Linux Kernel Module to Monitor Battery status

A REPORT ON PACKAGE SUBMITTED BY

SAKTHIVEL C R
Roll No. 18PT30

PUSHKARAN A
Roll No. 18PT27

APRIL 2020

DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCES

PSG COLLEGE OF TECHNOLOGY
(Autonomous Institution)

COIMBATORE – 641 004.

CONTENTS

CHAPTER 1

INTRODUCTION

A brief overview of the problem statement on which the project was done, introduction about the project, the configuration used to develop and deploy, tools and technologies used during the course of the project are described in this chapter.

1.1 WHY THIS ASSIGNMENT?

Linux provides a powerful and expansive API for applications but sometimes that's not enough.Interacting with a piece of hardware or conducting operations which require privileged instructions require communication with the kernel through kernel modules.

A linux kernel module is a piece of compiled binary code inserted directly into the kernel,thus operating with privileged mode and has access to everything in the system.

1.2 DESCRIPTION

As part of this assignment, A linux kernel module has been designed to monitor the battery power of our system.if our battery power is low a kernel alert is notified and the brightness of our system is reduced and bluetooth is switched off.To obtain the battery power of our system ,we access the file 'capacity' which is in the location '/sys/class/power_supply/BAT1/capacity'.

However, from within the linux kernel, reading data out of a file for configuration information is considered to be forbidden for a number of reasons.A major reason is that kernel uses multiple file system namespaces for user processes and the kernel should know which namespace it should use and the kernel code should be kept simple as any bug in it can have serious consequences. Working with files requires being aware of various locking issues and would add unnecessary complexity.

Hence we use filp_open() which comes under <kernel/fs.h> , a kernel library to work with file structures.Before reading from the file , we must handle the address space mismatch by using the functions get_fs() and set_fs().The reason is that the kernel expects the pointer passed to the filp_open() function call to be coming from user space. So, it makes a check of the pointer to verify it is in the proper address space in order to try to convert it to a kernel pointer that the rest of the kernel can use. So, when we are trying to pass a kernel pointer to the function, the error -EFAULT occurs.So we get the current address space limits using the get_fs() function and set the address space to kernel space by using 'set_fs(KERNEL_DS)' and then resetting the address space after reading the file.

Now, after reading our system's battery we can issue a kernel alert buy using macros under the library <kernel.h>.

1.3 SYSTEM CALLS

The system call used is

msleep(int msecs);

which allows you to sleep even with wait queue instructions.The arguments for sleep is msecs which is the time in milliseconds to sleep for.

1.4 TOOLS

The basic prerequisite for module programming is a Linux machine.While any Linux distribution will do, Ubuntu 18.04 LTS is used in this example, so if you’re using a different distribution you may need to slightly adjust your installation commands.

We need either a physical machine or virtual machine,but working in a virtual machine is more preferred because data loss can occur when we make a mistake. Our latest code changes may still be in the write buffer when the kernel panics, so it’s possible that your source files can become corrupted. Testing in a virtual machine eliminates this risk and we use C language to write our modules.

CHAPTER 2

WORKFLOW

2.1 INITIALIZATION

Whenever the module is loaded into the kernel, the start function also called the initialization function is called when the module is insmoded into the kernel.In this function a kernel thread is created using the kthread_run() function under the library <kthread.h>.

The thread runs the program batthread(), where the thread is run on a delay of 1second and the function check_battery() is called.

2.2 THREAD

In this kernel module, threads are used to call the functions under a delay instead of processes as they are lightweight and reduce CPU utilization making it less resource hungry. Since we are operating in kernel space,we create a kernel thread and call our function.

2.3 CHECK_BATTERY

In this function,we read the battery power from file /sys/class/power_supply/BAT1/capacity every 10 seconds and store it in a character buffer.To get the integer value of the battery percentage, we use a kernel string conversion function kstrtoint() .

