

DIPLOMADO INTERNET DE LAS COSAS IOT

CAPÍTULO 3

ELABORADO POR:

GLEIDYS GISELA SALAS ZUÑIGA
JOSÉ DANIEL ESTRADA PULGARIN
JUAN DIEGO VELÁSQUEZ LOAIZA

DOCENTE:

ALBERT MONTOYA

INSTITUTO TECNOLÓGICO METROPOLITANO

MEDELLÍN

2021

INTRODUCCIÓN

Linux es un sistema operativo como MacOS, Windows es multitarea y multiusuario, libre y gratuito, este es necesario para que los computadores permitan utilizar programas y funcione de forma correcta.

Linux tiene un entorno gráfico pero la manera más habitual y potente de utilizar es por medio de comandos que son unas líneas de órdenes que se ejecutan en el Terminal con el fin de realizar tareas, consultar, editar, permitir permisos a usuarios sobre algunas carpetas del sistema operativo; es por esto por lo que en este informe se explicaran los comandos básicos de Linux ilustrados con imágenes directamente ejecutados en Terminal.

OBJETIVOS

OBJETIVO GENERAL

Utilizar la máquina virtual de virtual box y el sistema operativo ubuntu-linux, para realizar diferentes actividades con algunos comandos y así obtener resultados que nos permiten evidenciar una implementación correcta y adecuada.

OBJETIVOS ESPECÍFICOS

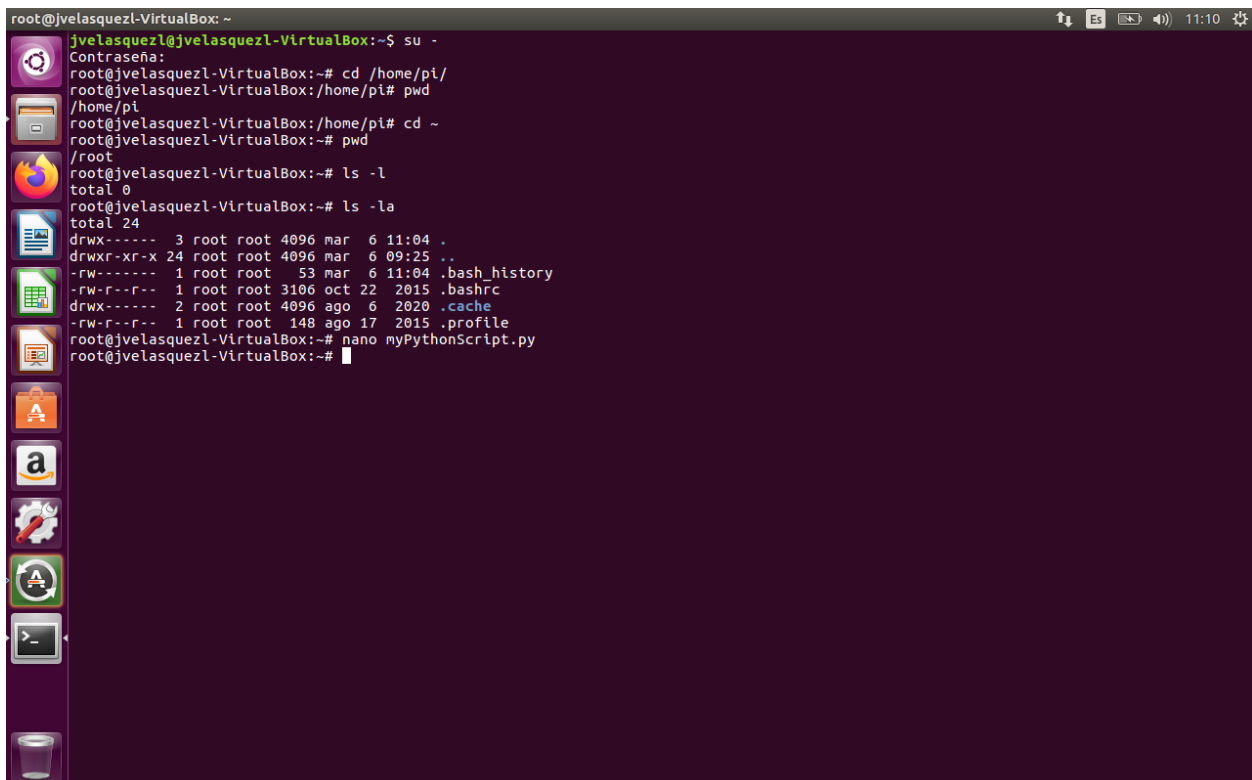
1. Aprender a dar permisos a los usuarios creados diferentes del usuario root que es el administrador.
2. Utilizar los comandos básicos para navegar, hacer copias, cambiar rutas de archivos, crear carpetas entre otros.
3. Crear y editar archivos que nos permite la integración entre Ubuntu-linux y phyton.

