OPERATING SYSTEMS MINI PROJECT (IT253)

Mouse Keyboard Driver

submitted by

Bhagyashree Bhamare (181IT111)
C. Sneha (181IT112)
Priyanka B. G. (181IT135)
Krishna Poojitha Vantakula (181IT223)

Group number-15

IV SEM B. Tech. (I.T.)

under the guidance of

Dr. B. Neelima

Dept. of I.T., NITK Surathkal

in partial fulfillment for the award of the degree

of

Bachelor of Technology

in

Information Technology

at

Department of Information Technology
National Institute of Technology Karnataka, Surathkal
June 2020

Table of Contents

1. Abstract

The project we are presenting is an implementation of a device driver for an Ubuntu system. We intended to develop the functioning of a mouse pointer using keyboard instructions provided to the system. At times we find that the click buttons of mice do not work properly or in some cases they are almost vulnerable to become faulty which can prevent us from working with our pc. In such scenarios, we can still be able to work if we can mimic the functioning of the mouse by some other means.

Considering resolution of this problem as the background, the functioning of the mouse is mimicked by certain keyboard keys provided in the driver, which could be changed to another intended key, in the device driver.

Device drivers control the interaction between the operating system and the device that they are meant for. Here the computer system is to take commands from the keyboard and insert them into kernel modules and those instructions work on the mouse pointer. The keyboard instructions fake as an external mouse with the help of device driver design.

2. Introduction

What are Device Drivers?

A device driver is a program that controls a particular type of device that is attached to your computer. There are device drivers for printers, displays, CD-ROM readers, diskette drives, and so on. When you buy an operating system, many device drivers are built into the product. However, if you later buy a new type of device that the operating system didn't anticipate, you'll have to install the new device driver. A device driver essentially converts the more general input/output instructions of the operating system to messages that the device type can understand.

The main purpose of device drivers is to provide abstraction by acting as a translator between a hardware device and the applications or operating systems that use it. Programmers can write the higher-level application code independently of whatever specific hardware the end-user is using.

How Device Drivers work?

A driver communicates with the device through the computer bus or communications subsystem to which the hardware connects. When a calling program invokes a routine in the driver, the driver issues commands to the device. Once the device sends data back to the driver, the driver may invoke routines in the original calling program. Drivers are hardware dependent and operating-system-specific.

Because of the device drivers we were able to create a device and fake the functioning of the mouse pointer with the help of commands given to the kernel module indicating what it should do. These provide an opportunity to control the mouse pointer as if some external mouse is being connected.

3. Objectives

The main objective is to develop a program that creates a virtual mouse with the help of the device drivers and resolve the problems associated with some of the improper functioning of the mouse pointer.

We create modules for this device and insert it into /dev where all the kernel modules exist and are used by the operating system to communicate with the hardware devices connected to the computer.

Because of the device drivers we would be able to create a device and fake the functioning of the mouse pointer with the help of commands given to the kernel module from the keyboard into the terminal, indicating what it should do.

4. System Specifications

Software Requirements:

- Linux based Operating systems like Ubuntu is the platform required to run the device drivers.
- Oracle VM VirtualBox
- gcc installation to compile device program written in C.
- Knowledge of device drivers

Hardware requirements:

The hardware requirements are very minimal and the device driver code can be made to run on most of the machines.

- Processor: 64-bit, Above x86
- Processor speed: 500 MHz and above
- RAM: storage space 4GB and more
- Mouse pointer which is not completely faulty

5. Methodology

A mouse object is created using the device driver program where a new data structure is created and the desired functionalities are achieved with the help of file operations and data stored in this structure. These are the file operations defined in our file:

- *int init_module(void)*
    - This is the function where the execution of the driver code starts.
    - It involves creating a mouse object for the above data structure and allocating memory in the kernel module present in the /dev file. Then allocating and registering the device is done in this.
- *ssize_t mousek_write(struct file *filp,const char *buf,size_t count,loff_t *offp)*
    - This is the function invoked whenever we try to input a command into the kernel module. This takes the input instruction and assigns it to the data of the mouse object declared before and calls for the interrupt immediately after storing the instruction. After raising the interrupt, it immediately performs the actions required by the instruction.
- *int mousek_open(struct inode *inode, struct file *filp)*
    - This function is called whenever the device tries to write something into the kernel module file. It is mainly used while giving input which involves writing the instruction into the mousek file.
- *ssize_t mousek_read(struct file *filp, char *buffer, size_t length, loff_t * offset)*
    - This is called when a device requests for the reading of kernel files.
    - But this operation is not to be supported so we just print a warning message in this case.
- *int mousek_release(struct inode *inode, struct file *filp)*
    - This function erases the data associated with the device which is assigned during the command execution. But the char device is not unregistered.

- *void cleanup_module(void)*
    - The above function is used to terminate the use of virtual devices. It unregisters the char device and also frees all the memory from the kernel module.
    - It indicates the end of device connection.

How to run the device file?

- Use *make* command to generate .ko file from written device driver file in c programming language
- Insert using *insmod mousek.ko* command
- Change to the device directory using *cd /dev*
- The major number assigned to the device can be viewed in logs using *dmesg | tail -n 5*
- Create *sudo mknod /dev/mousek c 241 0* where 241 is the major number given while inserting this module
- Provide permission to this char file *sudo chmod 666 mousek* to allow commands to be written into mousek file
- To remove the module, use *rmmod mousek*

Providing input to mousek file from keyboard

The commands are being fed into the /dev mousek char file for execution. This can be done by echo in the /dev directory.

Example:
The command issued for faking the right click operation of the mouse pointer.
echo “i wW” > mousek