After checking the battery and if the power reaches below the threshold ,the macro KERN_ALERT is used to issue an alert to the user and is printed to the console.To decrease the brightness of our system,we read the file /sys/class/backlight/nvidia_0/brightness.and write to the above file using the kernel_write() function.The parameters to be passed are character buffer to be written to the file.So scnprintf() is used to format the string and place it in a buffer and then written into the file

2.4 KERNEL LOG

Whenever the printk() function is called, output is not printed to the user but it is logged in the kernel log which can be accessed by the linux command ‘dmesg’,where the information logged by the kernel about the entire system can be viewed.

CHAPTER 3

SYSTEM IMPLEMENTATION

Visualizing a Kernel module monitoring battery status and issuing an alert and reducing brightness in case of a low battery.

3.1 ABOUT THE FILES

The package has 2 main files

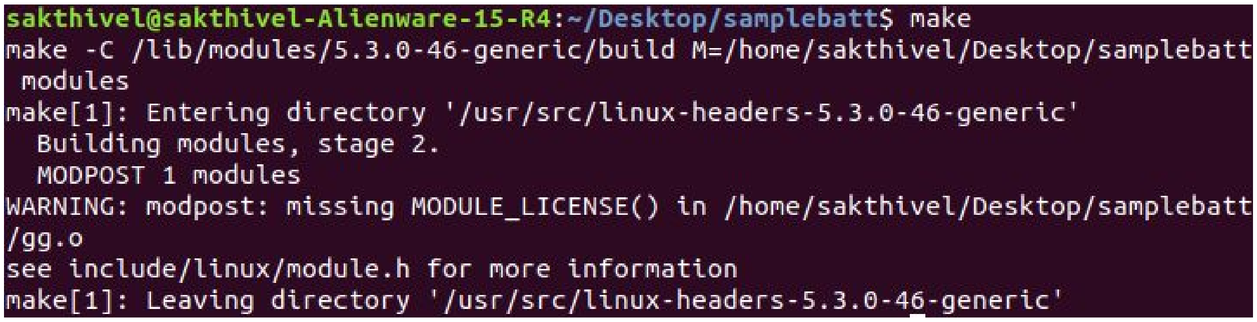
- gg.c -file containing the program
- Makefile-File to build the module

and 2 files which are created when our module is compiles

- gg.ko -kernel object file loaded into memory
- gg.mod.c - contains information about the module like version number etc.

3.2 RUNNING THE FILES

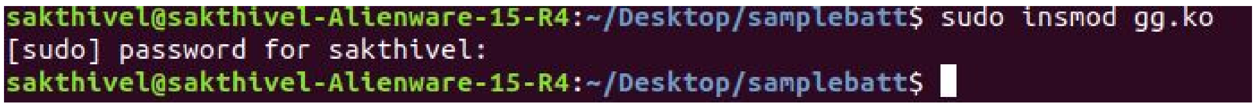
After creating the makefile for the kernel module,type the command ‘make’ to compile the module.



```
sakthivel@sakthivel-Alienware-15-R4:~/Desktop/samplebatt$ make
make -C /lib/modules/5.3.0-46-generic/build M=/home/sakthivel/Desktop/samplebatt
 modules
make[1]: Entering directory '/usr/src/linux-headers-5.3.0-46-generic'
  Building modules, stage 2.
  MODPOST 1 modules
WARNING: modpost: missing MODULE_LICENSE() in /home/sakthivel/Desktop/samplebatt
/gg.o
see include/linux/module.h for more information
make[1]: Leaving directory '/usr/src/linux-headers-5.3.0-46-generic'
```