1. Descubriendo comandos básicos:

En esta sección aprendemos como navegar entre directorios utilizando los comandos **pwd**, **cd** y la instrucción para retornar al directorio principal con **cd~**

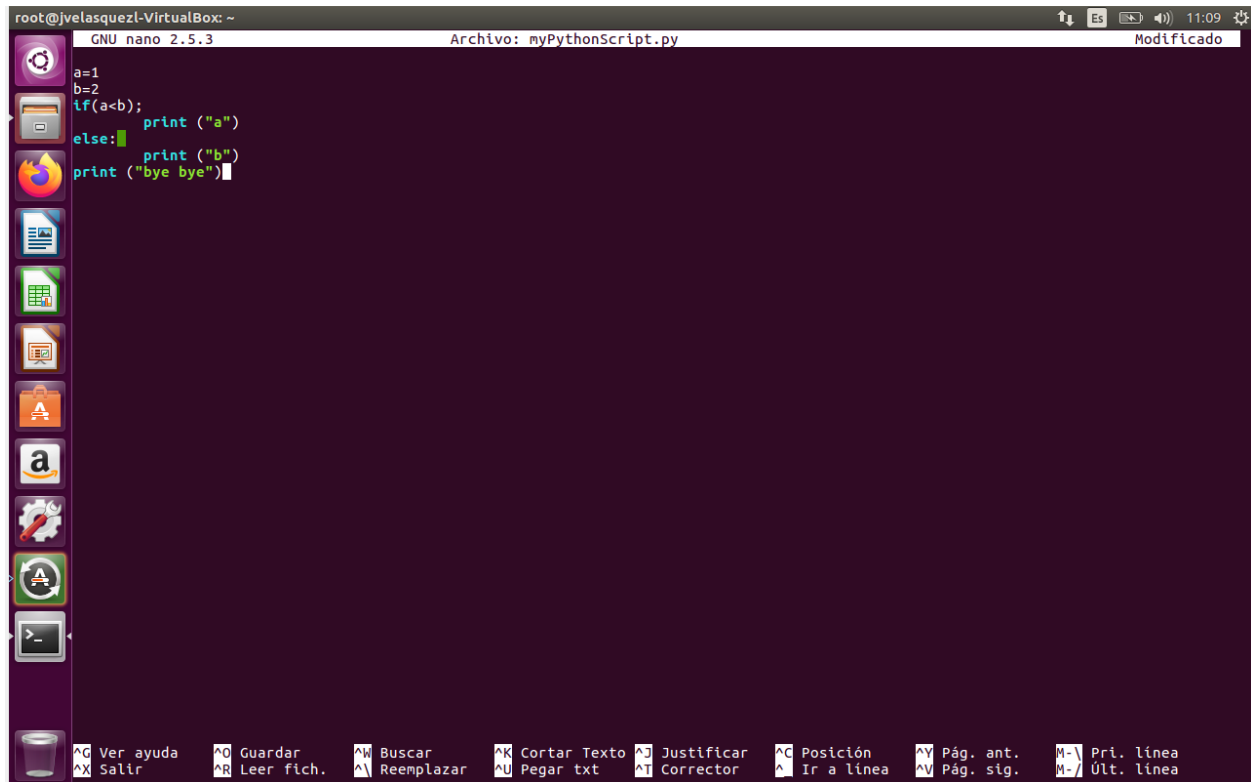
Además, con los comandos **ls** y **–l** podemos enumerar y ver de forma detallada el contenido del directorio en el que nos encontremos incluso utilizando **–la** podemos ver los archivos ocultos.

Mediante la instrucción **nano** nos abre un editor con el que podemos crear nuevos archivos



```
root@jvelasquezl-VirtualBox: ~
jvelasquezl@jvelasquezl-VirtualBox:~$ su -
Contraseña:
root@jvelasquezl-VirtualBox:~# cd /home/pi/
root@jvelasquezl-VirtualBox:/home/pi# pwd
/home/pi
root@jvelasquezl-VirtualBox:/home/pi# cd ~
root@jvelasquezl-VirtualBox:~# pwd
/root
root@jvelasquezl-VirtualBox:~# ls -l
total 0
root@jvelasquezl-VirtualBox:~# ls -la
total 24
drwx----- 3 root root 4096 mar  6 11:04 .
drwxr-xr-x 24 root root 4096 mar  6 09:25 ..
-rw----- 1 root root   53 mar  6 11:04 .bash_history
-rw-r--r-- 1 root root 3106 oct 22  2015 .bashrc
drwx----- 2 root root 4096 ago  6  2020 .cache
-rw-r--r-- 1 root root  148 ago 17  2015 .profile
root@jvelasquezl-VirtualBox:~# nano myPythonScript.py
root@jvelasquezl-VirtualBox:~#
```

Luego utilizar **nano** nos abre el editor para escribir las líneas de nuestro archivo Python



The screenshot shows a terminal window titled "root@jvelasquezl-VirtualBox: ~". The terminal is running the GNU nano 2.5.3 text editor, editing a file named "myPythonScript.py". The editor's status bar at the top right indicates "Modificado". The code being edited is a simple Python script:

```
a=1
b=2
if(a<b); print ("a")
else: print ("b")
print ("bye bye")
```

The terminal window has a sidebar on the left with various application icons. At the bottom, there is a menu bar with the following options:

Ver ayuda	Guardar	Buscar	Cortar Texto	Justificar	Posición	Pág. ant.	Pri. línea
Salir	Leer fich.	Reemplazar	Pegar txt	Corrector	Ir a línea	Pág. sig.	Últ. línea

Mediante el comando **ls** verificamos la correcta creación del archivo en el directorio.

