Lab02

Ahmed Siradj Eddine Bekkari

November 2024

Part01

- Github Link
- `make`

Figure 1: make command

- `sudo insmod hello_world.ko` then `sudo dmesg | tail`

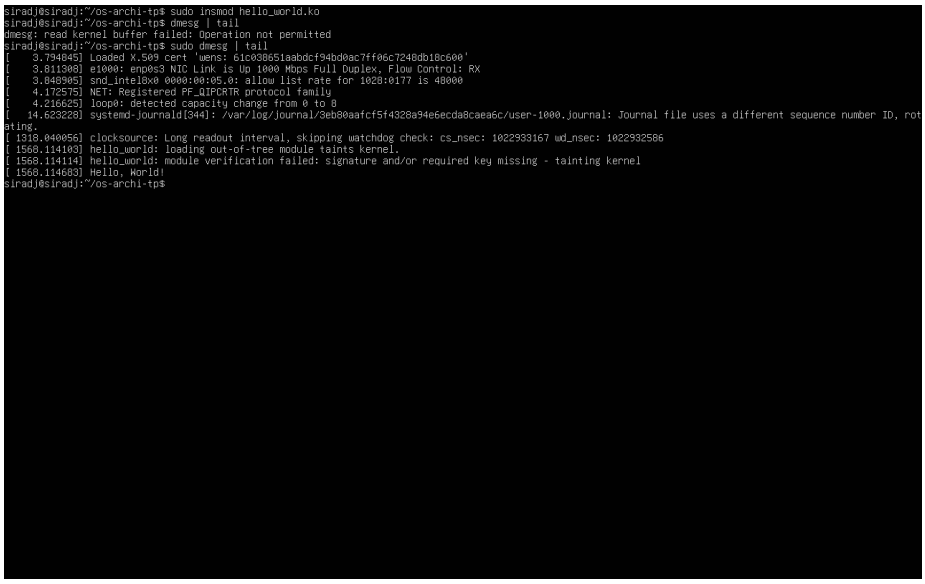


Figure 2: inject the module on the kernel then show the log message

- `sudo rmmod hello_world` then `sudo dmesg | tail`

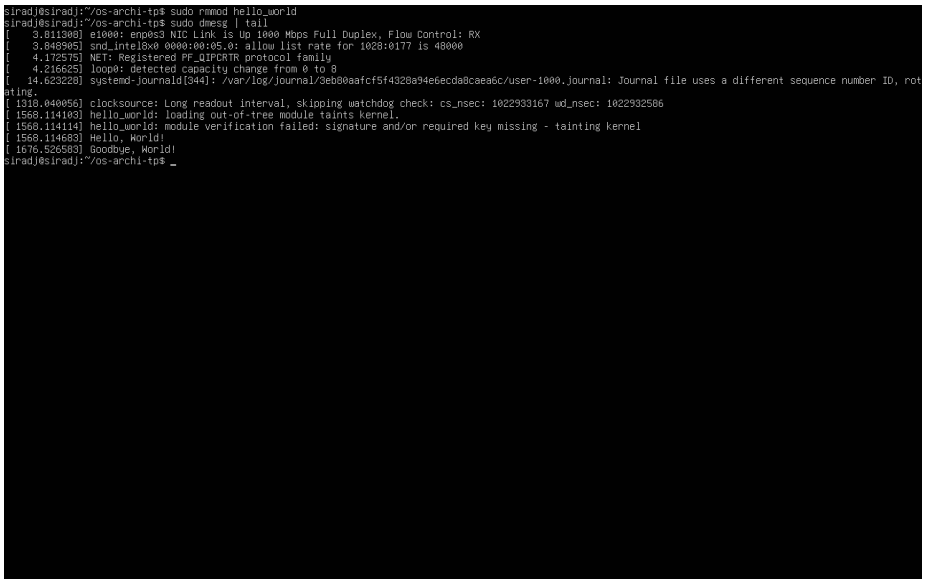


Figure 3: Remove the module from the kernel then show the log

Part02

- Github Link
- `make`

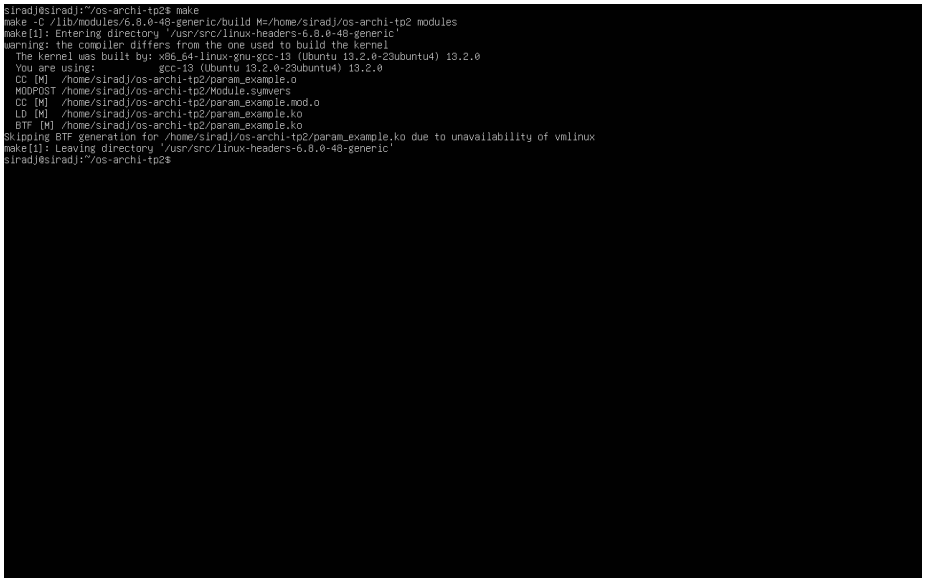


Figure 4: make command

- `sudo insmod hello_world.ko` then `sudo dmesg | tail`

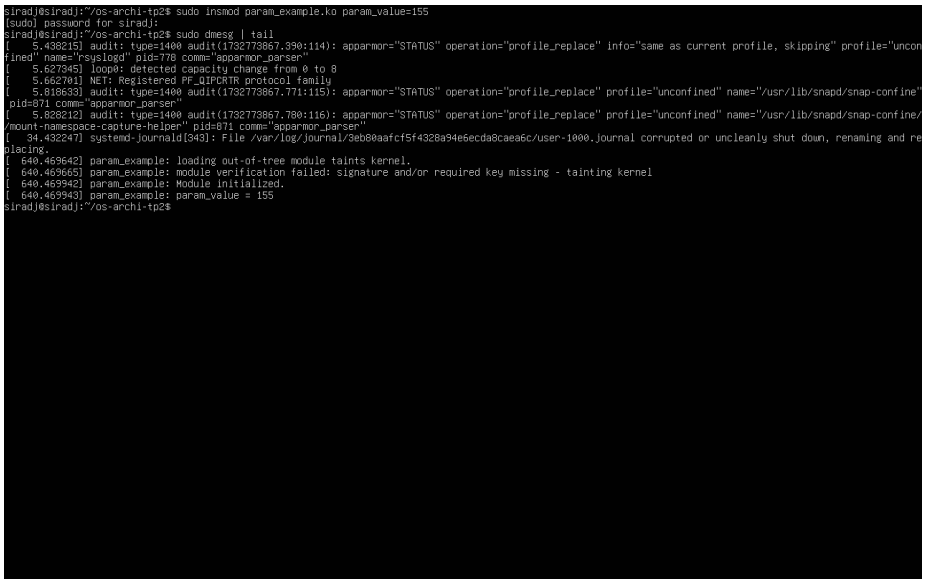


Figure 5: inject the module on the kernel then show the log message

- `sudo rmmod hello_world` then `sudo dmesg | tail`

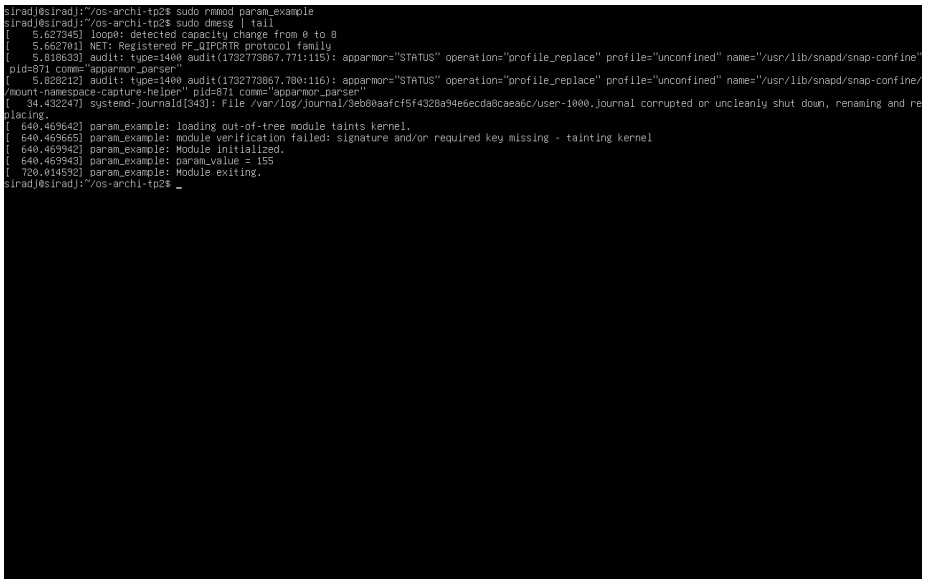


Figure 6: Remove the module from the kernel then show the log

Part03: Linked List Module

Linked list C code

```c
