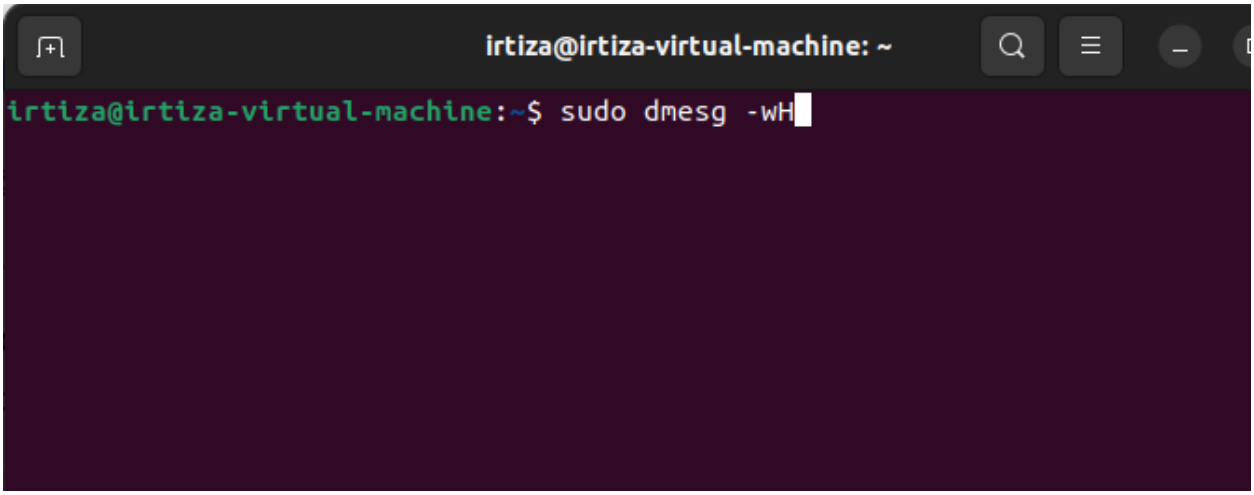
Interrupts Lab

Q1. Make a simple network interrupt.

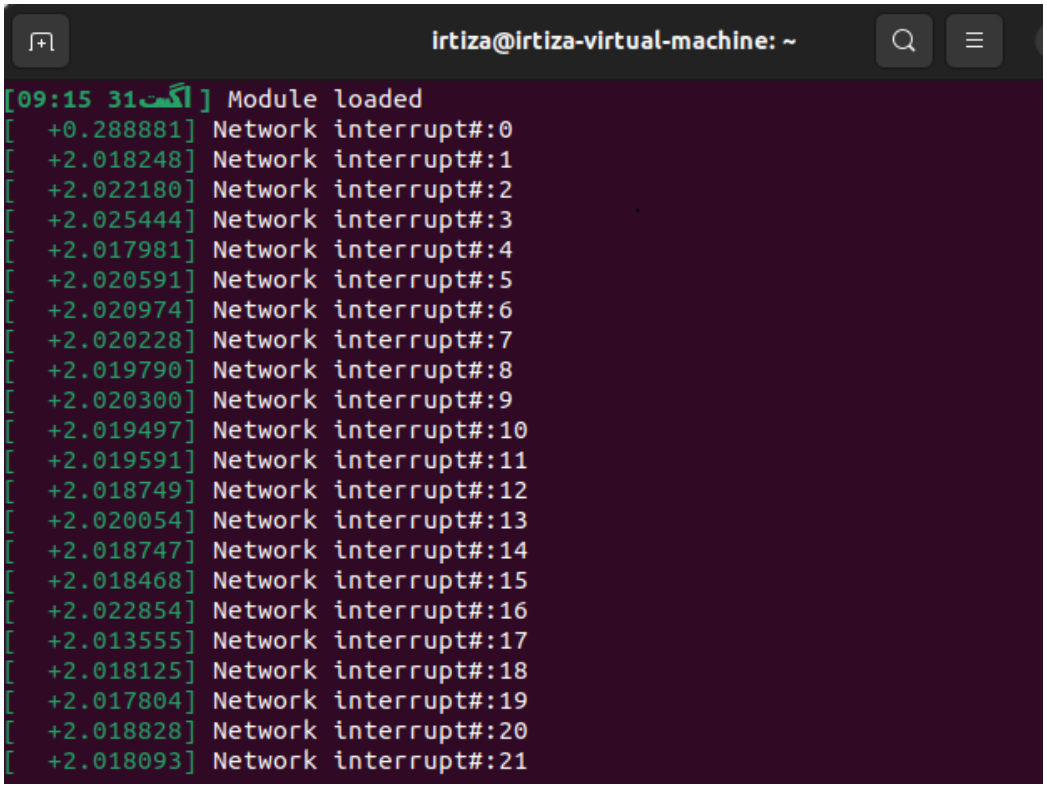
```
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/module.h>
#include <linux/interrupt.h>
MODULE_LICENSE("GPL");
static int irq = 19,  dev = 1, counter = 0;
static irqreturn_t network_handler(int irq, void *dev)
{
        printk(KERN_INFO"Network interrupt#:%d\n", counter++);

        return IRQ_HANDLED;
}

static int function_init(void)
{
        printk(KERN_INFO"Module loaded\n");
        return request_irq(irq, network_handler,
IRQF_SHARED,"my_network_handler", &dev);
}

static void function_exit(void)
{
        printk(KERN_INFO"Module unloaded\n");
        free_irq(irq, &dev);
}
module_init(function_init);
module_exit(function_exit);
```