- First character will tell what command to follow i : instruction sequence

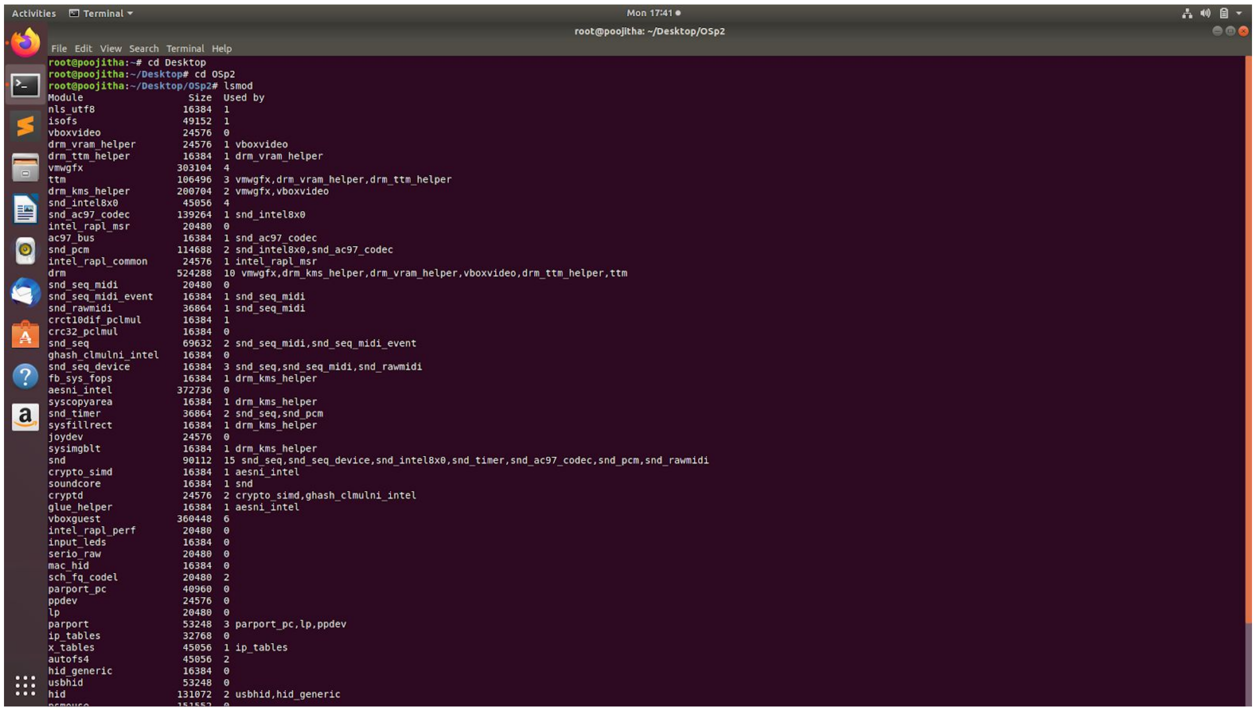
- Multiple characters accepted for this purpose could be issued in the instruction following i.

The characters accepted:

- "q" for left click down
- "Q" for left click Up
- "w" for right click down
- "W" for right click Up
- The up, down, right, left arrow keys can be used to accomplish the movement of mouse pointer

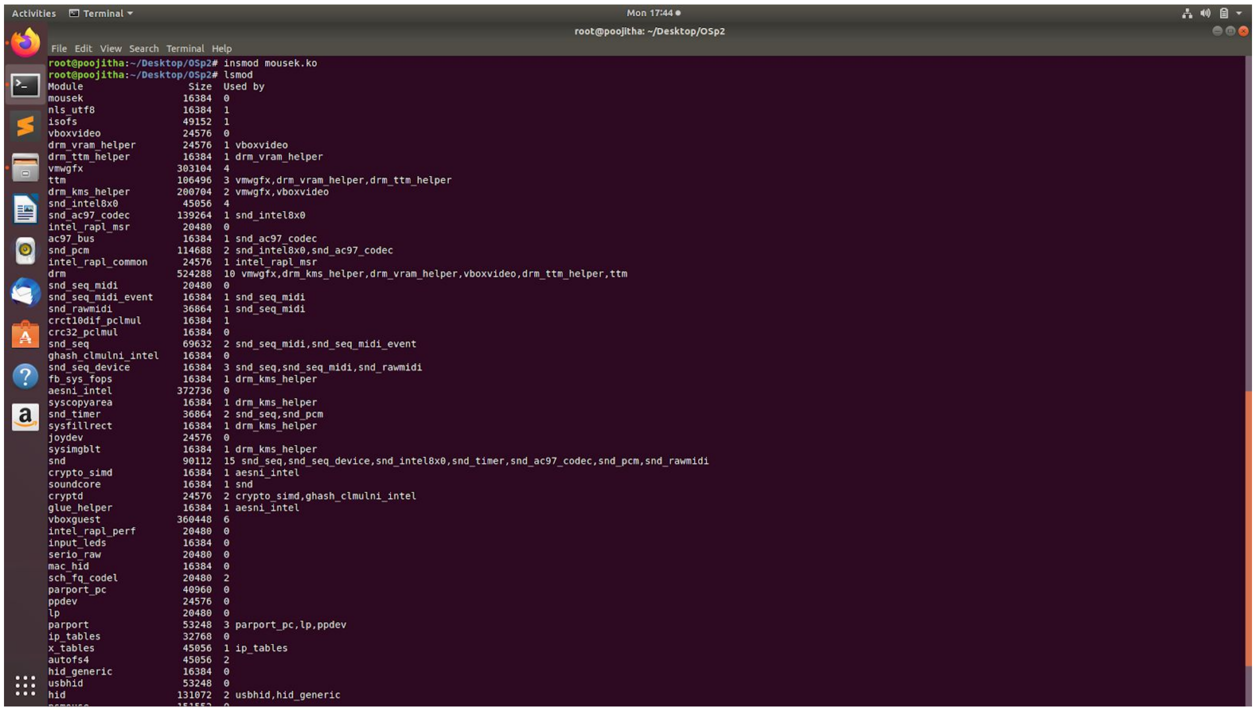
Note: We have used the above mentioned keys as a random choice. It is not mandatory to use the same keys while using the device driver. Keys can be changed manually by changing them in the device driver file.

6. Screenshots


Activities
Terminal
Mon 17:41
root@poojitha: ~/Desktop/OSp2
File Edit View Search Terminal Help