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/slab.h> // For kmalloc and kfree
#include <linux/list.h> // For linked list functions

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Devv");
MODULE_DESCRIPTION("A kernel module demonstrating linked list management.");

// Define the structure for each node in the linked list
typedef struct node {
    int data;
    struct list_head list;
} node_t;

// Declare the list head
static LIST_HEAD(my_list);

static int __init linked_list_init(void) {
    printk(KERN_INFO "Linked List Module: Initialized\n");

    // Add some nodes to the list
    int i;
    for (i = 1; i <= 5; i++) {
        node_t *new_node = kmalloc(sizeof(node_t), GFP_KERNEL);
        if (!new_node) {
            printk(KERN_ERR "Failed to allocate memory for node\n");
            return -ENOMEM; // Return appropriate error code
        }
        new_node->data = i * 10;
        list_add_tail(&new_node->list, &my_list);
    }

    // Traverse and print the list
    printk(KERN_INFO "Linked List contents:\n");
    node_t *current_node;
    list_for_each_entry(current_node, &my_list, list) {
        printk(KERN_INFO "Data: %d\n", current_node->data);
    }

    return 0;
}

static void __exit linked_list_exit(void) {
    // Safely traverse and free the list
    node_t *current_node, *temp_node;
    list_for_each_entry_safe(current_node, temp_node, &my_list, list) {
        list_del(&current_node->list);
        kfree(current_node);
    }
```

```c

    printk(KERN_INFO "Linked List Module: Exited, memory freed\n");
}

module_init(linked_list_init);
module_exit(linked_list_exit);
```

Listing 1: C Code

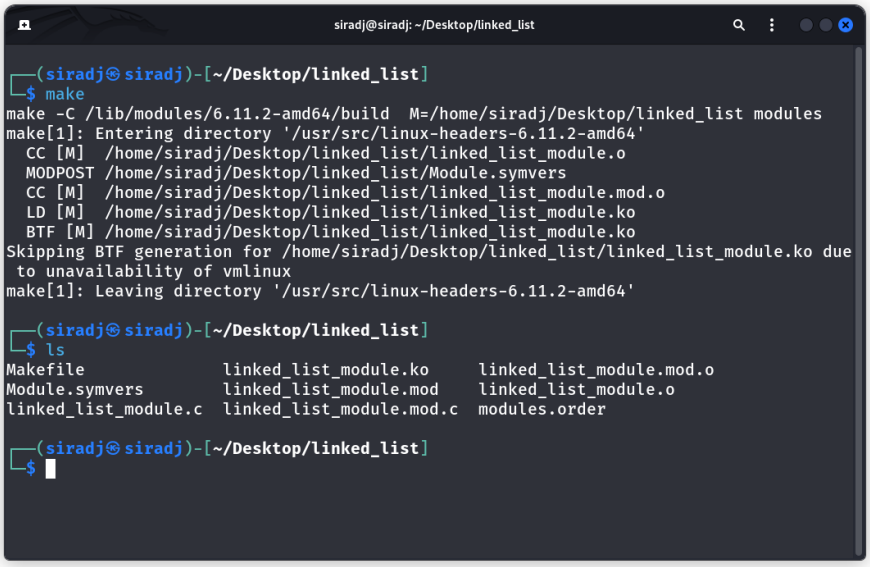
Makefile

```
obj-m += linked_list_module.o
KDIR := /lib/modules/$(shell uname -r)/build
all:
    make -C $(KDIR) M=$(PWD) modules
clean:
    make -C $(KDIR) M=$(PWD) clean
```