Open a terminal and enter

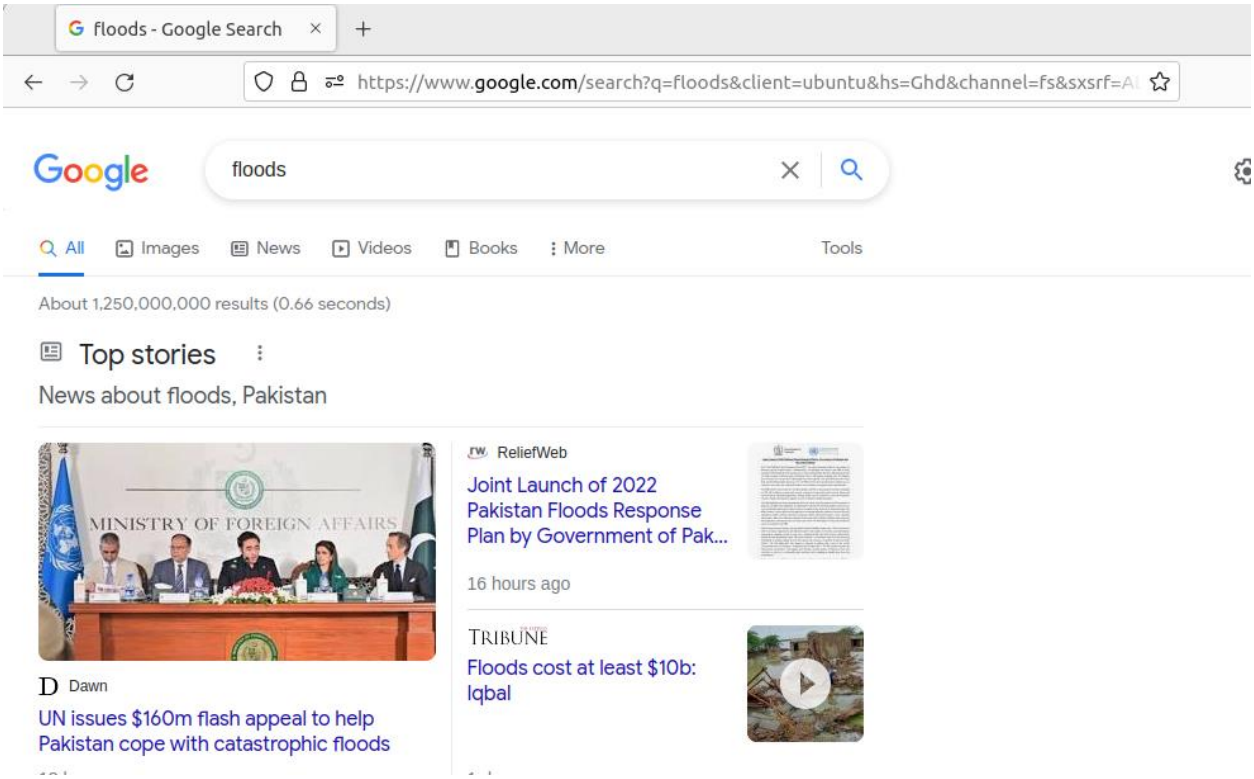
```
sudo dmesg -wH
```

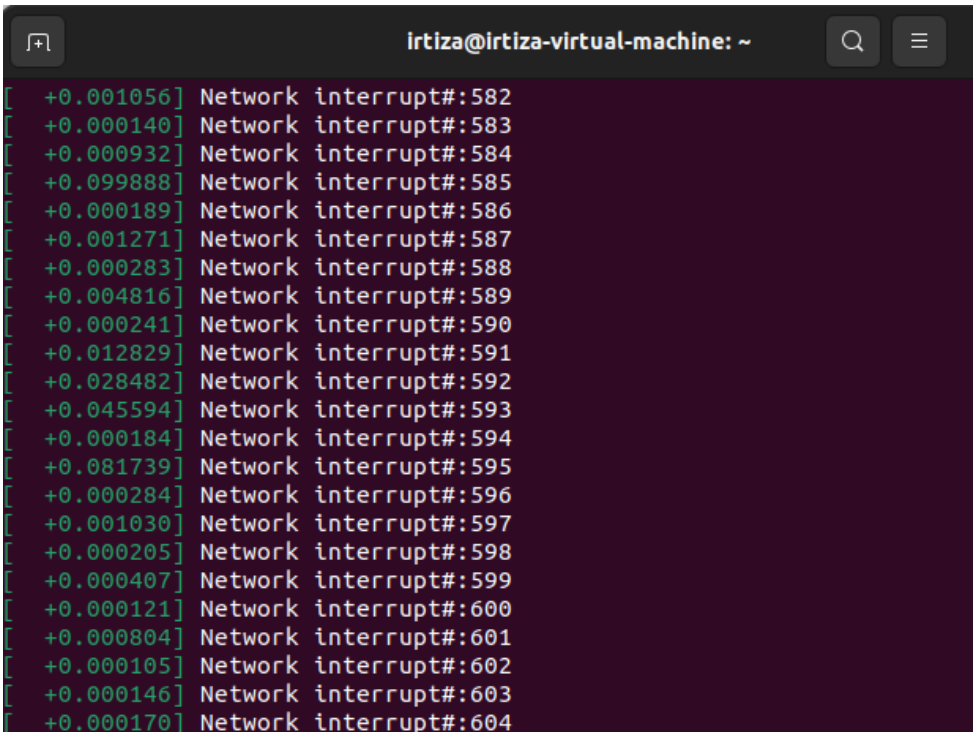


You will be able to see changing kernel log in real time



Now open your browser and open a website and see the above terminal. You will see that the kernel is receiving many interrupts from the network card.


floods - Google Search
https://www.google.com/search?q=floods&client=ubuntu&hs=Ghd&channel=fs&sxsrf=A
Google
floods
All
Images
News
Videos
Books
More
Tools
About 1,250,000,000 results (0.66 seconds)
Top stories
News about floods, Pakistan
MINISTRY OF FOREIGN AFFAIRS
Dawn
UN issues $160m flash appeal to help Pakistan cope with catastrophic floods
ReliefWeb
Joint Launch of 2022 Pakistan Floods Response Plan by Government of Pak...
16 hours ago
TRIBUNE
Floods cost at least $10b: Iqbal


irtiza@irtiza-virtual-machine: ~
[+0.001056] Network interrupt#:582
[+0.000140] Network interrupt#:583
[+0.000932] Network interrupt#:584
[+0.099888] Network interrupt#:585
[+0.000189] Network interrupt#:586
[+0.001271] Network interrupt#:587
[+0.000283] Network interrupt#:588
[+0.004816] Network interrupt#:589
[+0.000241] Network interrupt#:590
[+0.012829] Network interrupt#:591
[+0.028482] Network interrupt#:592
[+0.045594] Network interrupt#:593
[+0.000184] Network interrupt#:594
[+0.081739] Network interrupt#:595
