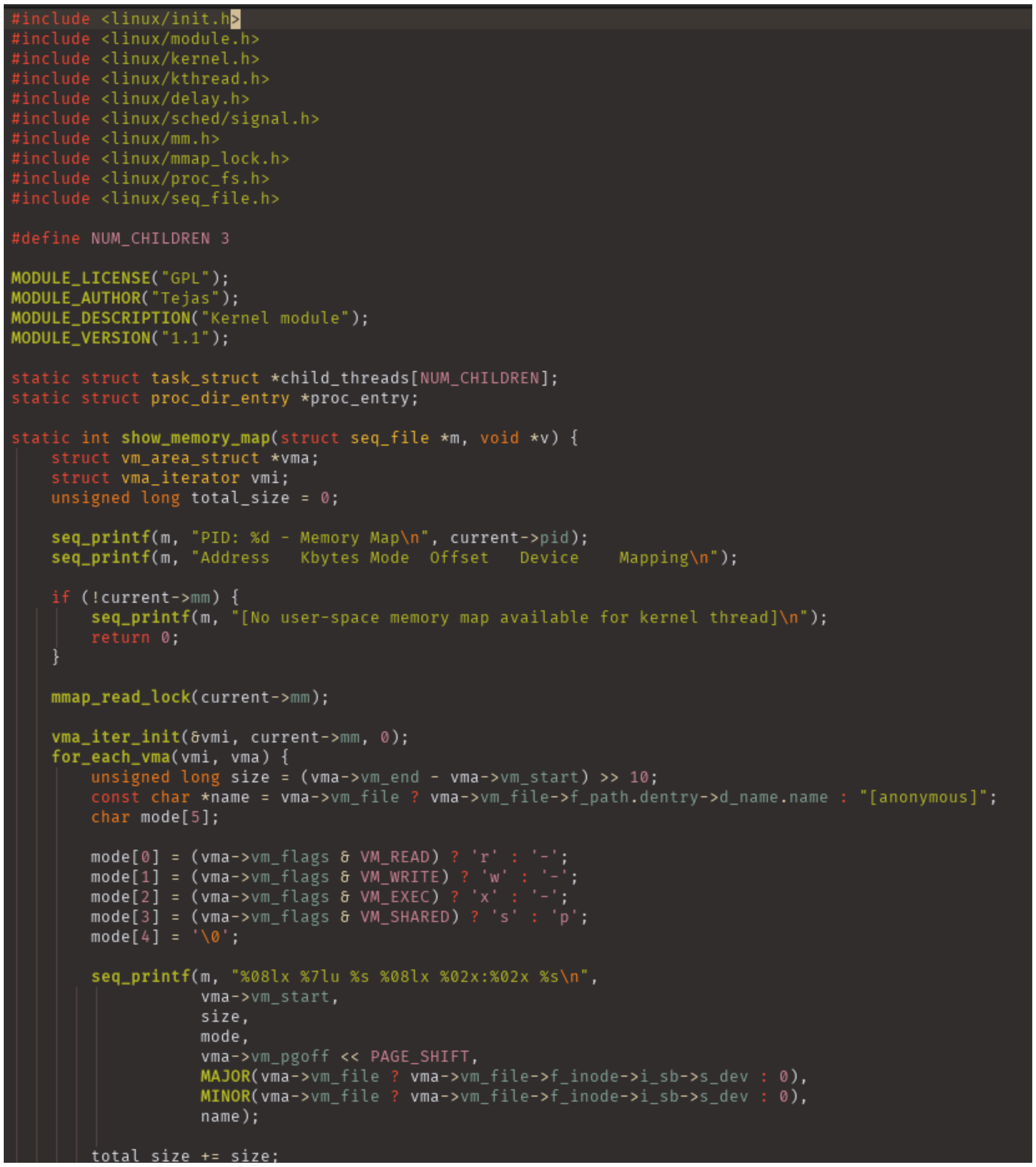
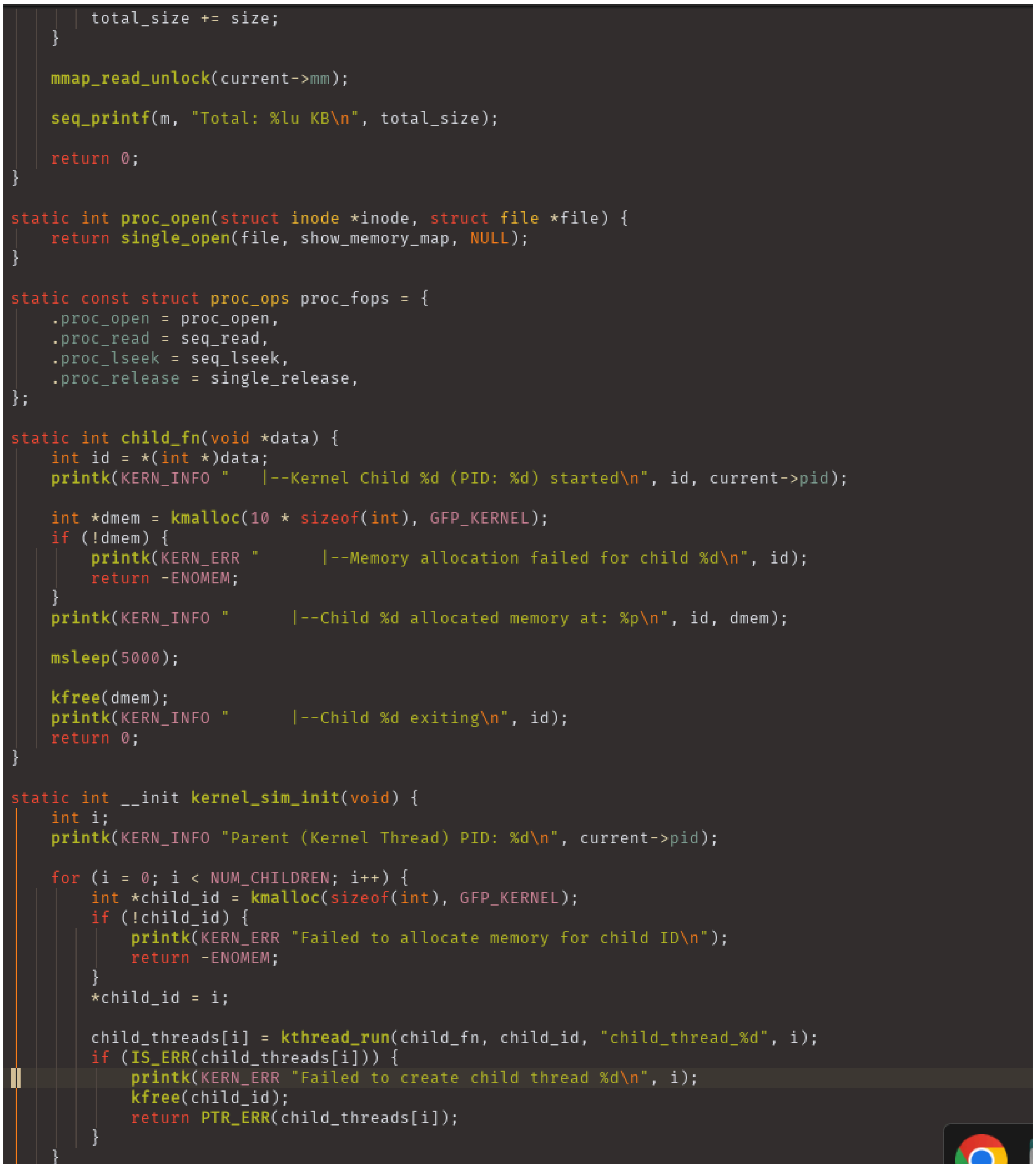
OS JACKFRUIT PROBLEM

Name : Tejas Ramakrishnan
SRN : PES2UG23CS648
Section : J

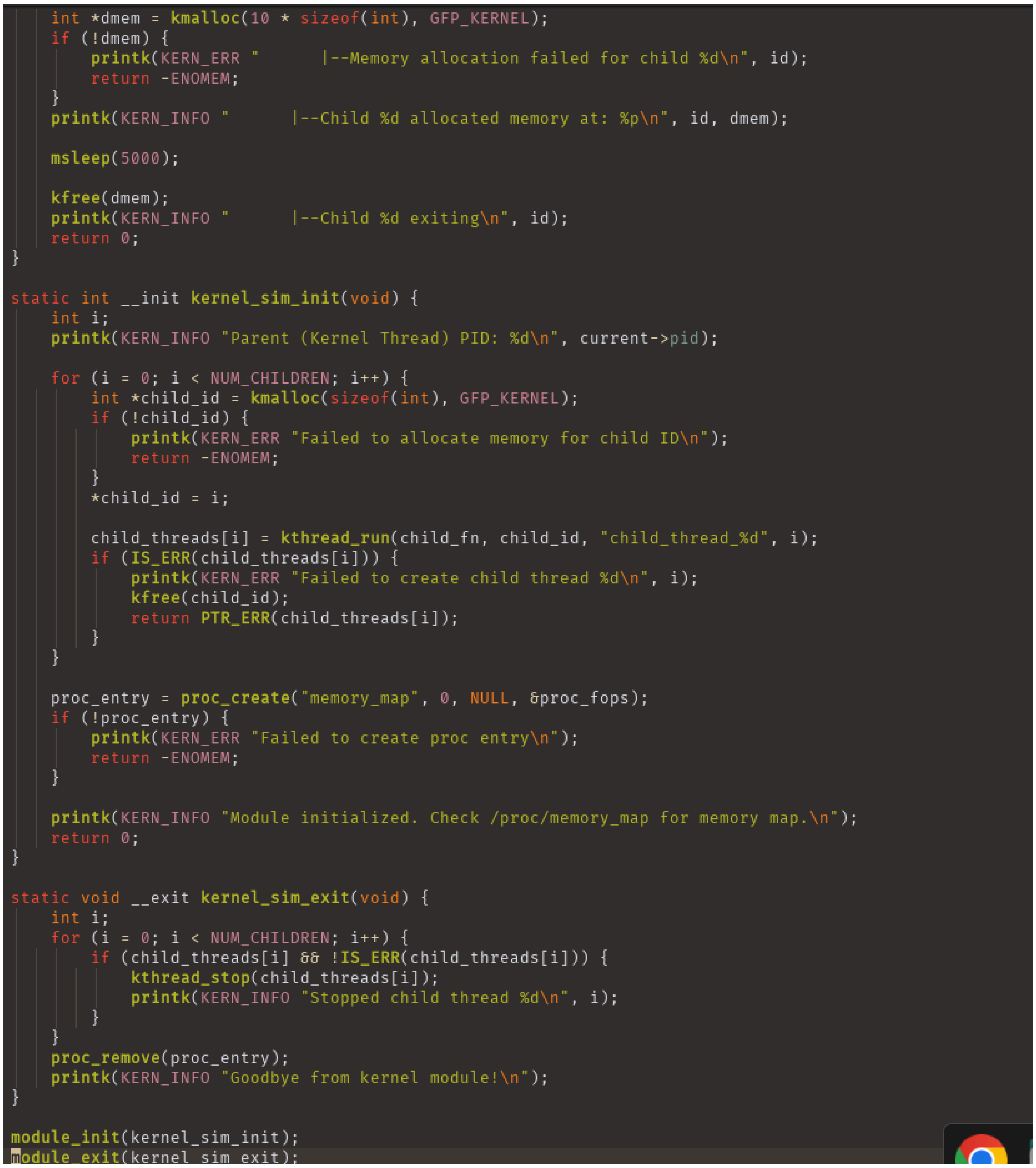
Kernel Module :
https://drive.google.com/file/d/16enx4VvxtcxfdVCD_DC-NEphIW4EaHQh/view?usp=sharin
C FILE :



```c
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/kthread.h>
#include <linux/delay.h>
#include <linux/sched/signal.h>
#include <linux/mm.h>
#include <linux/mmap_lock.h>
#include <linux/proc_fs.h>
#include <linux/seq_file.h>

#define NUM_CHILDREN 3

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Tejas");
MODULE_DESCRIPTION("Kernel module");
MODULE_VERSION("1.1");

static struct task_struct *child_threads[NUM_CHILDREN];
static struct proc_dir_entry *proc_entry;

static int show_memory_map(struct seq_file *m, void *v) {
    struct vm_area_struct *vma;
    struct vma_iterator vmi;
    unsigned long total_size = 0;

    seq_printf(m, "PID: %d - Memory Map\n", current->pid);
    seq_printf(m, "Address   Kbytes Mode  Offset   Device    Mapping\n");

    if (!current->mm) {
        seq_printf(m, "[No user-space memory map available for kernel thread]\n");
        return 0;
    }