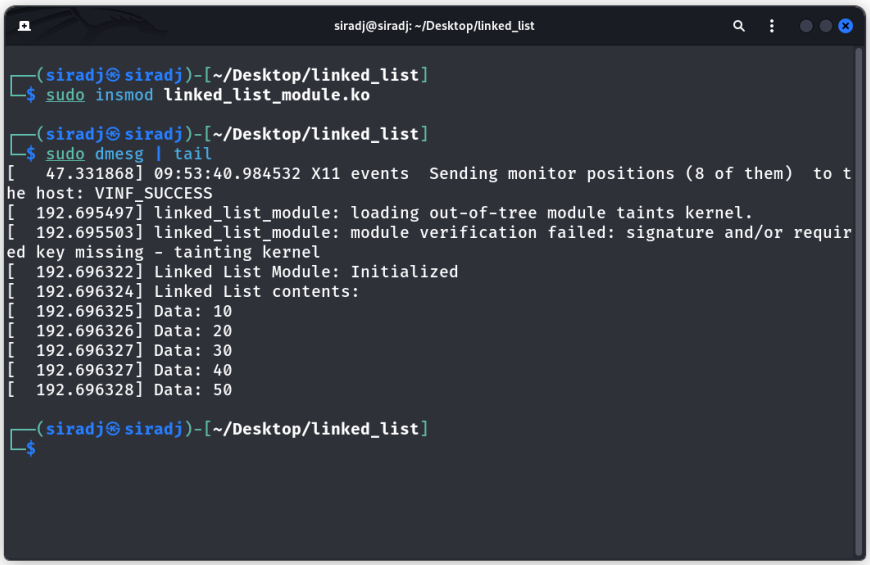
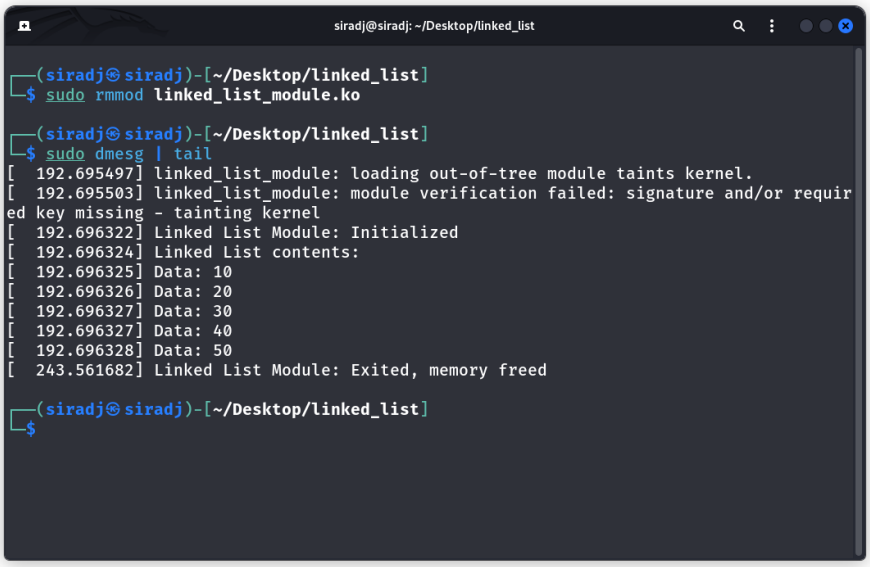
Listing 2: Makefile

Execution

Building:



Install the module:



Remove the module: