registering char device failed with %d\n", major);
        return major;
    }
    printk(KERN_INFO "Device registered with major number %d\n", major);
    return 0;
}

static void __exit sys_metrics_exit(void) {
    unregister_chrdev(major, DEVICE_NAME);
}

static int device_open(struct inode *inode, struct file *file) {
    sprintf(msg, "CPU Usage: %%\nMemory Usage: %%\n"); // Example placeholders
    msg_len = strlen(msg);
    return 0;
}

static ssize_t device_read(struct file *filp, char *buffer, size_t length, loff_t * offset) {
    int bytes_read = 0;
    if (*offset >= msg_len) return 0;

    while (length && *offset < msg_len) {
        put_user(msg[*offset], buffer++);
```

```
        length--;
        (*offset)++;
        bytes_read++;
    }
    return bytes_read;
}

static int device_release(struct inode *inode, struct file *file) {
    return 0;
}

module_init(sys_metrics_init);
module_exit(sys_metrics_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Your Name");
MODULE_DESCRIPTION("A simple device driver to print system metrics");
```