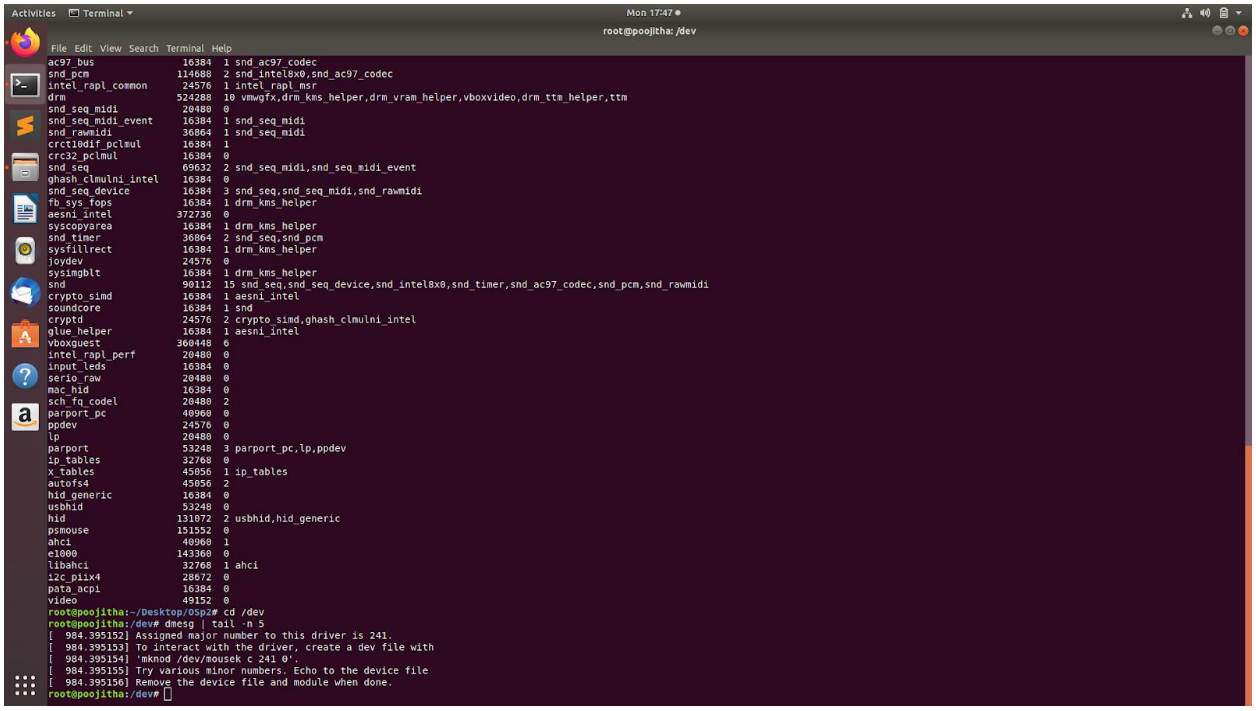
```
root@poojitha:~# cd Desktop
root@poojitha:~/Desktop# cd OSp2
root@poojitha:~/Desktop/OSp2# lsmod
Module                  Size  Used by
nls_utf8               16384  1
isofs                  49152  1
vboxvideo              24576  0
drm_vram_helper        24576  1 vboxvideo
drm_ttm_helper         16384  1 drm_vram_helper
vmwgfx                303104  4
ttm                   106496  3 vmwgfx,drm_vram_helper,drm_ttm_helper
drm_kms_helper        200704  2 vmwgfx,vboxvideo
snd_intel8x0           45056  4
snd_ac97_codec        139264  1 snd_intel8x0
intel_rapl_msr         20480  0
ac97_bus               16384  1 snd_ac97_codec
snd_pcm               114688  2 snd_intel8x0,snd_ac97_codec
intel_rapl_common      24576  1 intel_rapl_msr
drm                   524288  10 vmwgfx,drm_kms_helper,drm_vram_helper,vboxvideo,drm_ttm_helper,ttm
snd_seq_midi           20480  0
snd_seq_midi_event     16384  1 snd_seq_midi
snd_rawmidi            36864  1 snd_seq_midi
crct10dif_pclmul       16384  1
crc32_pclmul           16384  0
snd_seq                69632  2 snd_seq_midi,snd_seq_midi_event
ghash_clmulni_intel    16384  0
snd_seq_device         16384  3 snd_seq,snd_seq_midi,snd_rawmidi
fb_sys_fops            16384  1 drm_kms_helper
aesni_intel           372736  0
syscopyarea            16384  1 drm_kms_helper
snd_timer              36864  2 snd_seq,snd_pcm
sysfillrect            16384  1 drm_kms_helper
joydev                 24576  0
sysimgblt              16384  1 drm_kms_helper
snd                    90112  15 snd_seq,snd_seq_device,snd_intel8x0,snd_timer,snd_ac97_codec,snd_pcm,snd_rawmidi
crypto_simd            16384  1 aesni_intel
soundcore              16384  1 snd
cryptd                 24576  2 crypto_simd,ghash_clmulni_intel
glue_helper            16384  1 aesni_intel
vboxguest             360448  6
intel_rapl_perf        20480  0
input_leds             16384  0
serio_raw              20480  0
mac_hid                16384  0
sch_fq_codel           20480  2
parport_pc             40960  0
ppdev                  24576  0
lp                     20480  0
parport                53248  3 parport_pc,lp,ppdev
ip_tables              32768  0
x_tables               45056  1 ip_tables
autofs4                45056  2
hid_generic            16384  0
usbhid                 53248  0
hid                   131072  2 usbhid,hid_generic
```

Fig. 1: All modules initially running are listed using lsmod



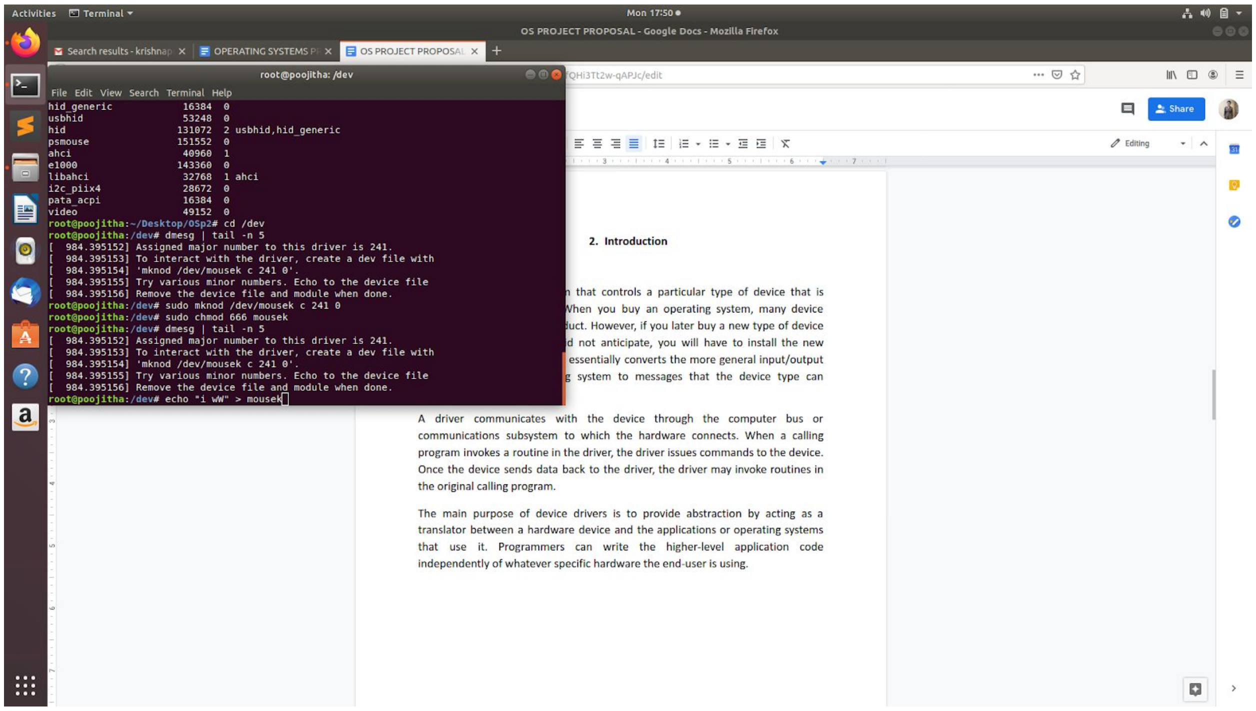
```
root@poojitha:~/Desktop/OSp2# insmod mousek.ko
root@poojitha:~/Desktop/OSp2# lsmod
Module                  Size  Used by
mousek                 16384  0
nls_utf8               16384  1
isofs                  49152  1
vboxvideo              24576  0
drm_vram_helper        24576  1 vboxvideo
drm_ttm_helper         16384  1 drm_vram_helper
vmwgfx                303104  4
ttm                   106496  3 vmwgfx,drm_vram_helper,drm_ttm_helper
drm_kms_helper        200704  2 vmwgfx,vboxvideo
snd_intel8x0           45056  4
snd_ac97_codec        139264  1 snd_intel8x0
intel_rapl_msr         20480  0
ac97_bus               16384  1 snd_ac97_codec
snd_pcm               114688  2 snd_intel8x0,snd_ac97_codec
intel_rapl_common      24576  1 intel_rapl_msr
drm                   524288  10 vmwgfx,drm_kms_helper,drm_vram_helper,vboxvideo,drm_ttm_helper,ttm
snd_seq_midi           20480  0
snd_seq_midi_event     16384  1 snd_seq_midi
snd_rawmidi            36864  1 snd_seq_midi
crct10dif_pclmul       16384  1
crc32_pclmul           16384  0
snd_seq                69632  2 snd_seq_midi,snd_seq_midi_event
ghash_clmulni_intel    16384  0
snd_seq_device         16384  3 snd_seq,snd_seq_midi,snd_rawmidi
fb_sys_fops            16384  1 drm_kms_helper
aesni_intel           372736  0
syscopyarea            16384  1 drm_kms_helper
snd_timer              36864  2 snd_seq,snd_pcm
sysfillrect            16384  1 drm_kms_helper
joydev                 24576  0
sysimgblt              16384  1 drm_kms_helper
snd                    90112  15 snd_seq,snd_seq_device,snd_intel8x0,snd_timer,snd_ac97_codec,snd_pcm,snd_rawmidi
crypto_simd            16384  1 aesni_intel
soundcore              16384  1 snd
cryptd                 24576  2 crypto_simd,ghash_clmulni_intel
glue_helper            16384  1 aesni_intel
vboxguest             360448  6
intel_rapl_perf        20480  0
input_leds             16384  0
serio_raw              20480  0
mac_hid                16384  0
sch_fq_codel           20480  2
parport_pc             40960  0
ppdev                  24576  0
lp                     20480  0
parport                53248  3 parport_pc,lp,ppdev
ip_tables              32768  0
x_tables               45056  1 ip_tables
autofs4                45056  2
hid_generic            16384  0
usbhid                 53248  0
hid                   131072  2 usbhid,hid_generic
```