    mmap_read_lock(current->mm);

    vma_iter_init(&vmi, current->mm, 0);
    for_each_vma(vmi, vma) {
        unsigned long size = (vma->vm_end - vma->vm_start) >> 10;
        const char *name = vma->vm_file ? vma->vm_file->f_path.dentry->d_name.name : "[anonymous]";
        char mode[5];

        mode[0] = (vma->vm_flags & VM_READ) ? 'r' : '-';
        mode[1] = (vma->vm_flags & VM_WRITE) ? 'w' : '-';
        mode[2] = (vma->vm_flags & VM_EXEC) ? 'x' : '-';
        mode[3] = (vma->vm_flags & VM_SHARED) ? 's' : 'p';
        mode[4] = '\0';

        seq_printf(m, "%08lx %7lu %s %08lx %02x:%02x %s\n",
                   vma->vm_start,
                   size,
                   mode,
                   vma->vm_pgoff << PAGE_SHIFT,
                   MAJOR(vma->vm_file ? vma->vm_file->f_inode->i_sb->s_dev : 0),
                   MINOR(vma->vm_file ? vma->vm_file->f_inode->i_sb->s_dev : 0),
                   name);

        total_size += size;
```

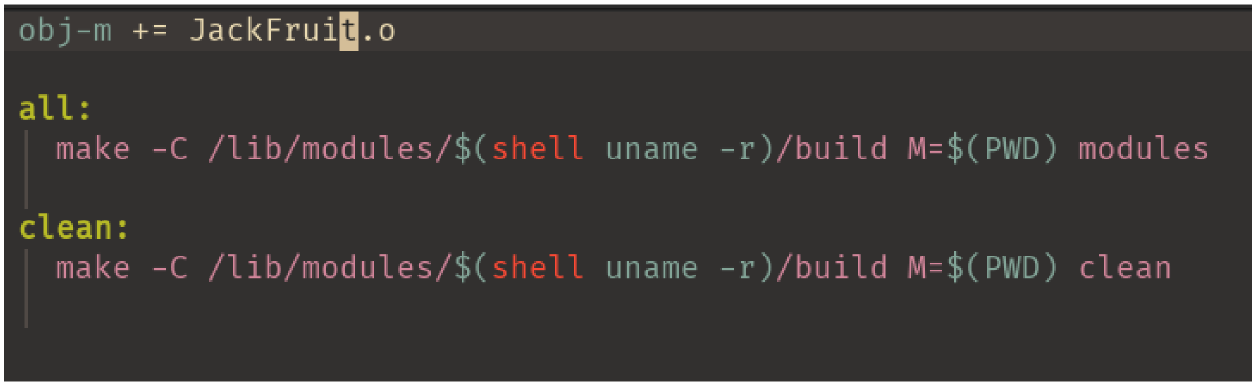


```c
        total_size += size;
    }

    mmap_read_unlock(current->mm);

    seq_printf(m, "Total: %lu KB\n", total_size);

    return 0;
}

static int proc_open(struct inode *inode, struct file *file) {
    return single_open(file, show_memory_map, NULL);
}

static const struct proc_ops proc_fops = {
    .proc_open = proc_open,
    .proc_read = seq_read,
    .proc_lseek = seq_lseek,
    .proc_release = single_release,
};

static int child_fn(void *data) {
    int id = *(int *)data;
    printk(KERN_INFO "   |--Kernel Child %d (PID: %d) started\n", id, current->pid);

    int *dmem = kmalloc(10 * sizeof(int), GFP_KERNEL);
    if (!dmem) {
        printk(KERN_ERR "      |--Memory allocation failed for child %d\n", id);
        return -ENOMEM;
    }
    printk(KERN_INFO "      |--Child %d allocated memory at: %p\n", id, dmem);

    msleep(5000);

    kfree(dmem);
    printk(KERN_INFO "      |--Child %d exiting\n", id);
    return 0;
}

static int __init kernel_sim_init(void) {
    int i;
    printk(KERN_INFO "Parent (Kernel Thread) PID: %d\n", current->pid);

    for (i = 0; i < NUM_CHILDREN; i++) {
        int *child_id = kmalloc(sizeof(int), GFP_KERNEL);
        if (!child_id) {
            printk(KERN_ERR "Failed to allocate memory for child ID\n");
            return -ENOMEM;
        }
        *child_id = i;

        child_threads[i] = kthread_run(child_fn, child_id, "child_thread_%d", i);
        if (IS_ERR(child_threads[i])) {
            printk(KERN_ERR "Failed to create child thread %d\n", i);
            kfree(child_id);
            return PTR_ERR(child_threads[i]);
        }
    }
```



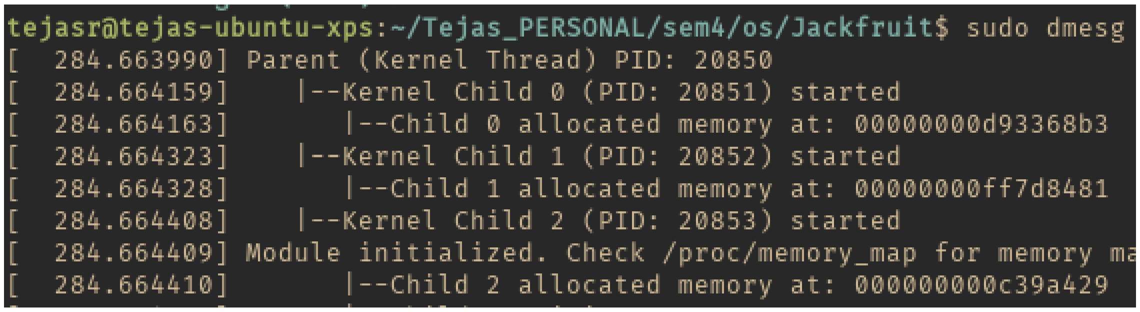
```
    int *dmem = kmalloc(10 * sizeof(int), GFP_KERNEL);
    if (!dmem) {
        printk(KERN_ERR "      |--Memory allocation failed for child %d\n", id);
        return -ENOMEM;
    }
    printk(KERN_INFO "      |--Child %d allocated memory at: %p\n", id, dmem);

    msleep(5000);

    kfree(dmem);
    printk(KERN_INFO "      |--Child %d exiting\n", id);
    return 0;
}

static int __init kernel_sim_init(void) {
    int i;
    printk(KERN_INFO "Parent (Kernel Thread) PID: %d\n", current->pid);

    for (i = 0; i < NUM_CHILDREN; i++) {
        int *child_id = kmalloc(sizeof(int), GFP_KERNEL);
        if (!child_id) {
            printk(KERN_ERR "Failed to allocate memory for child ID\n");
            return -ENOMEM;
        }
        *child_id = i;

        child_threads[i] = kthread_run(child_fn, child_id, "child_thread_%d", i);
        if (IS_ERR(child_threads[i])) {
            printk(KERN_ERR "Failed to create child thread %d\n", i);
            kfree(child_id);