El comando **cat** nos permite ver el contenido del archivo creado previamente

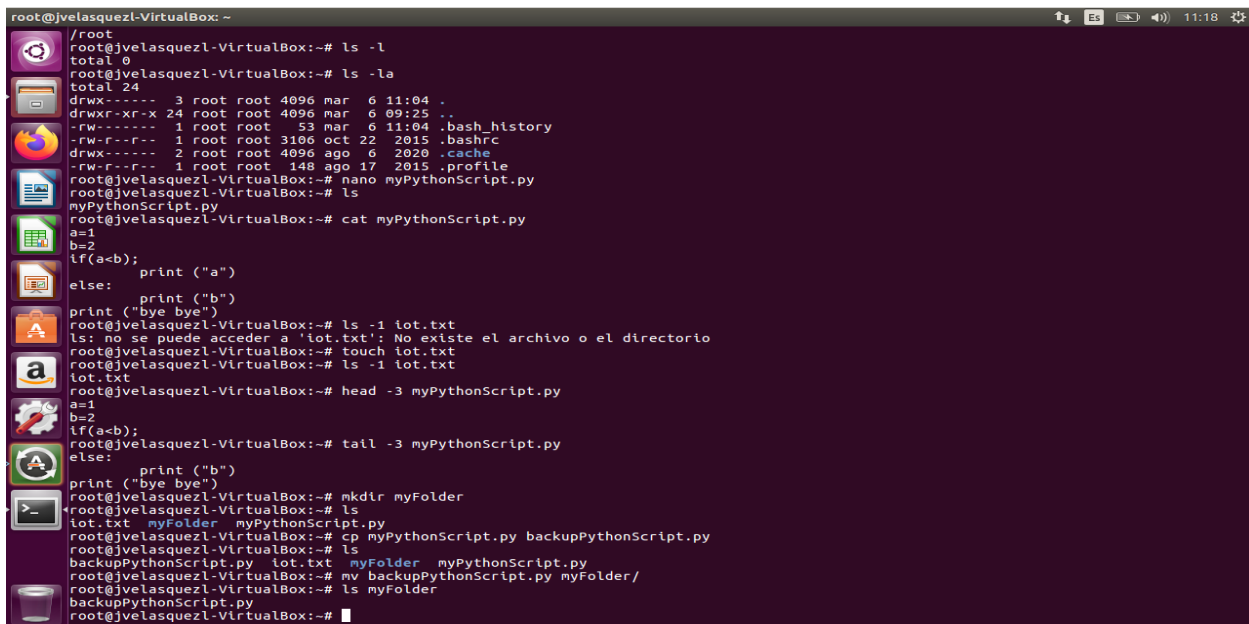
El commando **touch** nos permite crear directamente un archivo y usando **head** o **tail** podemos ver X líneas de la cabecera o por su defecto del pie del archivo.

```
root@jvelasquezl-VirtualBox: ~
jvelasquezl@jvelasquezl-VirtualBox:~$ su -
Contraseña:
root@jvelasquezl-VirtualBox:~# cd /home/pi/
root@jvelasquezl-VirtualBox:/home/pi# pwd
/home/pi
root@jvelasquezl-VirtualBox:/home/pi# cd ~
root@jvelasquezl-VirtualBox:~# pwd
/root
root@jvelasquezl-VirtualBox:~# ls -l
total 0
root@jvelasquezl-VirtualBox:~# ls -la
total 24
drwx----- 3 root root 4096 mar  6 11:04 .
drwxr-xr-x 24 root root 4096 mar  6 09:25 ../
-rw----- 1 root root  53 mar  6 11:04 .bash_history
-rw-r--r-- 1 root root 3106 oct 22  2015 .bashrc
drwx----- 2 root root 4096 ago  6  2020 .cache
-rw-r--r-- 1 root root  148 ago 17  2015 .profile
root@jvelasquezl-VirtualBox:~# nano myPythonScript.py
root@jvelasquezl-VirtualBox:~# ls
myPythonScript.py
root@jvelasquezl-VirtualBox:~# cat myPythonScript.py
a=1
b=2
if(a<b);
    print ("a")
else:
    print ("b")
print ("bye bye")
root@jvelasquezl-VirtualBox:~# ls -l iot.txt
ls: no se puede acceder a 'iot.txt': No existe el archivo o el directorio
root@jvelasquezl-VirtualBox:~# touch iot.txt
root@jvelasquezl-VirtualBox:~# ls -l iot.txt
iot.txt
root@jvelasquezl-VirtualBox:~# head -3 myPythonScript.py
a=1
b=2
if(a<b);
root@jvelasquezl-VirtualBox:~# tail -3 myPythonScript.py
else:
    print ("b")
print ("bye bye")
root@jvelasquezl-VirtualBox:~#
```

Utilizando el comando **mkdir** nos va a permitir crear un nuevo directorio, en este caso llamado myFolder

Con el comando **cp** realizamos un respaldo del archivo backupPythonScript.py, el respaldo lo llamaremos backupPythonScript.py

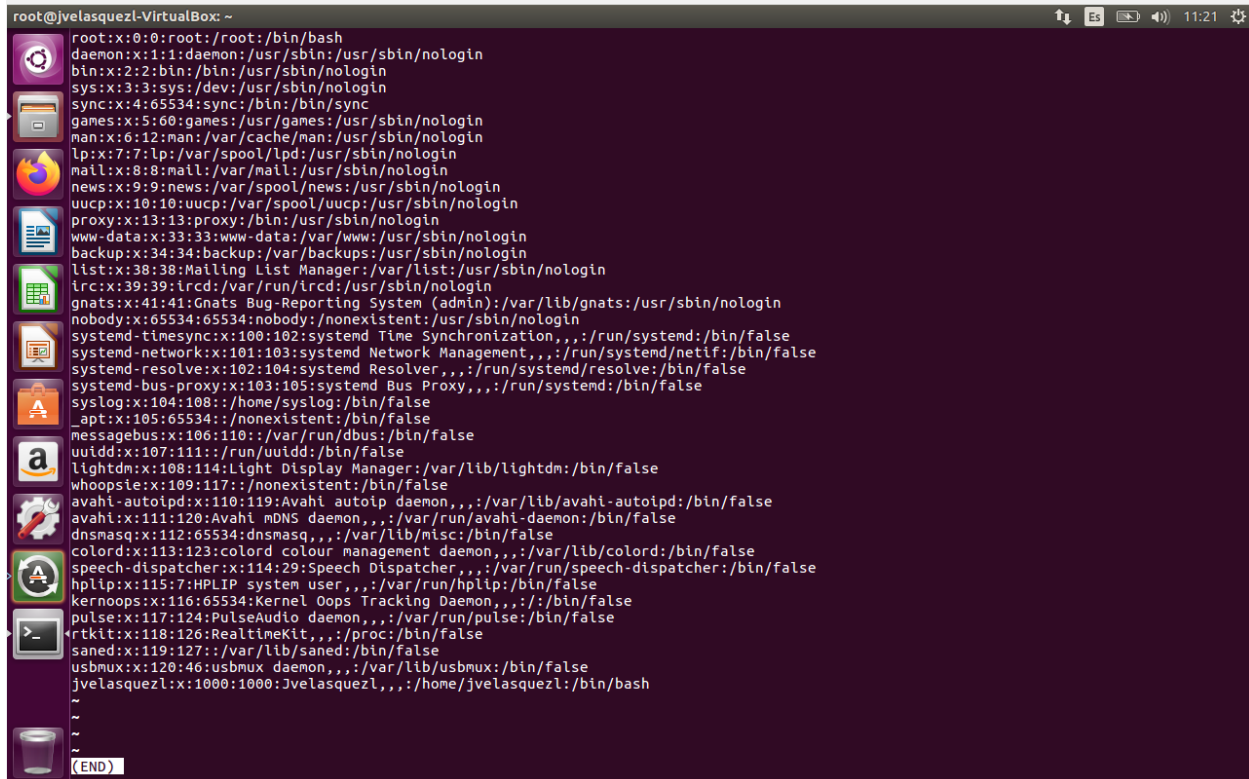
Usando el comando **mv** podremos mover el archivo backupPythonScript.py al directorio previamente creado myFolder



```
root@jvelasquezl-VirtualBox: ~  
/root  
root@jvelasquezl-VirtualBox:~# ls -l  
total 0  
root@jvelasquezl-VirtualBox:~# ls -la  
total 24  
drwxr-xr-x  3 root root 4096 mar  6 11:04 .  
drwxr-xr-x 24 root root 4096 mar  6 09:25 ..  
-rw-r--r--  1 root root   53 mar  6 11:04 .bash_history  
-rw-r--r--  1 root root 3106 oct 22  2015 .bashrc  
drwxr-xr-x  2 root root 4096 ago  6  2020 .cache  
-rw-r--r--  1 root root 148 ago 17  2015 .profile  
root@jvelasquezl-VirtualBox:~# nano myPythonScript.py  
root@jvelasquezl-VirtualBox:~# ls  
myPythonScript.py  
root@jvelasquezl-VirtualBox:~# cat myPythonScript.py  
a=1  
b=2  
if(a<b);  
    print ("a")  
else:  
    print ("b")  
print ("bye bye")  
root@jvelasquezl-VirtualBox:~# ls -l iot.txt  
ls: no se puede acceder a 'iot.txt': No existe el archivo o el directorio  
root@jvelasquezl-VirtualBox:~# touch iot.txt  
root@jvelasquezl-VirtualBox:~# ls -l iot.txt  
iot.txt  
root@jvelasquezl-VirtualBox:~# head -3 myPythonScript.py  
a=1  
b=2  
if(a<b);  
root@jvelasquezl-VirtualBox:~# tail -3 myPythonScript.py  
else:  
    print ("b")  
print ("bye bye")  
root@jvelasquezl-VirtualBox:~# mkdir myFolder  
root@jvelasquezl-VirtualBox:~# ls  
iot.txt  myFolder  myPythonScript.py  
root@jvelasquezl-VirtualBox:~# cp myPythonScript.py backupPythonScript.py  
root@jvelasquezl-VirtualBox:~# ls  
backupPythonScript.py  iot.txt  myFolder  myPythonScript.py  
root@jvelasquezl-VirtualBox:~# mv backupPythonScript.py myFolder/  
root@jvelasquezl-VirtualBox:~# ls myFolder  
backupPythonScript.py  
root@jvelasquezl-VirtualBox:~#
```


2. Obtención de ayuda:

Mediante el comando **less /etc/passwd** podemos ver el archivo de contraseña del usuario. Para finalizar y cerrar el visor lo realizamos con el presionando “q”



```
root@jvelasquezl-VirtualBox: ~  
root:x:0:0:root:/root:/bin/bash  
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
bin:x:2:2:bin:/bin:/usr/sbin/nologin  
sys:x:3:3:sys:/dev:/usr/sbin/nologin  
sync:x:4:65534:sync:/bin:/bin/sync  
games:x:5:60:games:/usr/games:/usr/sbin/nologin  
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin  
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin  
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin  
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin  
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin  
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin  
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin  
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin  
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin  
ircd:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin  
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin  
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin  
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false  
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false  
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false  
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false  
syslog:x:104:108:/:home/syslog:/bin/false  
apt:x:105:65534:/:nonexistent:/bin/false  
messagebus:x:106:110:/:var/run/dbus:/bin/false  
uidd:x:107:111:/:run/uidd:/bin/false  
lightdm:x:108:114:Light Display Manager:/var/lib/lightdm:/bin/false  
whoopsie:x:109:117:/:nonexistent:/bin/false  
avahi-autoipd:x:110:119:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/bin/false  
avahi:x:111:120:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false  
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/bin/false  
colord:x:113:123:colord colour management daemon,,,:/var/lib/colord:/bin/false  
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/false  
hplip:x:115:7:HPLIP system user,,,:/var/run/hplip:/bin/false  
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/bin/false  
pulse:x:117:124:PulseAudio daemon,,,:/var/run/pulse:/bin/false  
rtkit:x:118:126:RealtimeKit,,,:/proc:/bin/false  
saned:x:119:127:/:var/lib/saned:/bin/false  
usbmux:x:120:46:usbmux daemon,,,:/var/lib/usbmux:/bin/false  
jvelasquezl:x:1000:1000:jvelasquezl,,,:/home/jvelasquezl:/bin/bash  
~  
~  
~  
(END)
```

3. Monitoreo de los procesos del sistema:

Usamos el comando **ps** para visualizar la información de los procesos básicos del usuario actual.

Digitando el comando **ps aux** se mostrará información sobre todos los procesos en ejecución.

```
root@jvelasquezl-VirtualBox: ~
root@jvelasquezl-VirtualBox:~# clear
root@jvelasquezl-VirtualBox:~# ps
  PID TTY          TIME CMD
 2996 pts/4    00:00:00 su
 2997 pts/4    00:00:00 bash
 3148 pts/4    00:00:00 ps
root@jvelasquezl-VirtualBox:~# ps aux
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1   0.0  0.4 185080 4560 ?        Ss   10:02   0:01 /sbin/init splash
root           2   0.0  0.0      0     0 ?        S    10:02   0:00 [kthreadd]
root           4   0.0  0.0      0     0 ?        I<   10:02   0:00 [kworker/0:0H]
root           6   0.0  0.0      0     0 ?        I<   10:02   0:00 [mm_percpu_wq]
root           7   0.0  0.0      0     0 ?        S    10:02   0:00 [ksoftirqd/0]
root           8   0.0  0.0      0     0 ?        I    10:02   0:00 [rcu_sched]
root           9   0.0  0.0      0     0 ?        I    10:02   0:00 [rcu_bh]
root          10   0.0  0.0      0     0 ?        S    10:02   0:00 [migration/0]
root          11   0.0  0.0      0     0 ?        S    10:02   0:00 [watchdog/0]
root          12   0.0  0.0      0     0 ?        S    10:02   0:00 [cpuhp/0]
root          13   0.0  0.0      0     0 ?        S    10:02   0:00 [kdevtmpfs]
root          14   0.0  0.0      0     0 ?        I<   10:02   0:00 [netns]
root          15   0.0  0.0      0     0 ?        S    10:02   0:00 [rcu_tasks_kthre]
root          16   0.0  0.0      0     0 ?        S    10:02   0:00 [kauditd]
root          17   0.0  0.0      0     0 ?        S    10:02   0:00 [khungtaskd]
root          18   0.0  0.0      0     0 ?        S    10:02   0:00 [oom_reaper]
root          19   0.0  0.0      0     0 ?        I<   10:02   0:00 [writeback]
root          20   0.0  0.0      0     0 ?        S    10:02   0:00 [kcompactd0]
root          21   0.0  0.0      0     0 ?        SN   10:02   0:00 [ksmd]
root          22   0.0  0.0      0     0 ?        SN   10:02   0:00 [khugepaged]
root          23   0.0  0.0      0     0 ?        I<   10:02   0:00 [crypto]
root          24   0.0  0.0      0     0 ?        I<   10:02   0:00 [kintegrityd]
root          25   0.0  0.0      0     0 ?        I<   10:02   0:00 [kblockd]
root          26   0.0  0.0      0     0 ?        I<   10:02   0:00 [ata_sff]
root          27   0.0  0.0      0     0 ?        I<   10:02   0:00 [md]
root          28   0.0  0.0      0     0 ?        I<   10:02   0:00 [edac-poller]
root          29   0.0  0.0      0     0 ?        I<   10:02   0:00 [devfreq_wq]
root          30   0.0  0.0      0     0 ?        I<   10:02   0:00 [watchdogd]
root          34   0.0  0.0      0     0 ?        S    10:02   0:00 [kswapd0]
root          35   0.0  0.0      0     0 ?        I<   10:02   0:00 [kworker/u3:0]
root          78   0.0  0.0      0     0 ?        S    10:02   0:00 [ecryptfs-kthrea]
root          79   0.0  0.0      0     0 ?        I<   10:02   0:00 [kthrotld]
root          80   0.0  0.0      0     0 ?        S    10:02   0:00 [acpi_thermal_pm]
root          81   0.0  0.0      0     0 ?        I<   10:02   0:00 [scsi_eh_0]
root          82   0.0  0.0      0     0 ?        S    10:02   0:00 [scsi_eh_1]
```

Ahora si queremos ver los procesos que utilizan los recursos del sistema usamos el comando **top**

```
root@jvelasquezl-VirtualBox: ~
top - 11:23:40 up 1:21, 1 user, load average: 0.00, 0.02, 0.00
Tasks: 155 total, 1 ejecutar, 122 hibernar, 0 detener, 0 zombie
%CPU(s): 5.3 usuario, 1.0 sist, 0.0 adecuado, 93.7 inact, 0.0 en espera, 0.0 hardw int, 0.0 softw int, 0.0 robar tiempo
KiB Mem : 100840 total, 95052 free, 578260 used, 335528 buff/cache
KiB Swap: 998396 total, 779560 free, 218836 used, 264036 avail Mem

  PID USUARIO    PR  NI   VIRT   RES   SHR S %CPU %MEM    TIME+  ORDER
1874 jvelasq+    20   0 1297184 208048 45752 S  3.7 20.6 2:13.48 compiz
805 root        20   0 389764 55500 8336 S  1.7 5.5 0:40.17 Xorg
1975 jvelasq+    20   0 534104 14784 10688 S  0.7 1.5 0:00.15 update-notifier
2979 jvelasq+    20   0 676672 35520 27716 S  0.3 3.5 0:02.48 gnome-terminal
1 root        20   0 185080 4560 3336 S  0.0 0.5 0:01.05 systemd
2 root        20   0      0      0      0 S  0.0 0.0 0:00.00 kthreadd
4 root        0 -20      0      0      0 I  0.0 0.0 0:00.00 kworker/0:0H
6 root        20   0      0      0      0 S  0.0 0.0 0:00.00 mm_percpu_wq
7 root        20   0      0      0      0 S  0.0 0.0 0:00.16 ksoftirqd/0
8 root        20   0      0      0      0 I  0.0 0.0 0:00.47 rcu_sched
9 root        20   0      0      0      0 I  0.0 0.0 0:00.00 rcu_bh
10 root       rt    0      0      0      0 S  0.0 0.0 0:00.00 migration/0
11 root       20   0      0      0      0 S  0.0 0.0 0:00.01 watchdog/0
12 root       20   0      0      0      0 S  0.0 0.0 0:00.00 cpuhp/0
13 root       20   0      0      0      0 S  0.0 0.0 0:00.00 kdevtmpfs
14 root       0 -20      0      0      0 I  0.0 0.0 0:00.00 netns
15 root       20   0      0      0      0 S  0.0 0.0 0:00.00 rcu_tasks_kthre
16 root       20   0      0      0      0 S  0.0 0.0 0:00.00 kauditd
17 root       20   0      0      0      0 S  0.0 0.0 0:00.00 khungtaskd
18 root       20   0      0      0      0 S  0.0 0.0 0:00.00 oom_reaper
19 root       0 -20      0      0      0 I  0.0 0.0 0:00.00 writeback
20 root       20   0      0      0      0 S  0.0 0.0 0:00.00 kcompactd0
21 root       25   5      0      0      0 S  0.0 0.0 0:00.00 ksmd
22 root       39  19      0      0      0 S  0.0 0.0 0:00.00 khugepaged
23 root       0 -20      0      0      0 I  0.0 0.0 0:00.00 crypto
24 root       0 -20      0      0      0 I  0.0 0.0 0:00.00 kintegrityd
25 root       0 -20      0      0      0 I  0.0 0.0 0:00.00 kblockd
26 root       0 -20      0      0      0 I  0.0 0.0 0:00.00 ata_sff
27 root       0 -20      0      0      0 I  0.0 0.0 0:00.00 md
28 root       0 -20      0      0      0 I  0.0 0.0 0:00.00 edac-poller
29 root       0 -20      0      0      0 I  0.0 0.0 0:00.00 devfreq_wq
30 root       0 -20      0      0      0 I  0.0 0.0 0:00.00 watchdogd
34 root       20   0      0      0      0 S  0.0 0.0 0:00.94 kswapd0
35 root       0 -20      0      0      0 I  0.0 0.0 0:00.00 kworker/u3:0
36 root       20   0      0      0      0 S  0.0 0.0 0:00.00 encryptfs-kthrea
78 root       0 -20      0      0      0 I  0.0 0.0 0:00.00 kthrotld
79 root       0 -20      0      0      0 I  0.0 0.0 0:00.00 acpi_thermal_pm
80 root       20   0      0      0      0 S  0.0 0.0 0:00.01 scsi_eh_0
```

4. Administración de procesos:

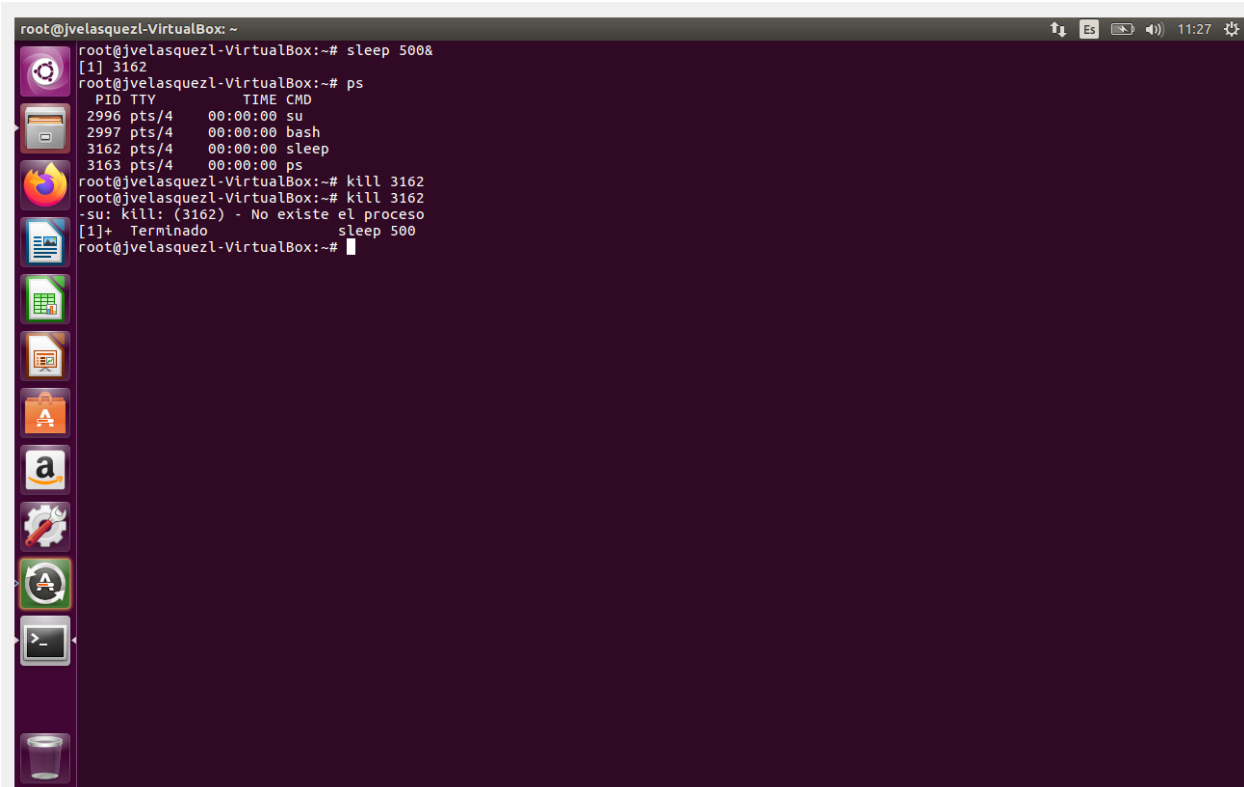
En este paso se crea un proceso que se ejecuta en segundo plano y el cual se mantendrá suspendido durante 500 segs. Al momento de ejecutarse podremos determinar el ID del proceso, el cual usaremos para la terminación de este.

Explicación de comandos:

sleep 500 & Ejecutando este comando iniciamos el proceso en segundo plano.

ps Nos permite determinar el ID del proceso. Para nuestro ejemplo es 3162

kill 3162 Con este comando podemos finalizar el proceso.

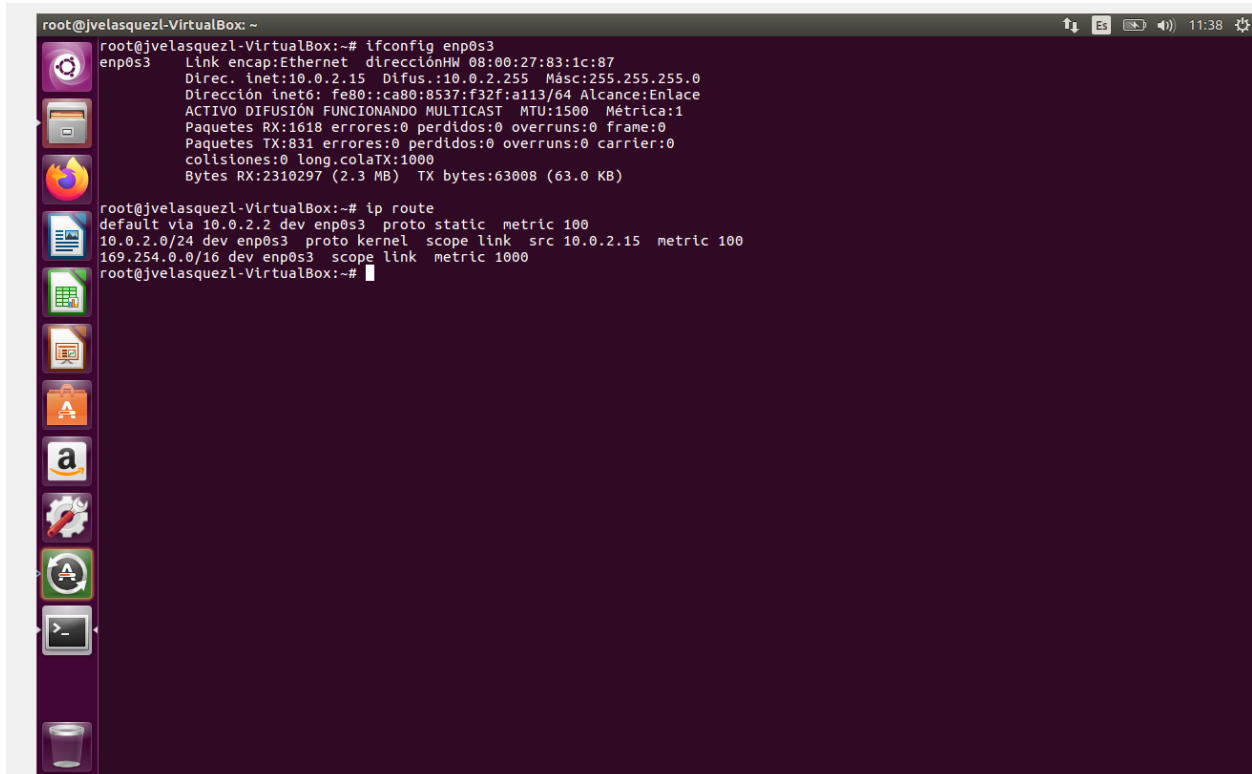


```
root@jvelasquezl-VirtualBox: ~  
root@jvelasquezl-VirtualBox:~# sleep 500&  
[1] 3162  
root@jvelasquezl-VirtualBox:~# ps  
  PID TTY          TIME CMD  
 2996 pts/4    00:00:00 su  
 2997 pts/4    00:00:00 bash  
 3162 pts/4    00:00:00 sleep  
 3163 pts/4    00:00:00 ps  
root@jvelasquezl-VirtualBox:~# kill 3162  
root@jvelasquezl-VirtualBox:~# kill 3162  
-su: kill: (3162) - No existe el proceso  
[1]+  Terminado                  sleep 500  
root@jvelasquezl-VirtualBox:~#
```