[+0.000284] Network interrupt#:596
[+0.001030] Network interrupt#:597
[+0.000205] Network interrupt#:598
[+0.000407] Network interrupt#:599
[+0.000121] Network interrupt#:600
[+0.000804] Network interrupt#:601
[+0.000105] Network interrupt#:602
[+0.000146] Network interrupt#:603
[+0.000170] Network interrupt#:604

Q2. Use taslets to make a top half and bottom half counter

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/interrupt.h>
MODULE_LICENSE("GPL");
static int irq = 19,  dev = 1, counter1 = 0,counter2=0;
static const unsigned long dat=2;
static void tasklet_func(unsigned long dat1){
        printk(KERN_INFO"Bottom Half called %d\n",counter2++);
}
struct tasklet_struct
my_tasklet={.next=NULL,.state=0,.count=ATOMIC_INIT(0),.func=tasklet_func,.data=dat};
static irqreturn_t network_handler(int irq, void *dev)
{
        printk(KERN_INFO"Top half called:%d\n", counter1++);
        tasklet_schedule(&my_tasklet);
        return IRQ_HANDLED;
}

static int function_init(void)
{
        printk(KERN_INFO"Module loaded\n");
        return request_irq(irq, network_handler, IRQF_SHARED,"my_network_handler",
&dev);
}

static void function_exit(void)
{
        printk(KERN_INFO"Module unloaded\n");
        tasklet_kill(&my_tasklet);
        free_irq(irq, &dev);
}
module_init(function_init);
module_exit(function_exit);
```