            return PTR_ERR(child_threads[i]);
        }
    }

    proc_entry = proc_create("memory_map", 0, NULL, &proc_fops);
    if (!proc_entry) {
        printk(KERN_ERR "Failed to create proc entry\n");
        return -ENOMEM;
    }

    printk(KERN_INFO "Module initialized. Check /proc/memory_map for memory map.\n");
    return 0;
}

static void __exit kernel_sim_exit(void) {
    int i;
    for (i = 0; i < NUM_CHILDREN; i++) {
        if (child_threads[i] && !IS_ERR(child_threads[i])) {
            kthread_stop(child_threads[i]);
            printk(KERN_INFO "Stopped child thread %d\n", i);
        }
    }
    proc_remove(proc_entry);
    printk(KERN_INFO "Goodbye from kernel module!\n");
}

module_init(kernel_sim_init);
module_exit(kernel_sim_exit);
```

MAKEFILE :

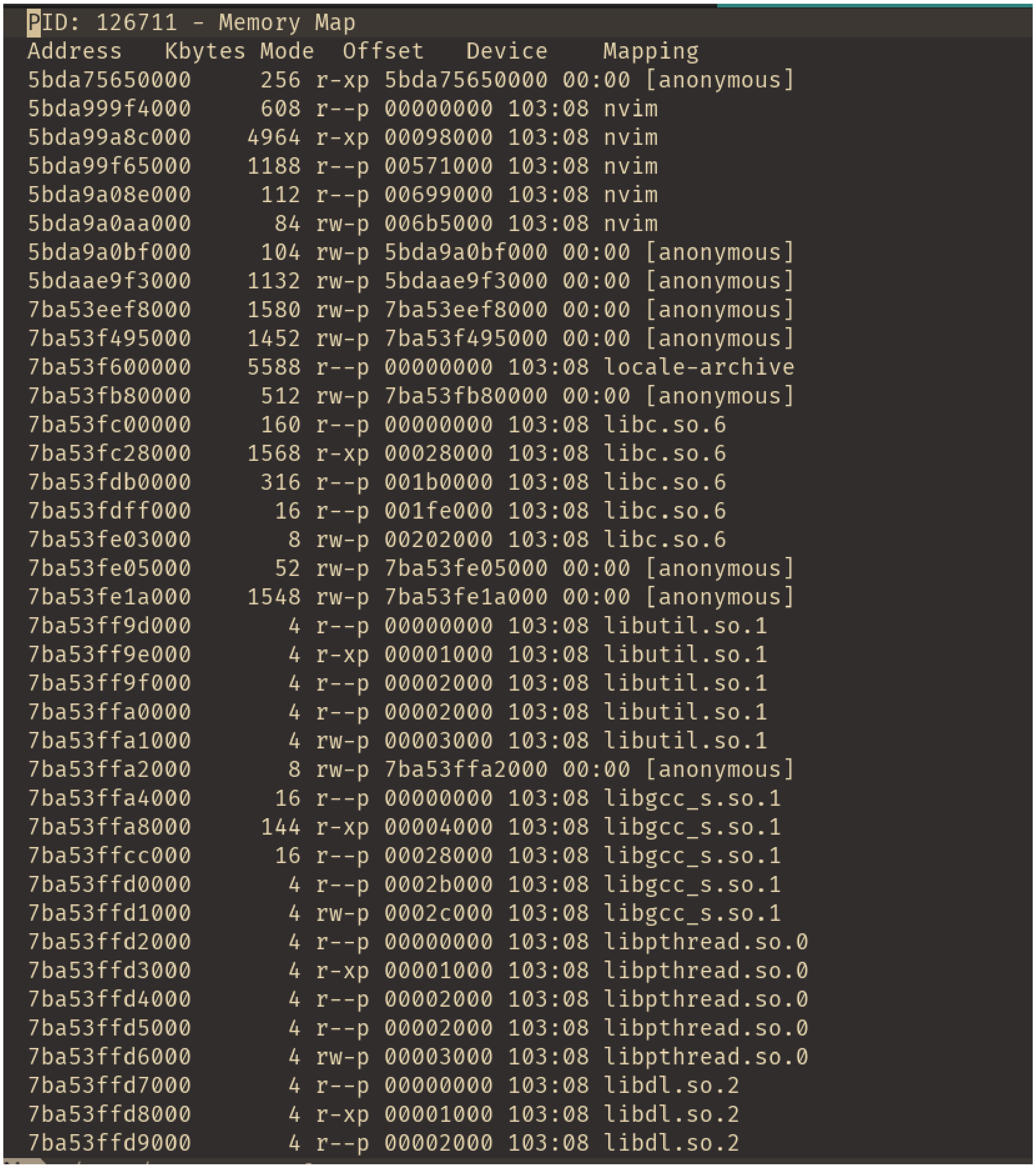


```
obj-m += JackFruit.o

all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

OUTPUT:



```
tejasr@tejas-ubuntu-xps:~/Tejas_PERSONAL/sem4/os/Jackfruit$ sudo dmesg
[  284.663990] Parent (Kernel Thread) PID: 20850
[  284.664159]    |--Kernel Child 0 (PID: 20851) started
[  284.664163]       |--Child 0 allocated memory at: 00000000d93368b3
[  284.664323]    |--Kernel Child 1 (PID: 20852) started