Fig. 2: Inserted mousek module using insmod mousek.ko (top-most module)



```
ac97_bus               16384  1 snd_ac97_codec
snd_pcm               114688  2 snd_intel8x0,snd_ac97_codec
intel_rapl_common      24576  1 intel_rapl_msr
drm                   524288  10 vmwgfx,drm_kms_helper,drm_vram_helper,vboxvideo,drm_ttm_helper,ttm
snd_seq_midi           20480  0
snd_seq_midi_event     16384  1 snd_seq_midi
snd_rawmidi            36864  1 snd_seq_midi
crct10dif_pclmul       16384  1
crc32_pclmul           16384  0
snd_seq                69632  2 snd_seq_midi,snd_seq_midi_event
ghash_clmulni_intel    16384  0
snd_seq_device         16384  3 snd_seq,snd_seq_midi,snd_rawmidi
fb_sys_fops            16384  1 drm_kms_helper
aesni_intel           372736  0
syscopyarea            16384  1 drm_kms_helper
snd_timer              36864  2 snd_seq,snd_pcm
sysfillrect            16384  1 drm_kms_helper
joydev                 24576  0
sysimgblt              16384  1 drm_kms_helper
snd                    90112  15 snd_seq,snd_seq_device,snd_intel8x0,snd_timer,snd_ac97_codec,snd_pcm,snd_rawmidi
crypto_simd            16384  1 aesni_intel
soundcore              16384  1 snd
cryptd                 24576  2 crypto_simd,ghash_clmulni_intel
glue_helper            16384  1 aesni_intel
vboxguest             360448  6
intel_rapl_perf        20480  0
input_leds             16384  0
serio_raw              20480  0
mac_hid                16384  0
sch_fq_codel           20480  2
parport_pc             40960  0
ppdev                  24576  0
lp                     20480  0
parport                53248  3 parport_pc,lp,ppdev
ip_tables              32768  0
x_tables               45056  1 ip_tables
autofs4                45056  2
hid_generic            16384  0
usbhid                 53248  0
hid                   131072  2 usbhid,hid_generic
psmouse               151552  0
ahci                   40960  1
e1000                 143360  0
libahci                32768  1 ahci
i2c_piix4              28672  0
pata_acpi              16384  0
video                  49152  0
root@poojitha:~/Desktop/OSp2# cd /dev
root@poojitha:/dev# dmesg | tail -n 5
[  984.395152] Assigned major number to this driver is 241.
[  984.395153] To interact with the driver, create a dev file with
[  984.395154] 'mknod /dev/mousek c 241 0'.
[  984.395155] Try various minor numbers. Echo to the device file
[  984.395156] Remove the device file and module when done.
root@poojitha:/dev# 
```

Fig. 3: Major number 241 allocated to the device developed



Activities
Terminal
Mon 17:50
OS PROJECT PROPOSAL - Google Docs - Mozilla Firefox
Search results - krishnap
OPERATING SYSTEMS P
OS PROJECT PROPOSAL
root@poojitha: /dev
File Edit View Search Terminal Help
Share
Editing

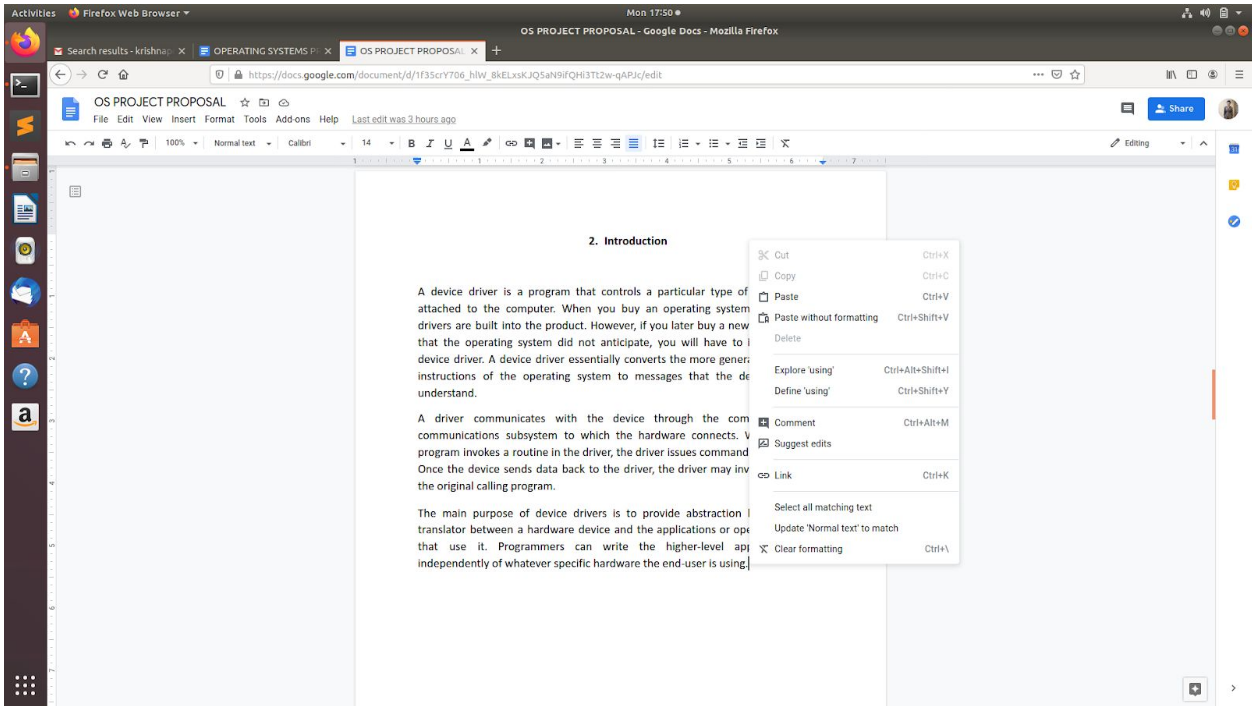
```
hid_generic            16384  0
usbhid                 53248  0
hid                   131072  2 usbhid,hid_generic
psmouse               151552  0
ahci                   40960  1
e1000                 143360  0
libahci                32768  1 ahci
i2c_piix4              28672  0
pata_acpi              16384  0
video                  49152  0
root@poojitha:~/Desktop/OSp2# cd /dev
root@poojitha:/dev# dmesg | tail -n 5
[  984.395152] Assigned major number to this driver is 241.
[  984.395153] To interact with the driver, create a dev file with
[  984.395154] 'mknod /dev/mousek c 241 0'.
[  984.395155] Try various minor numbers. Echo to the device file
[  984.395156] Remove the device file and module when done.
root@poojitha:/dev# sudo mknod /dev/mousek c 241 0
root@poojitha:/dev# sudo chmod 666 mousek
root@poojitha:/dev# dmesg | tail -n 5
[  984.395152] Assigned major number to this driver is 241.
[  984.395153] To interact with the driver, create a dev file with
[  984.395154] 'mknod /dev/mousek c 241 0'.
[  984.395155] Try various minor numbers. Echo to the device file
[  984.395156] Remove the device file and module when done.
root@poojitha:/dev# echo "i wW" > mousek
```

2. Introduction

A driver communicates with the device through the computer bus or communications subsystem to which the hardware connects. When a calling program invokes a routine in the driver, the driver issues commands to the device. Once the device sends data back to the driver, the driver may invoke routines in the original calling program.

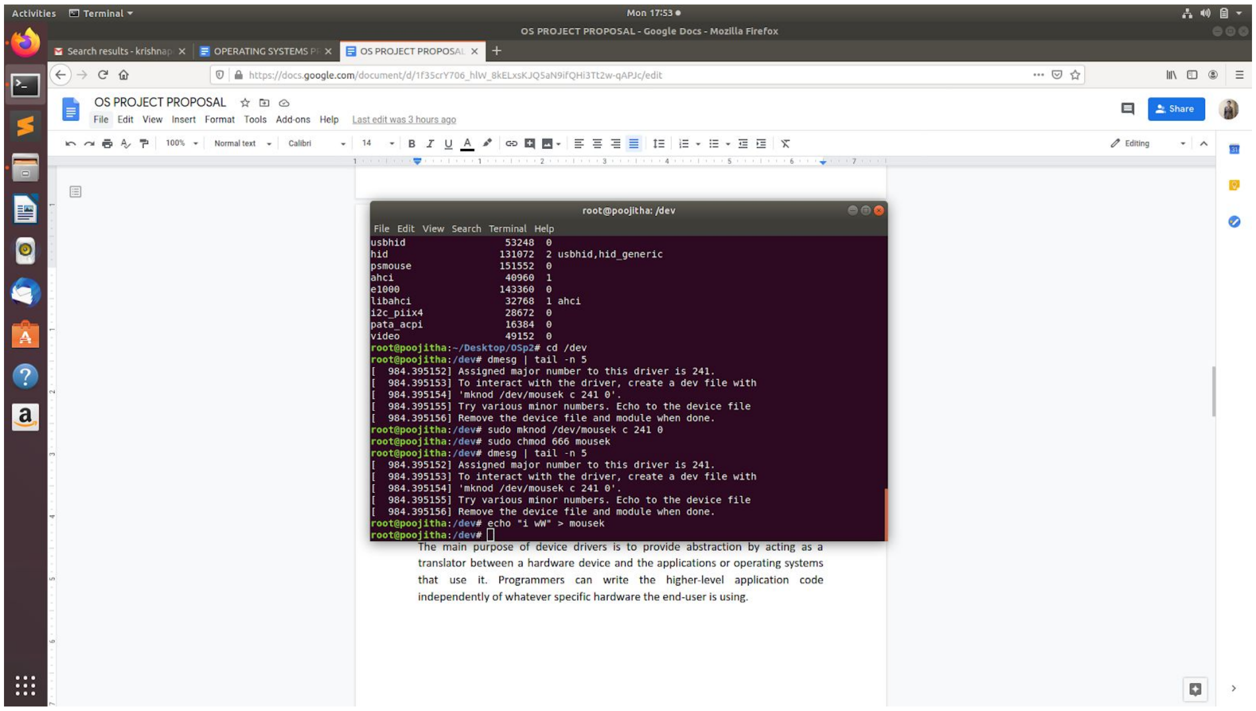
The main purpose of device drivers is to provide abstraction by acting as a translator between a hardware device and the applications or operating systems that use it. Programmers can write the higher-level application code independently of whatever specific hardware the end-user is using.

Fig 4: Command for right click entered into the terminal



Activities
Firefox Web Browser
Mon 17:50
OS PROJECT PROPOSAL - Google Docs - Mozilla Firefox
OS PROJECT PROPOSAL
File Edit View Insert Format Tools Add-ons Help Last edit was 3 hours ago
100% Normal text Calibri 14
Share
Editing
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Paste without formatting Ctrl+Shift+V
Delete
Explore 'using' Ctrl+Alt+Shift+I
Define 'using' Ctrl+Shift+Y
Comment Ctrl+Alt+M
Suggest edits
Link Ctrl+K
Select all matching text
Update 'Normal text' to match
Clear formatting Ctrl+\

Fig. 5: Output produced on working of the command associated with right click

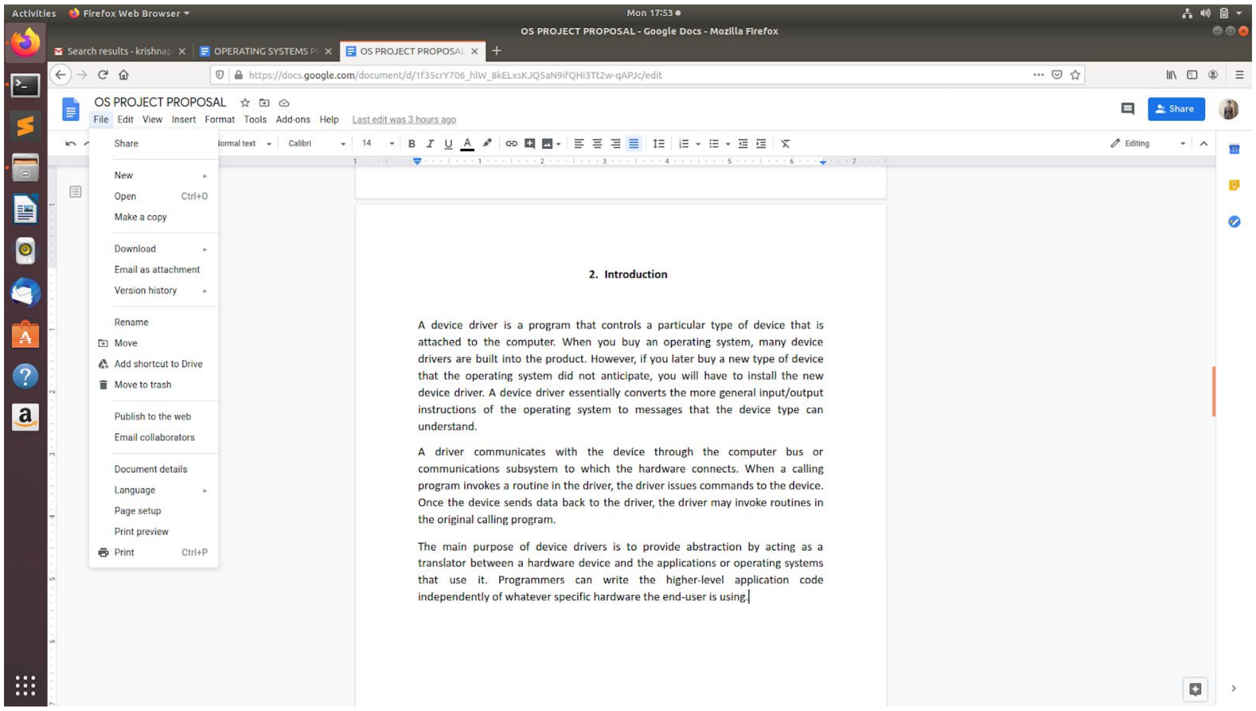


OS PROJECT PROPOSAL - Google Docs - Mozilla Firefox
OS PROJECT PROPOSAL
File Edit View Insert Format Tools Add-ons Help
root@poojitha: /dev
File Edit View Search Terminal Help

```
usbhid                 53248  0
hid                   131072  2 usbhid,hid_generic
psmouse               151552  0
ahci                   40960  1
e1000                 143360  0
libahci                32768  1 ahci
i2c_piix4              28672  0
pata_acpi              16384  0
video                  49152  0
root@poojitha:~/Desktop/OSp2# cd /dev
root@poojitha:/dev# dmesg | tail -n 5
[  984.395152] Assigned major number to this driver is 241.
[  984.395153] To interact with the driver, create a dev file with
[  984.395154] 'mknod /dev/mousek c 241 0'.
[  984.395155] Try various minor numbers. Echo to the device file
[  984.395156] Remove the device file and module when done.
root@poojitha:/dev# sudo mknod /dev/mousek c 241 0
root@poojitha:/dev# sudo chmod 666 mousek
root@poojitha:/dev# dmesg | tail -n 5
[  984.395152] Assigned major number to this driver is 241.
[  984.395153] To interact with the driver, create a dev file with
[  984.395154] 'mknod /dev/mousek c 241 0'.
[  984.395155] Try various minor numbers. Echo to the device file
[  984.395156] Remove the device file and module when done.
root@poojitha:/dev# echo "i wW" > mousek
root@poojitha:/dev# 
```

The main purpose of device drivers is to provide abstraction by acting as a translator between a hardware device and the applications or operating systems that use it. Programmers can write the higher-level application code independently of whatever specific hardware the end-user is using.

Fig. 6: Mouse pointer placed on the file menu bar and command for left click is entered



