3.Create and load a basic LKM into the Linux kernel, which prints a message when loaded and unloaded.

sudo apt update

sudo apt install build-essential linux-headers-$(uname -r)

mkdir ~/basic_lkm

cd ~/basic_lkm

gedit hello_lkm.c

#include <linux/init.h>

#include <linux/module.h>

#include <linux/kernel.h>

```
MODULE_LICENSE("GPL");
MODULE_AUTHOR("YourName");
MODULE_DESCRIPTION("A simple Hello World LKM");
MODULE_VERSION("1.0");

static int __init hello_lkm_init(void) {
    printk(KERN_INFO "Hello: LKM loaded into the kernel\n");
    return 0;
}

static void __exit hello_lkm_exit(void) {
    printk(KERN_INFO "Goodbye: LKM unloaded from the kernel\n");
}

module_init(hello_lkm_init);
module_exit(hello_lkm_exit);
```

gedit Makefile

```
obj-m += hello_lkm.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
```

make

sudo insmod hello_lkm.ko

dmesg | tail -n 10

Hello: LKM loaded into the kernel

sudo rmmod hello_lkm

dmesg | tail -n 10

Goodbye: LKM unloaded from the kernel

make clean

4 . Create and load an LKM that accepts parameters into the Linux kernel, and observe how parameter values affect the LKM's behavior

```
sudo apt update
sudo apt install build-essential linux-headers-$(uname -r)
```

```
mkdir ~/my_lkm
cd ~/my_lkm
```

```
gedit param_lkm.c
```

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/moduleparam.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("YourName");
MODULE_DESCRIPTION("A simple Linux driver with parameters");
MODULE_VERSION("1.0");

// Declare module parameters
static int myint = 0;
module_param(myint, int, 0660);
MODULE_PARM_DESC(myint, "An integer");

static char *mystring = "default";
module_param(mystring, charp, 0660);
```

```
 MODULE_PARM_DESC(mystring, "A string");

// Init and Exit functions
static int __init param_lkm_init(void) {
    printk(KERN_INFO "param_lkm: Module loaded\n");
    printk(KERN_INFO "param_lkm: myint = %d, mystring = %s\n", myint, mystring);
    return 0;
}

static void __exit param_lkm_exit(void) {
    printk(KERN_INFO "param_lkm: Module unloaded\n");
}

module_init(param_lkm_init);
module_exit(param_lkm_exit);
```

```
gedit Makefile
obj-m += param_lkm.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
```

```
make
```

```
sudo insmod param_lkm.ko myint=42 mystring="hello_kernel"
```

```
dmesg | tail -n 20
```

5. Create an LKM that generates a /proc file containing the PIDs and names of all running processes

```
sudo apt update
sudo apt install build-essential linux-headers-$(uname -r)
```

```
mkdir ~/proc_lkm
cd ~/proc_lkm
```

```
gedit proc_lkm.c
```

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/proc_fs.h>
#include <linux/seq_file.h>
#include <linux/sched/signal.h>

MODULE_LICENSE("GPL");
```

```
MODULE_AUTHOR("YourName");
MODULE_DESCRIPTION("LKM that lists all process names and PIDs in /proc/process_list");
MODULE_VERSION("1.0");

#define PROC_NAME "process_list"

static int show_processes(struct seq_file *m, void *v) {
    struct task_struct *task;

    seq_printf(m, "PID\tProcess Name\n");
    for_each_process(task) {
        seq_printf(m, "%d\t%s\n", task->pid, task->comm);
    }

    return 0;
}

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

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

```
static int __init proc_lkm_init(void) {
    proc_create(PROC_NAME, 0, NULL, &proc_file_ops);
    printk(KERN_INFO "/proc/%s created\n", PROC_NAME);
    return 0;
}

static void __exit proc_lkm_exit(void) {
    remove_proc_entry(PROC_NAME, NULL);
    printk(KERN_INFO "/proc/%s removed\n", PROC_NAME);
}

module_init(proc_lkm_init);
module_exit(proc_lkm_exit);
```

```
gedit Makefile
```

```
obj-m += proc_lkm.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
```

```
make
```

```
sudo insmod proc_lkm.ko
```

dmesg | tail -n 5

cat /proc/process_list

sudo rmmod proc_lkm

dmesg | tail -n 5

ls /proc/process_list

ls /proc/process_list

make clean

Create an LKM that changes the priority of a specific process identified by its PID

sudo apt update

sudo apt install build-essential linux-headers-$(uname -r)

mkdir ~/priority_lkm

cd ~/priority_lkm

gedit priority_lkm.c