[  284.664328]       |--Child 1 allocated memory at: 00000000ff7d8481
[  284.664408]    |--Kernel Child 2 (PID: 20853) started
[  284.664409] Module initialized. Check /proc/memory_map for memory ma
[  284.664410]       |--Child 2 allocated memory at: 000000000c39a429
```



```
PID: 126711 - Memory Map
Address     Kbytes Mode  Offset    Device    Mapping
5bda75650000     256 r-xp 5bda75650000 00:00 [anonymous]
5bda999f4000     608 r--p 00000000 103:08 nvim
5bda99a8c000    4964 r-xp 00098000 103:08 nvim
5bda99f65000    1188 r--p 00571000 103:08 nvim
5bda9a08e000     112 r--p 00699000 103:08 nvim
5bda9a0aa000      84 rw-p 006b5000 103:08 nvim
5bda9a0bf000     104 rw-p 5bda9a0bf000 00:00 [anonymous]
5bdaae9f3000    1132 rw-p 5bdaae9f3000 00:00 [anonymous]
7ba53eef8000    1580 rw-p 7ba53eef8000 00:00 [anonymous]
7ba53f495000    1452 rw-p 7ba53f495000 00:00 [anonymous]
7ba53f600000    5588 r--p 00000000 103:08 locale-archive
7ba53fb80000     512 rw-p 7ba53fb80000 00:00 [anonymous]
7ba53fc00000     160 r--p 00000000 103:08 libc.so.6
7ba53fc28000    1568 r-xp 00028000 103:08 libc.so.6
7ba53fdb0000     316 r--p 001b0000 103:08 libc.so.6
7ba53fdff000      16 r--p 001fe000 103:08 libc.so.6