```
irtiza@irtiza-virtual-machine: ~
[  +2.015417] Top half called:4
[  +0.000143] Bottom Half called 4
[  +2.016281] Top half called:5
[  +0.000140] Bottom Half called 5
[  +2.016489] Top half called:6
[  +0.000145] Bottom Half called 6
[  +2.015832] Top half called:7
[  +0.000143] Bottom Half called 7
[  +2.015926] Top half called:8
[  +0.000191] Bottom Half called 8
[  +2.016281] Top half called:9
[  +0.000139] Bottom Half called 9
[  +0.288497] Top half called:10
[  +0.000203] Bottom Half called 10
[  +0.002032] Top half called:11
[  +0.000220] Bottom Half called 11
[  +0.001458] Top half called:12
[  +0.000065] Bottom Half called 12
[  +0.292383] Top half called:13
[  +0.000380] Top half called:14
[  +0.000220] Bottom Half called 13
```

If you load a website on your browser you will notice that bottom half is called less number of times than top half.

Q3.Use tasklets to print a message after every 1000 interrupts.

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/interrupt.h>
MODULE_LICENSE("GPL");
static int irq = 19,  dev = 1, counter1 = 0,counter2=0;
static const unsigned long dat=2;
static void tasklet_func(unsigned long dat1){
        if(counter2>=1000){
                printk("More than 1000\n");
                counter2=0;
        }
}
struct tasklet_struct
my_tasklet={.next=NULL,.state=0,.count=ATOMIC_INIT(0),.func=tasklet_func,.data=dat};
static irqreturn_t keyboard_handler(int irq, void *dev)
{
        counter1++;
        counter2++;
        printk(KERN_INFO"Interrupt counter:%d\n", counter1);
        tasklet_schedule(&my_tasklet);
        return IRQ_HANDLED;
}

static int function_init(void)
{
        printk(KERN_INFO"Module loaded\n");
        return request_irq(irq, keyboard_handler, IRQF_SHARED,"my_network_handler", &dev);
}

static void function_exit(void)
{
        printk(KERN_INFO"Module unloaded\n");
        tasklet_kill(&my_tasklet);
        free_irq(irq, &dev);
}
module_init(function_init);
module_exit(function_exit);
```

Q4.Use Workqueues to make top half and bottom half counter.

```
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/module.h>
#include <linux/interrupt.h>
#include <linux/workqueue.h>
MODULE_LICENSE("GPL");
static int irq = 19,  dev = 1, counter1 = 0,counter2=0;
void workqueue_func(struct work_struct *work){
        printk(KERN_INFO"Bottom Half called %d\n",counter2++);
}
DECLARE_WORK(workq,workqueue_func);
static irqreturn_t network_handler(int irq, void *dev)
{
        printk(KERN_INFO"Top half called:%d\n", counter1++);
        schedule_work(&work);
        return IRQ_HANDLED;
}

static int function_init(void)
{
        printk(KERN_INFO"Module loaded\n");
        return request_irq(irq, network_handler, IRQF_SHARED,"my_network_handler",
&dev);
}
static void function_exit(void)
{
        printk(KERN_INFO"Module unloaded\n");
        flush_scheduled_work();
        free_irq(irq, &dev);
}
module_init(function_init);
module_exit(function_exit);
```

Q5.Use Workqueues to print a message after every 1000 interrupts.

```
#include <linux/module.h>
#include <linux/interrupt.h>
#include <linux/workqueue.h>
MODULE_LICENSE("GPL");
static int irq = 19,  dev = 1, counter1 = 0,counter2=0;

static void work_func(struct work_struct *work){
        if(counter2>=1000){
                printk(KERN_INFO"more than 1000\n");
                counter2=0;
        }
}
DECLARE_WORK(work,work_func);
static irqreturn_t network_handler(int irq, void *dev)
{
        counter1++;
        counter2++;
        printk(KERN_INFO"Network interrupt#:%d\n", counter1);
        schedule_work(&work);
        return IRQ_HANDLED;
}

static int function_init(void)
{
        printk(KERN_INFO"Module loaded\n");
        return request_irq(irq, network_handler, IRQF_SHARED,"my_network_handler",
&dev);
}

static void function_exit(void)
{
        printk(KERN_INFO"Module unloaded\n");
        flush_scheduled_work();
        free_irq(irq, &dev);
}
module_init(function_init);
module_exit(function_exit);
```