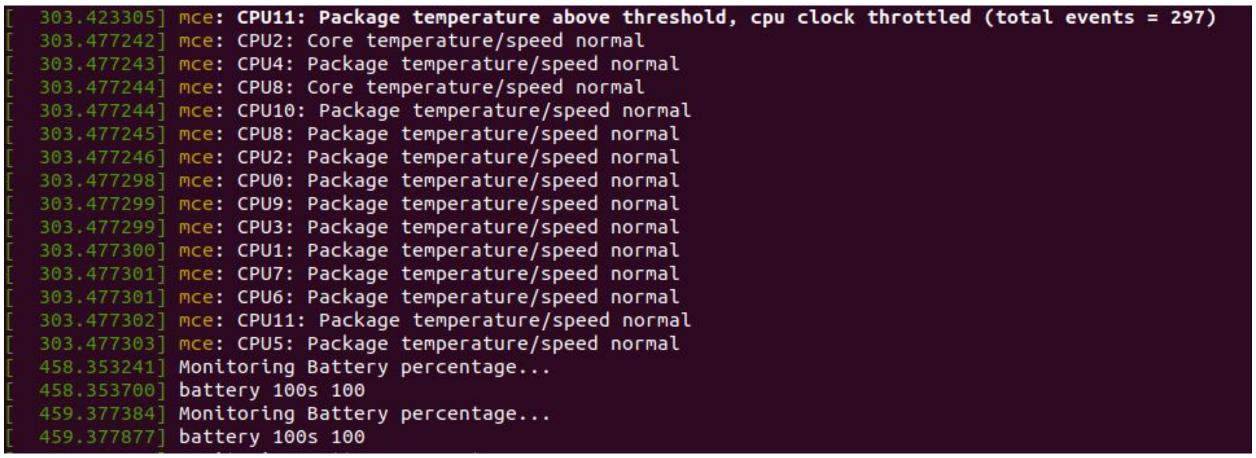
Figure 4.1

To insert module into the kernel,use command ‘sudo insmod gg.ko



```
sakthivel@sakthivel-Alienware-15-R4:~/Desktop/samplebatt$ sudo insmod gg.ko
[sudo] password for sakthivel:
sakthivel@sakthivel-Alienware-15-R4:~/Desktop/samplebatt$
```

Figure 4.2

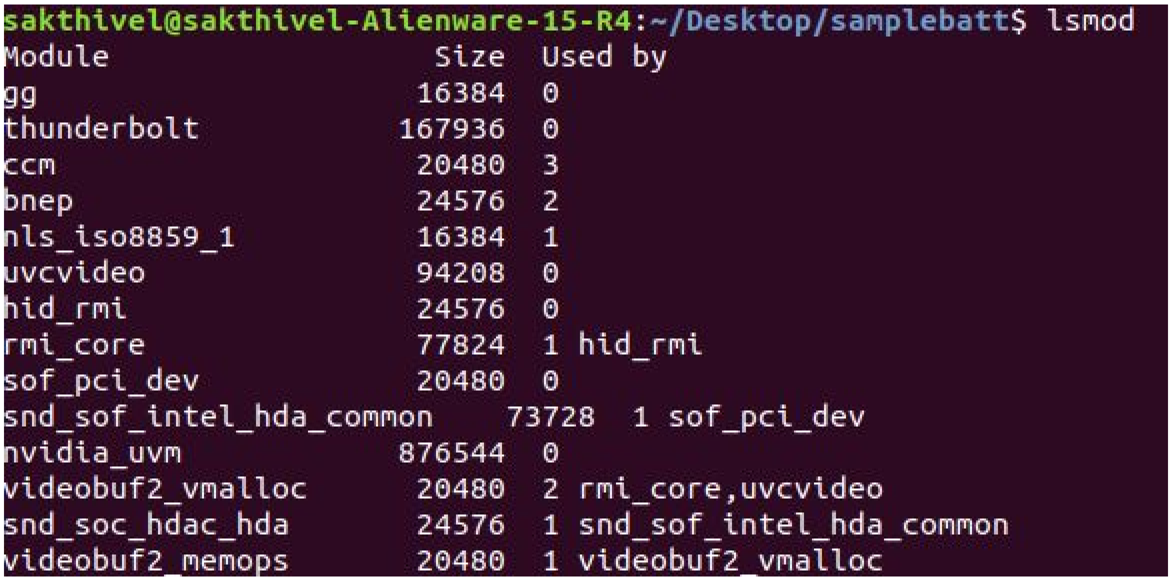


```
[  303.423305] mce: CPU11: Package temperature above threshold, cpu clock throttled (total events = 297)
[  303.477242] mce: CPU2: Core temperature/speed normal
[  303.477243] mce: CPU4: Package temperature/speed normal
[  303.477244] mce: CPU8: Core temperature/speed normal
[  303.477244] mce: CPU10: Package temperature/speed normal
[  303.477245] mce: CPU8: Package temperature/speed normal
[  303.477246] mce: CPU2: Package temperature/speed normal
[  303.477298] mce: CPU0: Package temperature/speed normal
[  303.477299] mce: CPU9: Package temperature/speed normal
[  303.477299] mce: CPU3: Package temperature/speed normal
[  303.477300] mce: CPU1: Package temperature/speed normal
[  303.477301] mce: CPU7: Package temperature/speed normal
[  303.477301] mce: CPU6: Package temperature/speed normal
[  303.477302] mce: CPU11: Package temperature/speed normal
[  303.477303] mce: CPU5: Package temperature/speed normal
[  458.353241] Monitoring Battery percentage...
[  458.353700] battery 100s 100
[  459.377384] Monitoring Battery percentage...
[  459.377877] battery 100s 100
```