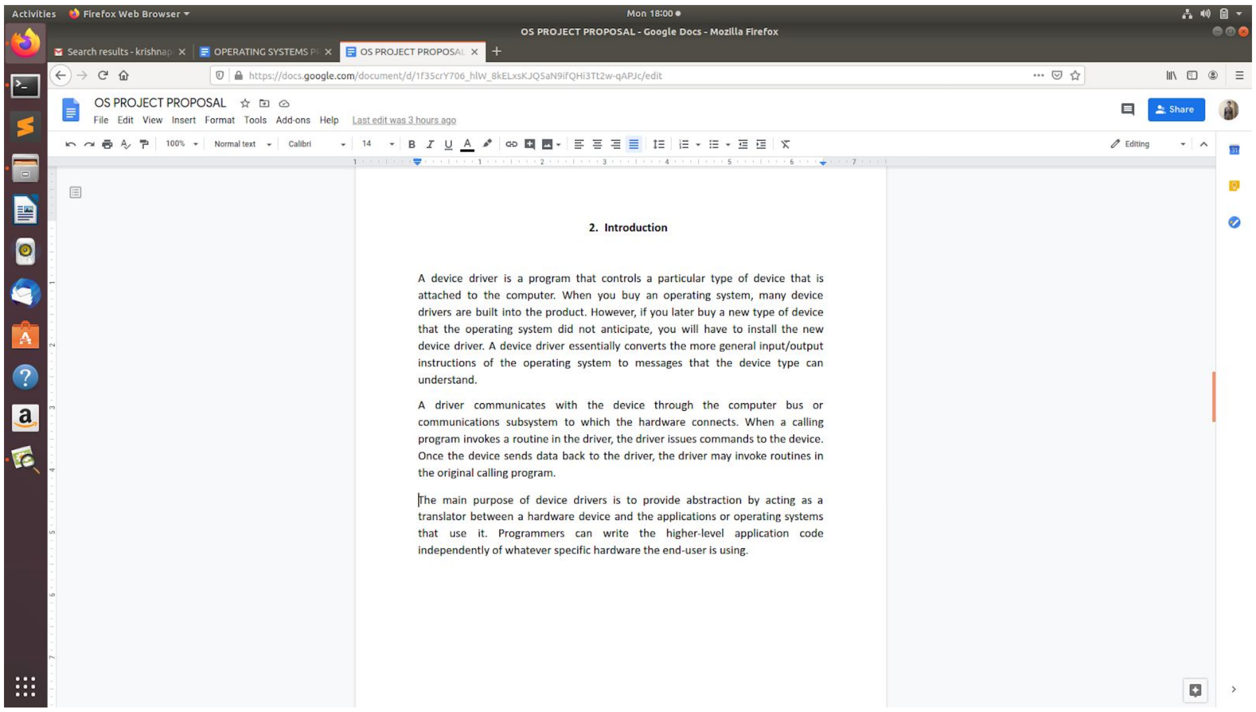
2. Introduction

A device driver is a program that controls a particular type of device that is attached to the computer. When you buy an operating system, many device drivers are built into the product. However, if you later buy a new type of device that the operating system did not anticipate, you will have to install the new device driver. A device driver essentially converts the more general input/output instructions of the operating system to messages that the device type can understand.

A driver communicates with the device through the computer bus or communications subsystem to which the hardware connects. When a calling program invokes a routine in the driver, the driver issues commands to the device. Once the device sends data back to the driver, the driver may invoke routines in the original calling program.

The main purpose of device drivers is to provide abstraction by acting as a translator between a hardware device and the applications or operating systems that use it. Programmers can write the higher-level application code independently of whatever specific hardware the end-user is using.

Fig. 7: Output produced on working of the command associated with left click


Activities
Firefox Web Browser
Mon 18:00
OS PROJECT PROPOSAL - Google Docs - Mozilla Firefox
Search results - krishnap
OPERATING SYSTEMS PR
OS PROJECT PROPOSAL
https://docs.google.com/document/d/1f35crY706_hlW_8kELxsKJQ5aN9ifQHi3Tt2w-qAPJc/edit
OS PROJECT PROPOSAL
File Edit View Insert Format Tools Add-ons Help
Last edit was 3 hours ago
Share
100%
Normal text
Calibri
14
Editing

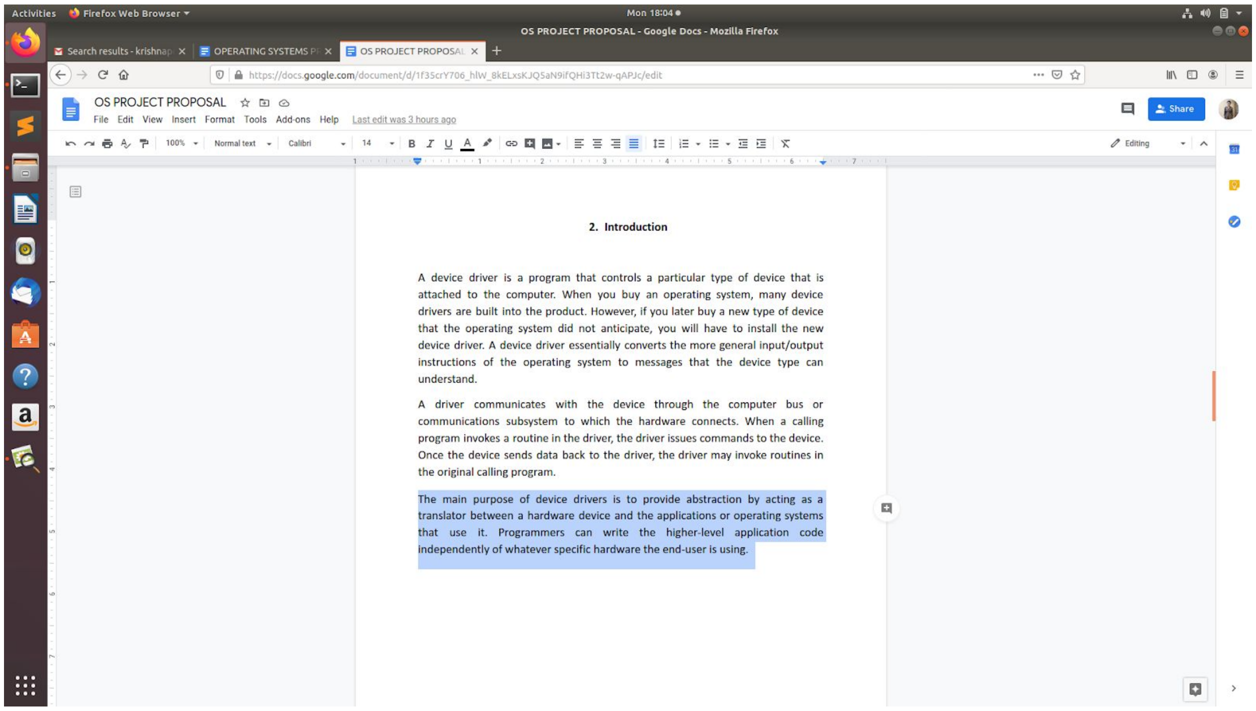
2. Introduction

A device driver is a program that controls a particular type of device that is attached to the computer. When you buy an operating system, many device drivers are built into the product. However, if you later buy a new type of device that the operating system did not anticipate, you will have to install the new device driver. A device driver essentially converts the more general input/output instructions of the operating system to messages that the device type can understand.

A driver communicates with the device through the computer bus or communications subsystem to which the hardware connects. When a calling program invokes a routine in the driver, the driver issues commands to the device. Once the device sends data back to the driver, the driver may invoke routines in the original calling program.

The main purpose of device drivers is to provide abstraction by acting as a translator between a hardware device and the applications or operating systems that use it. Programmers can write the higher-level application code independently of whatever specific hardware the end-user is using.

Fig. 8: Mouse cursor placed at the beginning of third paragraph in docs file


Activities
Firefox Web Browser
Mon 18:04
OS PROJECT PROPOSAL - Google Docs - Mozilla Firefox
Search results - krishnap
OPERATING SYSTEMS PR
OS PROJECT PROPOSAL
https://docs.google.com/document/d/1f35crY706_hlW_8kELxsKJQ5aN9ifQHi3Tt2w-qAPJc/edit
OS PROJECT PROPOSAL
File Edit View Insert Format Tools Add-ons Help
Last edit was 3 hours ago
Share
100%
Normal text
Calibri
14
Editing