```
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/sched/signal.h>
```

```
#include <linux/moduleparam.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("YourName");
MODULE_DESCRIPTION("LKM to change process priority by PID");
MODULE_VERSION("1.0");

static int pid = -1;
static int new_nice = 0;

module_param(pid, int, 0644);
MODULE_PARM_DESC(pid, "PID of the process to modify");

module_param(new_nice, int, 0644);
MODULE_PARM_DESC(new_nice, "New nice value (priority) for the process");

static int __init priority_lkm_init(void) {
    struct task_struct *task;

    if (pid <= 0 || new_nice < -20 || new_nice > 19) {
        printk(KERN_ERR "Invalid PID or nice value (must be between -20 and 19)\n");
        return -EINVAL;
    }

    for_each_process(task) {
        if (task->pid == pid) {
            printk(KERN_INFO "Found process: %s (PID: %d), current nice: %d\n",
                task->comm, task->pid, task_nice(task));
```

```
            set_user_nice(task, new_nice);
            printk(KERN_INFO "Priority changed: new nice value = %d\n", task_nice(task));
            return 0;
        }
    }

    printk(KERN_ERR "Process with PID %d not found\n", pid);
    return -ESRCH;
}

static void __exit priority_lkm_exit(void) {
    printk(KERN_INFO "priority_lkm: Module unloaded\n");
}

module_init(priority_lkm_init);
module_exit(priority_lkm_exit);
```

gedit Makefile

```
obj-m += priority_lkm.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
```

make

sudo insmod priority_lkm.ko pid=1234 new_nice=5

dmesg | tail -n 10

Found process: bash (PID: 1234), current nice: 0

Priority changed: new nice value = 5

Process with PID 1234 not found

sudo rmmod priority_lkm

make clean

8. Create an LKM that that will display a list of only those tasks which are ‘kernel threads’? (i.e., task->mm == 0). How many ‘kernel threads’ on your list?

sudo apt update

sudo apt install build-essential linux-headers-$(uname -r)

mkdir ~/kernel_threads_lkm

cd ~/kernel_threads_lkm

gedit kernel_threads_lkm.c

#include <linux/init.h>

#include <linux/module.h>

```
#include <linux/kernel.h>
#include <linux/sched/signal.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("YourName");
MODULE_DESCRIPTION("LKM to list kernel threads (task->mm == NULL)");
MODULE_VERSION("1.0");

static int __init kernel_threads_init(void) {
    struct task_struct *task;
    int count = 0;

    printk(KERN_INFO "Listing all kernel threads (task->mm == NULL):\n");
    for_each_process(task) {
        if (task->mm == NULL) {
            printk(KERN_INFO "PID: %d\tName: %s\n", task->pid, task->comm);
            count++;
        }
    }

    printk(KERN_INFO "Total kernel threads: %d\n", count);
    return 0;
}

static void __exit kernel_threads_exit(void) {
    printk(KERN_INFO "Kernel thread lister unloaded.\n");
}
```

```
module_init(kernel_threads_init);
module_exit(kernel_threads_exit);
```

```
gedit Makefile
```

```
obj-m += kernel_threads_lkm.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
```

```
make
```

```
sudo insmod kernel_threads_lkm.ko
```

```
dmesg | tail -n 30
```

1. Configure the Linux kernel according to specific hardware and software requirements

```
sudo apt update
sudo apt install git build-essential libncurses-dev bison flex libssl-dev libelf-dev

mkdir ~/kernel_config
cd ~/kernel_config
git clone https://github.com/torvalds/linux.git
cd linux
```

```
git checkout v6.1

make defconfig

make menuconfig

make -j$(nproc)

sudo make modules_install

sudo make install

sudo update-initramfs -c -k $(make kernelrelease)
sudo update-grub
sudo reboot

make clean
make mrproper
```