Figure 4.3

To view the kernel log, type ‘dmesg’,we’ll be able to view the kernel log.

To view the current modules loaded into the kernel,use ‘lsmod’

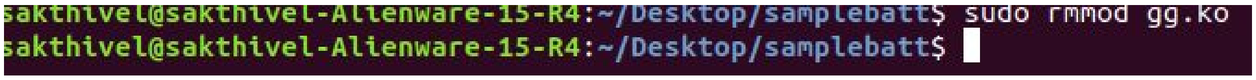


```
sakthivel@sakthivel-Alienware-15-R4:~/Desktop/samplebatt$ lsmod
Module                  Size  Used by
gg                     16384  0
thunderbolt           167936  0
ccm                    20480  3
bnep                   24576  2
nls_iso8859_1          16384  1
uvcvideo               94208  0
hid_rmi                24576  0
rmi_core               77824  1 hid_rmi
sof_pci_dev            20480  0
snd_sof_intel_hda_common    73728  1 sof_pci_dev
nvidia_uvm            876544  0
videobuf2_vmalloc      20480  2 rmi_core,uvcvideo
snd_soc_hdac_hda       24576  1 snd_sof_intel_hda_common
videobuf2_memops       20480  1 videobuf2_vmalloc
```

Figure 4.4

here,gg is our kernel module

To remove a kernel module,use command ‘rmmod gg.ko’



```
sakthivel@sakthivel-Alienware-15-R4:~/Desktop/samplebatt$ sudo rmmod gg.ko
sakthivel@sakthivel-Alienware-15-R4:~/Desktop/samplebatt$
```

Figure 4.5

CHAPTER 4

CONCLUSION

A Kernel Module has been designed to monitor the Battery power in our systems. This has a main moduel program and makefile; printing to the kernel log happens and output is received respectively.

Here battery percentage is stored in a buffer and When the battery is low, Kernel alert is issued and brightness is decreased accordingly.

BIBLIOGRAPHY

BOOKS

1. Silberschatz, Gavin, Gagne, “Operating System Concepts”, Wiley, 2011.

2.Linux Kernel Module Programming Guide,Ori Pomerantz,2000.

WEBSITES

1.https://www.linuxtopia.org/online_books/Linux_Kernel_Module_Programming_Guide/x279.html

2. https://www.linuxjournal.com/article/8110