7ba53fe03000       8 rw-p 00202000 103:08 libc.so.6
7ba53fe05000      52 rw-p 7ba53fe05000 00:00 [anonymous]
7ba53fe1a000    1548 rw-p 7ba53fe1a000 00:00 [anonymous]
7ba53ff9d000       4 r--p 00000000 103:08 libutil.so.1
7ba53ff9e000       4 r-xp 00001000 103:08 libutil.so.1
7ba53ff9f000       4 r--p 00002000 103:08 libutil.so.1
7ba53ffa0000       4 r--p 00002000 103:08 libutil.so.1
7ba53ffa1000       4 rw-p 00003000 103:08 libutil.so.1
7ba53ffa2000       8 rw-p 7ba53ffa2000 00:00 [anonymous]
7ba53ffa4000      16 r--p 00000000 103:08 libgcc_s.so.1
7ba53ffa8000     144 r-xp 00004000 103:08 libgcc_s.so.1
7ba53ffcc000      16 r--p 00028000 103:08 libgcc_s.so.1
7ba53ffd0000       4 r--p 0002b000 103:08 libgcc_s.so.1
7ba53ffd1000       4 rw-p 0002c000 103:08 libgcc_s.so.1
7ba53ffd2000       4 r--p 00000000 103:08 libpthread.so.0
7ba53ffd3000       4 r-xp 00001000 103:08 libpthread.so.0
7ba53ffd4000       4 r--p 00002000 103:08 libpthread.so.0
7ba53ffd5000       4 r--p 00002000 103:08 libpthread.so.0
7ba53ffd6000       4 rw-p 00003000 103:08 libpthread.so.0
7ba53ffd7000       4 r--p 00000000 103:08 libdl.so.2
7ba53ffd8000       4 r-xp 00001000 103:08 libdl.so.2
7ba53ffd9000       4 r--p 00002000 103:08 libdl.so.2
```