INT - 301: OPEN SOURCE TECHNOGOLIES

A Project Report

Submitted in partial fulfilment for the requirements of the award of the degree of

Bachelor of Technology

Computer Science Engineering (Hons.)

Lovely Professional University
Phagwara, Punjab.

Submitted by:

Name of the Student: Mohammad Rasool

Registration Number: 11901876

Roll No: 03

Section: KE059

CA-3

Q42. Using desired Open Source Software display an overview of all the hardware and operating system detail; also do live monitoring to show the temperature and current usage of various hardware components.

To display an overview of hardware and operating system details in Linux, you can use the following open-source software:

1. lshw

- It is a command-line utility that provides detailed information about the hardware in our system.

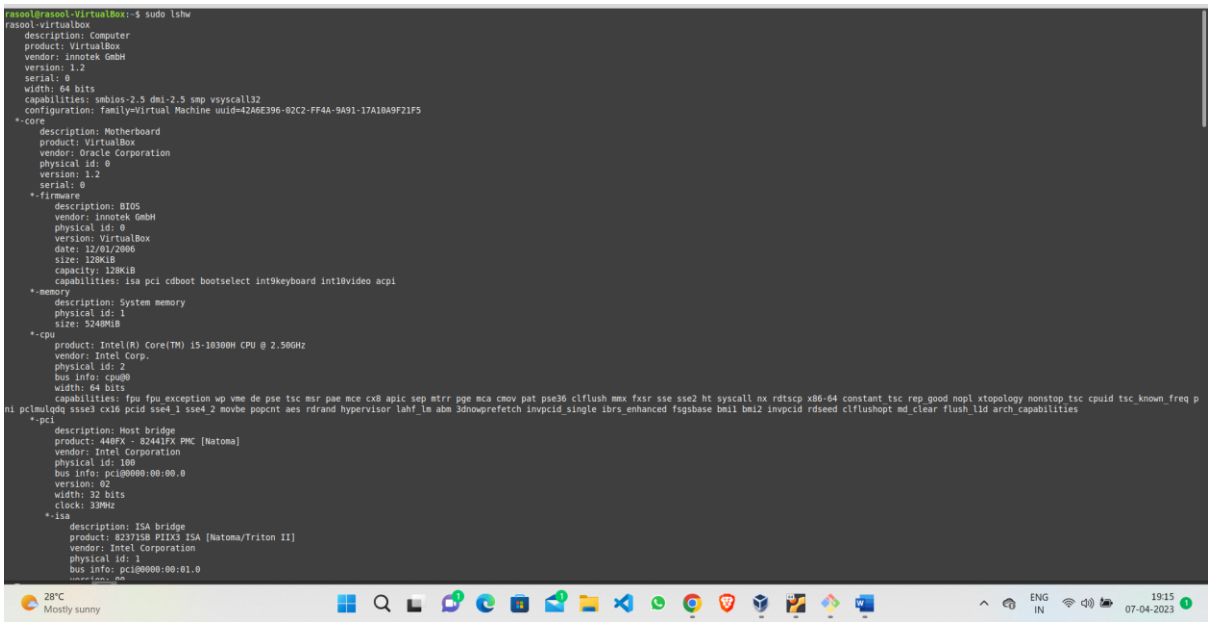
To install lshw on Ubuntu or Debian-based systems, we can use the following command:

sudo apt-get install lshw

Once installed, we can use the following command to display the hardware details:

We can use this command:

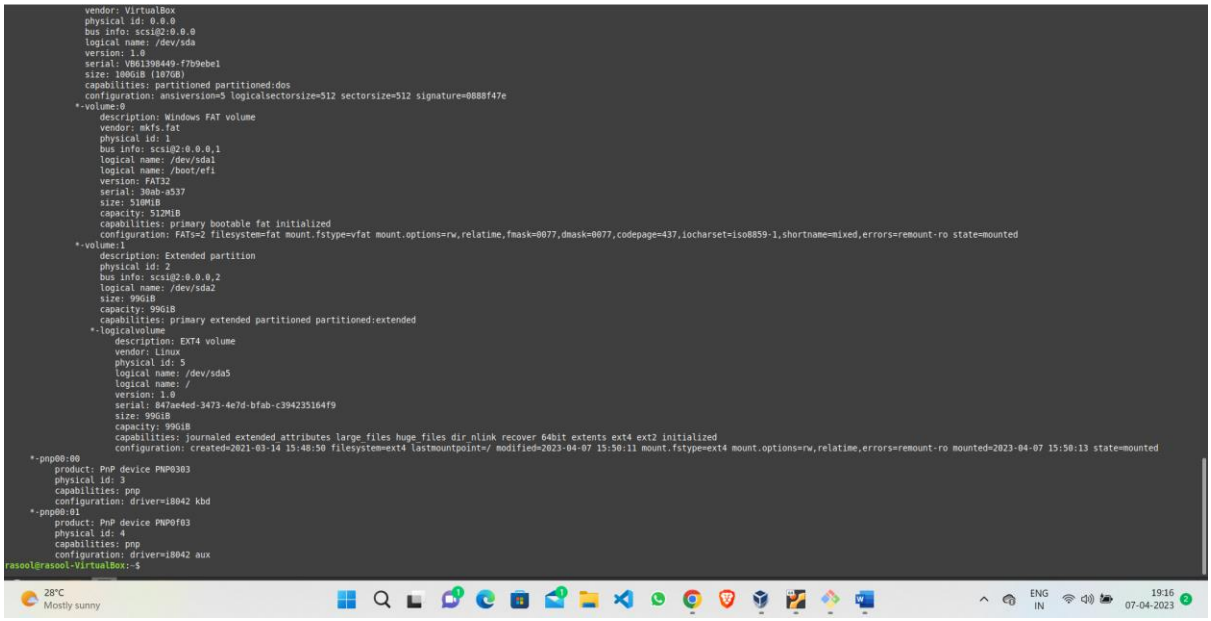
sudo lshw



```
*-isa
     description: ISA bridge
     product: 82371SB PIIX3 ISA [Natoma/Triton II]
     vendor: Intel Corporation
     physical id: 1
     bus info: pci@0000:00:01.0
     version: 00
     width: 32 bits
     clock: 33MHz
     capabilities: isa bus_master
     configuration: latency=0
*-ide
     description: IDE interface
     product: 82371AB/EB/MB PIIX4 IDE
     vendor: Intel Corporation
     physical id: 1.1
     bus info: pci@0000:00:01.1
     logical name: scsi1
     version: 01
     width: 32 bits
     clock: 33MHz
     capabilities: ide isa_compat_mode pci_native_mode bus_master emulated
     configuration: driver=ata_piix latency=64
     resources: irq:0 ioport:1f0(size=8) ioport:3f6 ioport:170(size=8) ioport:376 ioport:d000(size=16)
   *-cdrom
        description: DVD reader
        product: CD-ROM
        vendor: VBOX
        physical id: 0.0.0
        bus info: scsi@1:0.0.0
        logical name: /dev/cdrom
        logical name: /dev/dvd
        logical name: /dev/sr0
        logical name: /media/rasool/VBox_GAs_7.0.6
        version: 1.0
        capabilities: removable audio dvd
        configuration: ansiversion=5 mount.fstype=iso9660 mount.options=ro,nosuid,nodev,relatime,nojoliet,check=s,map=n,blocksize=2048,uid=1000,gid=1000,dmode=500,fmode=400 state=mounted status=ready
      *-medium
           physical id: 0
           logical name: /dev/cdrom
           logical name: /media/rasool/VBox_GAs_7.0.6
           configuration: mount.fstype=iso9660 mount.options=ro,nosuid,nodev,relatime,nojoliet,check=s,map=n,blocksize=2048,uid=1000,gid=1000,dmode=500,fmode=400 state=mounted
*-display
     description: VGA compatible controller
     product: SVGA II Adapter
     vendor: VMware
     physical id: 2
     bus info: pci@0000:00:02.0
     version: 00
     width: 32 bits
     clock: 33MHz
     capabilities: vga_controller bus_master rom
     configuration: driver=vmwgfx latency=64
```

```
*-network
     description: Ethernet interface
     product: 82540EM Gigabit Ethernet Controller
     vendor: Intel Corporation
     physical id: 3
     bus info: pci@0000:00:03.0
     logical name: enp0s3
     version: 02
     serial: 08:00:27:b4:eb:3f
     size: 1Gbit/s
     capacity: 1Gbit/s
     width: 32 bits
     clock: 66MHz
     capabilities: pm pcix bus_master cap_list ethernet physical tp 10bt 10bt-fd 100bt 100bt-fd 1000bt-fd autonegotiation
     configuration: autonegotiation=on broadcast=yes driver=e1000 driverversion=7.3.21-k8-NAPI duplex=full ip=10.0.2.15 latency=64 link=yes mingnt=255 multicast=yes port=twisted pair speed=1Gbit/s
     resources: irq:19 memory:f0200000-f021ffff ioport:d020(size=8)
*-generic
     description: System peripheral
     product: VirtualBox Guest Service
     vendor: InnoTek Systemberatung GmbH
     physical id: 4
     bus info: pci@0000:00:04.0
     version: 00
     width: 32 bits
     clock: 33MHz
     configuration: driver=vboxguest latency=0
     resources: irq:20 ioport:d040(size=32) memory:f0400000-f07fffff memory:f0800000-f0803fff
*-multimedia
     description: Multimedia audio controller
     product: 82801AA AC'97 Audio Controller
     vendor: Intel Corporation
     physical id: 5
     bus info: pci@0000:00:05.0
     version: 01
     width: 32 bits
     clock: 33MHz
     capabilities: bus_master
     configuration: driver=snd_intel8x0 latency=64
     resources: irq:21 ioport:d100(size=256) ioport:d200(size=64)
*-usb
     description: USB controller
     product: KeyLargo/Intrepid USB
     vendor: Apple Inc.
     physical id: 6
     bus info: pci@0000:00:06.0
     version: 00
     width: 32 bits
     clock: 33MHz
     capabilities: ohci bus_master cap_list
     configuration: driver=ohci-pci latency=64
     resources: irq:22 memory:f0804000-f0804fff
   *-usbhost
        product: OHCI PCI host controller
```

```
   *-usbhost
        product: OHCI PCI host controller
        vendor: Linux 5.4.0-58-generic ohci_hcd
        physical id: 1
        bus info: usb@1
        logical name: usb1
        version: 5.04
        capabilities: usb-1.10
        configuration: driver=hub slots=12 speed=12Mbit/s
      *-usb
           description: Human interface device
           product: USB Tablet
           vendor: VirtualBox
           physical id: 1
           bus info: usb@1:1
           version: 1.00
           capabilities: usb-1.10
           configuration: driver=usbhid maxpower=100mA speed=12Mbit/s
*-bridge
     description: Bridge
     product: 82371AB/EB/MB PIIX4 ACPI
     vendor: Intel Corporation
     physical id: 7
     bus info: pci@0000:00:07.0
     version: 08
     width: 32 bits
     clock: 33MHz
     capabilities: bridge
     configuration: driver=piix4_smbus latency=0
     resources: irq:9
*-sata
     description: SATA controller
     product: 82801HM/HEM (ICH8M/ICH8M-E) SATA Controller [AHCI mode]
     vendor: Intel Corporation
     physical id: d
     bus info: pci@0000:00:0d.0
     logical name: scsi2
     version: 02
     width: 32 bits
     clock: 33MHz
     capabilities: sata pm ahci_1.0 bus_master cap_list emulated
     configuration: driver=ahci latency=64
     resources: irq:21 ioport:d240(size=8) ioport:d248(size=4) ioport:d250(size=8) ioport:d258(size=4) ioport:d260(size=16) memory:f0806000-f0807fff
   *-disk
        description: ATA Disk
        product: VBOX HARDDISK
        vendor: VirtualBox
        physical id: 0.0.0
        bus info: scsi@2:0.0.0
        logical name: /dev/sda
        version: 1.0
        serial: VB61398449-f7b9ebe1
        size: 100GiB (107GB)
```



2. Neofetch

- It is a command-line system information tool that displays the operating system, CPU, GPU, RAM, and other system details in an aesthetically pleasing way.

To install Neofetch on Ubuntu or Debian-based systems, use the following command:

sudo apt-get install neofetch

Once installed, we can use the following command to display the system details:

sudo neofetch

To monitor the temperature and current usage of various hardware components in Linux, you can use the following open-source software:

3. lm-sensors

- It is a command-line utility that provides information about the temperature, voltage, and fan speed sensors in your system.

To install lm-sensors on Ubuntu or Debian-based systems, use the following command:

sudo apt-get install lm-sensors

Once installed, we can use the following command to detect the sensors in our system:

sudo sensors-detect

```
rasool@rasool-VirtualBox:~$ sudo sensors-detect
# sensors-detect version 3.6.0
# System: innotek GmbH VirtualBox [1.2]
# Board: Oracle Corporation VirtualBox
# Kernel: 5.4.0-58-generic x86_64
# Processor: Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz (6/165/2)

This program will help you determine which kernel modules you need
to load to use lm_sensors most effectively. It is generally safe
and recommended to accept the default answers to all questions,
unless you know what you're doing.

Some south bridges, CPUs or memory controllers contain embedded sensors.
Do you want to scan for them? This is totally safe. (YES/no): y
Silicon Integrated Systems SIS5595...                       No
VIA VT82C686 Integrated Sensors...                          No
VIA VT8231 Integrated Sensors...                            No
AMD K8 thermal sensors...                                   No
AMD Family 10h thermal sensors...                           No
AMD Family 11h thermal sensors...                           No
AMD Family 12h and 14h thermal sensors...                   No
AMD Family 15h thermal sensors...                           No
AMD Family 16h thermal sensors...                           No
AMD Family 17h thermal sensors...                           No
AMD Family 15h power sensors...                             No
AMD Family 16h power sensors...                             No
Hygon Family 18h thermal sensors...                         No
Intel digital thermal sensor...                             No
Intel AMB FB-DIMM thermal sensor...                         No
Intel 5500/5520/X58 thermal sensor...                       No
VIA C7 thermal sensor...                                    No
VIA Nano thermal sensor...                                  No

Some Super I/O chips contain embedded sensors. We have to write to
standard I/O ports to probe them. This is usually safe.
Do you want to scan for Super I/O sensors? (YES/no): y
Probing for Super-I/O at 0x2e/0x2f
Trying family `National Semiconductor/ITE'...               No
Trying family `SMSC'...                                     No
Trying family `VIA/Winbond/Nuvoton/Fintek'...               No
Trying family `ITE'...                                      No
Probing for Super-I/O at 0x4e/0x4f
Trying family `National Semiconductor/ITE'...               No
Trying family `SMSC'...                                     No
Trying family `VIA/Winbond/Nuvoton/Fintek'...               No
Trying family `ITE'...                                      No

Some systems (mainly servers) implement IPMI, a set of common interfaces
through which system health data may be retrieved, amongst other things.
We first try to get the information from SMBIOS. If we don't find it
there, we have to read from arbitrary I/O ports to probe for such
interfaces. This is normally safe. Do you want to scan for IPMI
interfaces? (YES/no): y
```

```
Some systems (mainly servers) implement IPMI, a set of common interfaces
through which system health data may be retrieved, amongst other things.
We first try to get the information from SMBIOS. If we don't find it
there, we have to read from arbitrary I/O ports to probe for such
interfaces. This is normally safe. Do you want to scan for IPMI
interfaces? (YES/no): y
Probing for `IPMI BMC KCS' at 0xca0...                      No
Probing for `IPMI BMC SMIC' at 0xca8...                     No

Some hardware monitoring chips are accessible through the ISA I/O ports.
We have to write to arbitrary I/O ports to probe them. This is usually
safe though. Yes, you do have ISA I/O ports even if you do not have any
ISA slots! Do you want to scan the ISA I/O ports? (YES/no): y
Probing for `National Semiconductor LM78' at 0x290...       No
Probing for `National Semiconductor LM79' at 0x290...       No
Probing for `Winbond W83781D' at 0x290...                   No
Probing for `Winbond W83782D' at 0x290...                   No

Lastly, we can probe the I2C/SMBus adapters for connected hardware
monitoring devices. This is the most risky part, and while it works
reasonably well on most systems, it has been reported to cause trouble
on some systems.
Do you want to probe the I2C/SMBus adapters now? (YES/no): y
Using driver `i2c-piix4' for device 0000:00:07.0: Intel 82371AB PIIX4 ACPI

Next adapter: SMBus PIIX4 adapter at 4100 (i2c-0)
Do you want to scan it? (YES/no/selectively): y

Sorry, no sensors were detected.
Either your system has no sensors, or they are not supported, or
they are connected to an I2C or SMBus adapter that is not
supported. If you find out what chips are on your board, check
https://hwmon.wiki.kernel.org/device_support_status for driver status.
```

After detecting the sensors, you can use the following command to display the sensor readings:

sudo sensors

```
rasool@rasool-VirtualBox:~$ sensors
BAT0-acpi-0
Adapter: ACPI interface
in0:          10.00 V

rasool@rasool-VirtualBox:~$
```

4. htop

- It is a command-line utility that provides real-time monitoring of CPU usage, memory usage, and other system details.

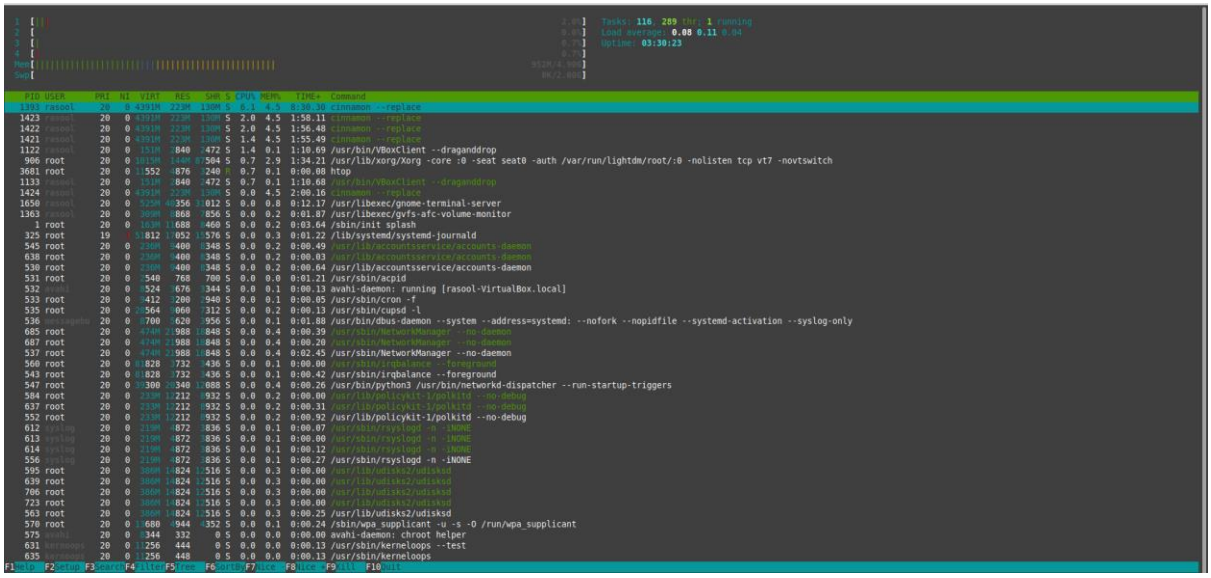
To install htop on Ubuntu or Debian-based systems, use the following command:

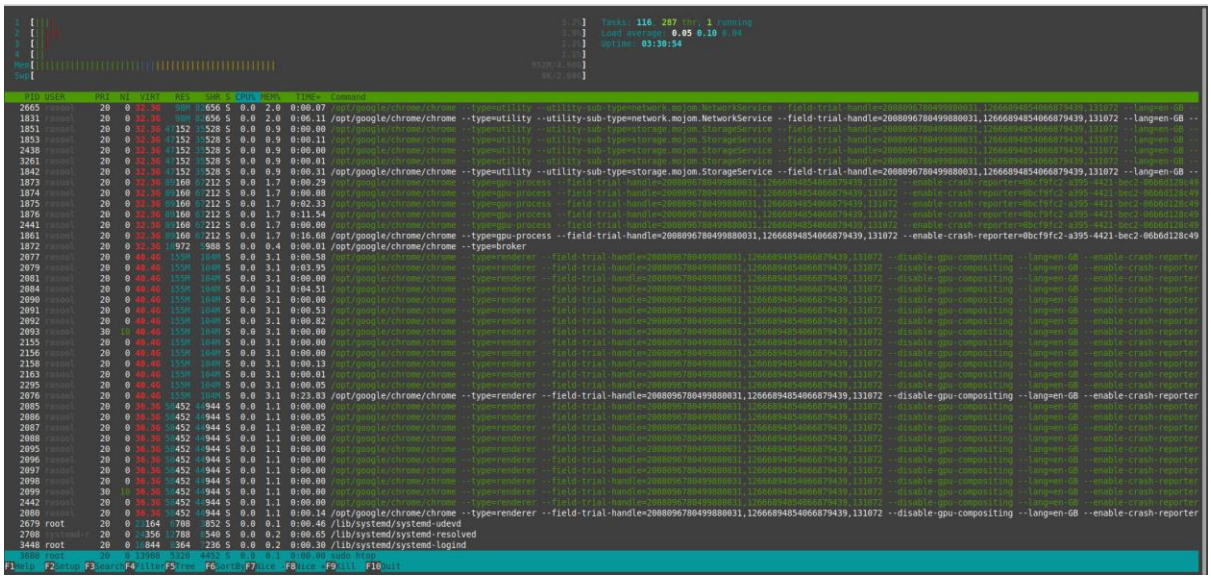
sudo apt-get install htop

Once installed, we can use the following command to display the system details:

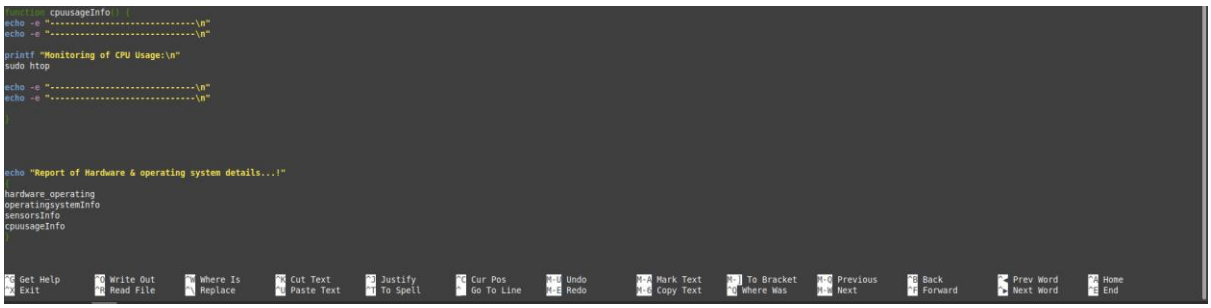
htop

We can use these open-source software tools to display an overview of hardware and operating system details and monitor the temperature and current usage of various hardware components in Linux.

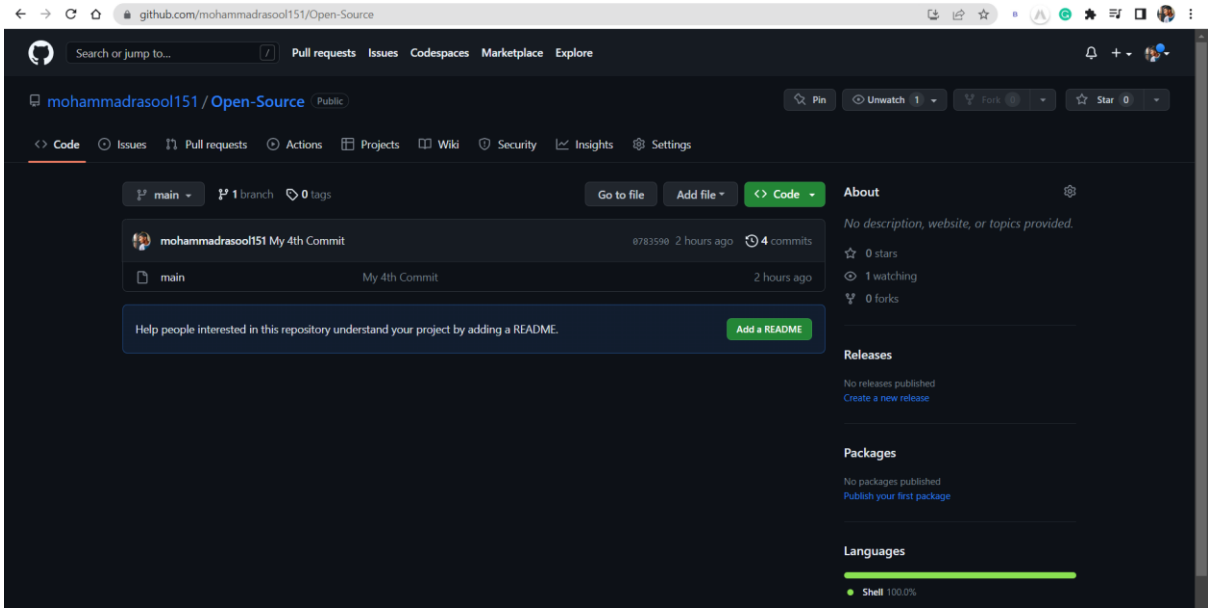


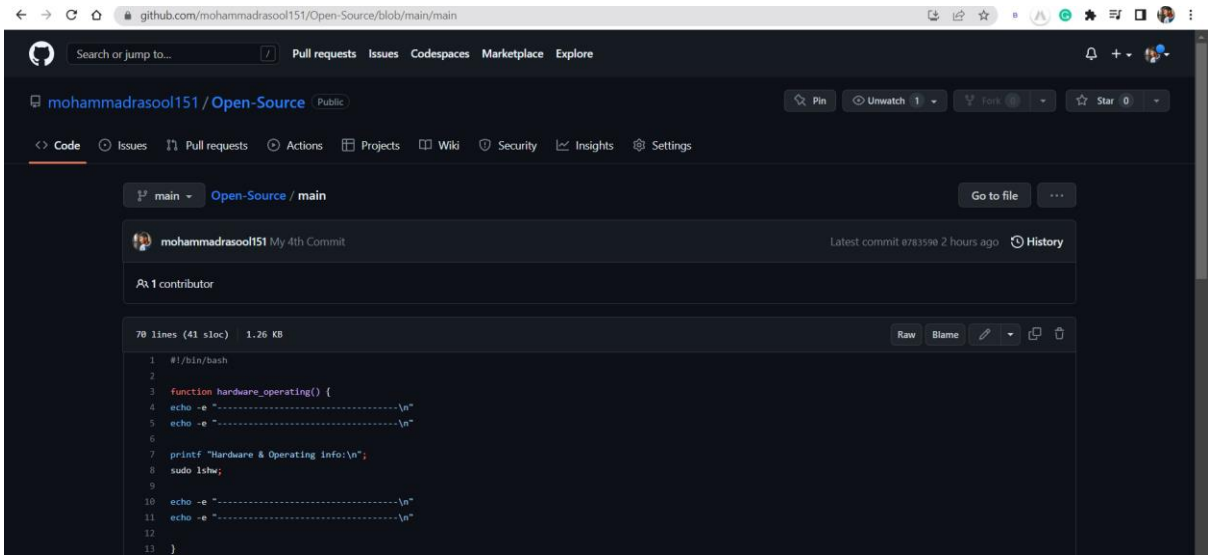


Created a File called “main” and using functions I had executed all 4 commands for getting info of Hardware overview & operating system details, and live monitoring of temperature and current working of various hardware components



GITHUB:


github.com/mohammadrasool151/Open-Source
Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
mohammadrasool151 / Open-Source
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
mohammadrasool151 My 4th Commit
0783590 2 hours ago
4 commits
main
My 4th Commit
2 hours ago
Help people interested in this repository understand your project by adding a README.
Add a README
About
No description, website, or topics provided.
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages
Shell 100.0%


github.com/mohammadrasool151/Open-Source/blob/main/main
Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
mohammadrasool151 / Open-Source
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
Open-Source / main
Go to file
mohammadrasool151 My 4th Commit
Latest commit 0783590 2 hours ago
History
1 contributor
70 lines (41 sloc)
1.26 KB
Raw
Blame
#!/bin/bash
function hardware_operating() {
echo -e "-----------------------------------\n"
echo -e "-----------------------------------\n"
printf "Hardware & Operating info:\n";
sudo lshw;
echo -e "-----------------------------------\n"
echo -e "-----------------------------------\n"
}

```
function operatingsystemInfo() {
echo -e "------------------------------\n"
echo -e "------------------------------\n"

printf "Operating System Info in graphical way:\n";
sudo neofetch
echo -e "------------------------------\n"
echo -e "------------------------------\n"

}

function sensorsInfo() {
echo -e "------------------------------\n"
echo -e "------------------------------\n"

printf "Temperature and current usage of Hardware systems:\n";
sudo sensors-detect;
sudo sensors;

echo -e "------------------------------\n"
echo -e "------------------------------\n"

}

function cpuusageInfo() {
echo -e "------------------------------\n"
echo -e "------------------------------\n"

printf "Monitoring of CPU Usage:\n"
sudo htop

echo -e "------------------------------\n"
echo -e "------------------------------\n"

}
```

```
echo "Report of all Hardware & operating system details, sensors(temparature), cpu usage...!"
{
hardware_operating
operatingsystemInfo
sensorsInfo
cpuusageInfo
}
```

GIT CODE:

```
#!/bin/bash

function hardware_operating() {
echo -e "---------------------------------\n"
echo -e "---------------------------------\n"

printf "Hardware & Operating info:\n";
sudo lshw;

echo -e "---------------------------------\n"
echo -e "---------------------------------\n"

}


function operatingsystemInfo() {
echo -e "----------------------------\n"
echo -e "----------------------------\n"

printf "Operating System Info in graphical way:\n";
sudo neofetch
echo -e "----------------------------\n"
echo -e "----------------------------\n"

}
```

```
function sensorsInfo() {
echo -e "-----------------------------\n"
echo -e "-----------------------------\n"

printf "Temperature and current usage of Hardware systems:\n";
sudo sensors-detect;
sudo sensors;

echo -e "-----------------------------\n"
echo -e "-----------------------------\n"

}

function cpuusageInfo() {
echo -e "-----------------------------\n"
echo -e "-----------------------------\n"

printf "Monitoring of CPU Usage:\n"
sudo htop

echo -e "-----------------------------\n"
echo -e "-----------------------------\n"

}
```

```
echo "Report of all Hardware & operating system details,
sensors(temparature), cpu usage...!"
{
hardware_operating
operatingsystemInfo
sensorsInfo
cpuusageInfo
}
```

-Thank You