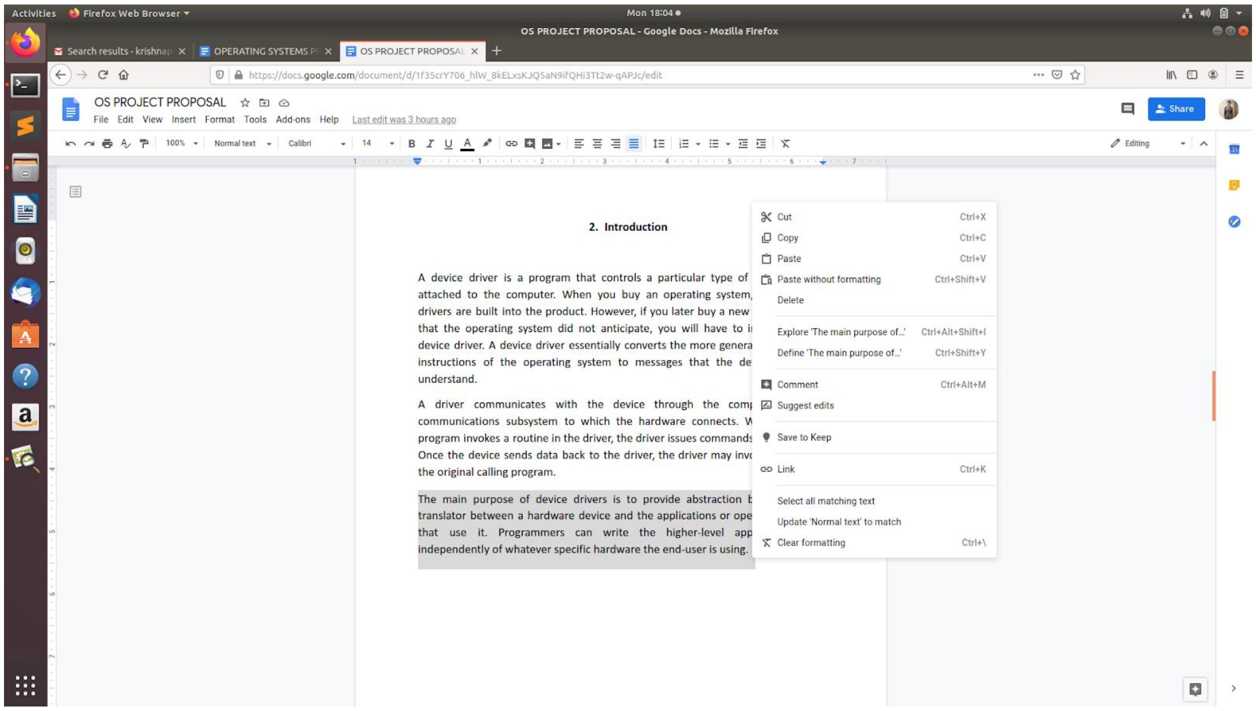
2. Introduction

A device driver is a program that controls a particular type of device that is attached to the computer. When you buy an operating system, many device drivers are built into the product. However, if you later buy a new type of device that the operating system did not anticipate, you will have to install the new device driver. A device driver essentially converts the more general input/output instructions of the operating system to messages that the device type can understand.

A driver communicates with the device through the computer bus or communications subsystem to which the hardware connects. When a calling program invokes a routine in the driver, the driver issues commands to the device. Once the device sends data back to the driver, the driver may invoke routines in the original calling program.

The main purpose of device drivers is to provide abstraction by acting as a translator between a hardware device and the applications or operating systems that use it. Programmers can write the higher-level application code independently of whatever specific hardware the end-user is using.

Fig. 9: Paragraph selected using command for left click followed by pressing the shift and page down keys


Activities
Firefox Web Browser
Mon 18:04
OS PROJECT PROPOSAL - Google Docs - Mozilla Firefox
Search results - krishnap
OPERATING SYSTEMS P
OS PROJECT PROPOSAL
https://docs.google.com/document/d/1f35crY706_hlW_8kELxsKJQ5aN9ifQHi3Tt2w-qAPJc/edit
OS PROJECT PROPOSAL
File Edit View Insert Format Tools Add-ons Help
Last edit was 3 hours ago
Share
100%
Normal text
Calibri
14
Editing
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Paste without formatting Ctrl+Shift+V
Delete
Explore 'The main purpose of...' Ctrl+Alt+Shift+I
Define 'The main purpose of...' Ctrl+Shift+Y
Comment Ctrl+Alt+M
Suggest edits
Save to Keep
Link Ctrl+K
Select all matching text
Update 'Normal text' to match
Clear formatting Ctrl+\

2. Introduction

A device driver is a program that controls a particular type of
attached to the computer. When you buy an operating system,
drivers are built into the product. However, if you later buy a new
that the operating system did not anticipate, you will have to i
device driver. A device driver essentially converts the more genera
instructions of the operating system to messages that the de
understand.

A driver communicates with the device through the comp
communications subsystem to which the hardware connects. W
program invokes a routine in the driver, the driver issues commands
Once the device sends data back to the driver, the driver may invo
the original calling program.

The main purpose of device drivers is to provide abstraction b
translator between a hardware device and the applications or ope
that use it. Programmers can write the higher-level app
independently of whatever specific hardware the end-user is using.

Fig. 10: Command issued for right click along with mouse pointer on selected paragraph gives the following output, where options could be selected using page up and page down keys.

7. Timeline

Week 1

- Understanding the concepts of device drivers and its types.
- Learning to code for taking the input from the keyboard.

Week 2

- Learning about the functioning and working of a well functioning mouse pointer.
- Understanding the usage and finding out how to write a makefile.

Week 3

Determining the file operations necessary for creating the intended functionality.

Week 4

Creating the new mouse structure and developing insert,clean modules.

Week 5

- Creating the remaining functions and makefile.
- Dealing with all issues encountered while running the entire process and obtaining desired functionality.

8. Challenges Faced

Throughout the development of project we faced few challenges, which are listed below:

- Figuring out the operations corresponding to click of mouse pointer and combining them with keyboard instructions is the critical part of our project.
- Resolving the errors caused by writing incorrect makefile in the beginning made us refer to quite a few resources and finally could write appropriate one.
- As beginners to this new concept, we initially tried to insert the module using the general terminal and frequently obtained the error of access denied.
- We took some effort to resolve the insertion of the driver into the kernel module via the terminal opened as root.

9. References

Given below are the tutorials and some online sources which helped us in progressing with the project implementation.

- https://m.youtube.com/playlist?list=PL1zwAXk5ZrWKxLyCq73lzFn3oSLruM750
- https://www.apriorit.com/dev-blog/195-simple-driver-for-linux-os
- https://www.tutorialspoint.com/ubuntu/ubuntu_device_drivers.htm
- https://m.youtube.com/playlist?list=PL2GL6HVUQAuksbptmKC7X4zruZlIl59is

10. Conclusion

Through this project we tried to find out a solution to eliminate the problem of the mouse whose click functionality turns out faulty, using the concept of device drivers, by writing our own character device driver. In Spite of some malfunctioning of the mouse, we could use the keyboard and perform the same operations done by the mouse.

While working with the project, we understood the utility of device drivers which are important for the proper functioning of our computers. The project gave all of us a good exposure to the concept of device drivers which we found to be an interesting and important topic in the field of Operating Systems.