5. Determine la información de red básica:

Ejecutando el comando **ifconfig NOMBRE_ADAPTADOR** que para nuestro ejemplo es **enp0s3** podemos visualizar la información básica de la red.

Con **ip route** visualizamos la información de red adicional



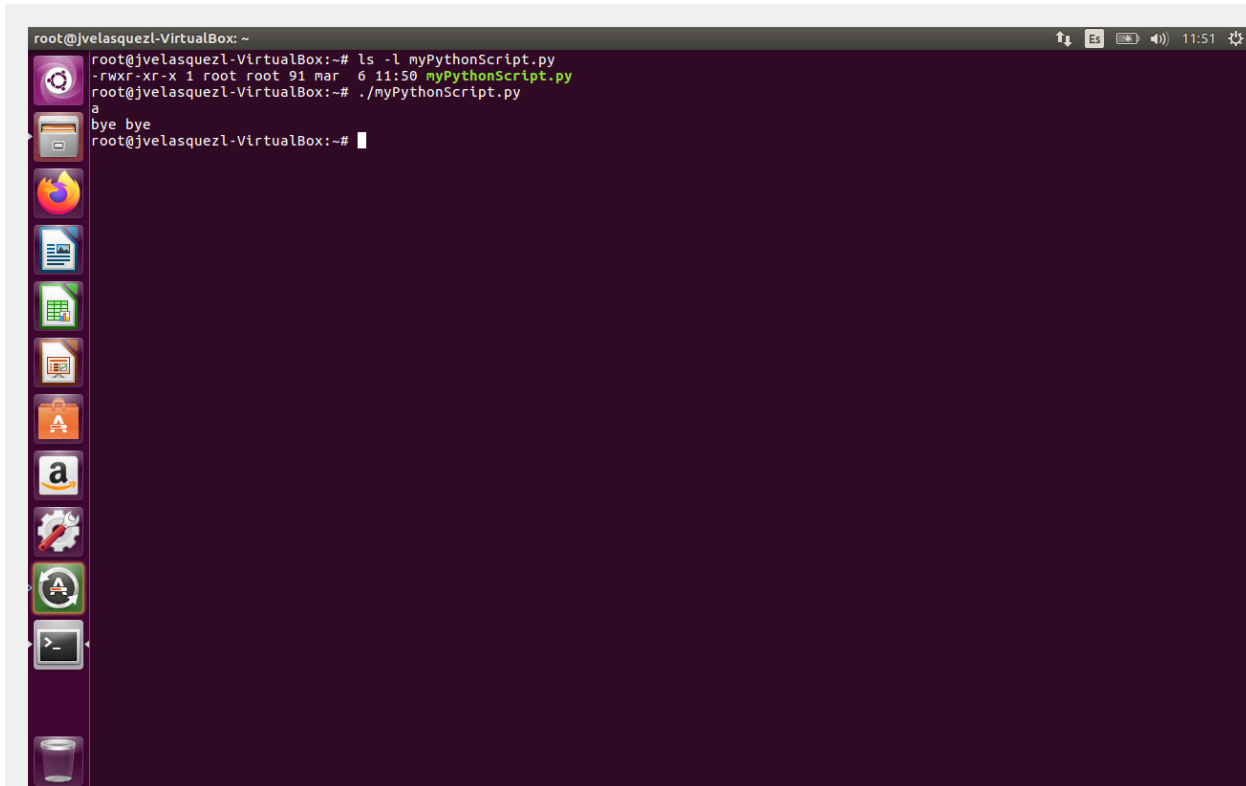
```
root@jvelasquezl-VirtualBox: ~  
root@jvelasquezl-VirtualBox:~# ifconfig enp0s3  
enp0s3      Link encap:Ethernet  direcciónHW 08:00:27:83:1c:87  
            Direc. inet:10.0.2.15  Difus.:10.0.2.255  Másc:255.255.255.0  
            Dirección inet6: fe80::ca80:8537:f32f:a113/64 Alcance:Enlace  
            ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST  MTU:1500  Métrica:1  
            Paquetes RX:1618 errores:0 perdidos:0 overruns:0 frame:0  
            Paquetes TX:831 errores:0 perdidos:0 overruns:0 carrier:0  
            colisiones:0 long.colaTX:1000  
            Bytes RX:2310297 (2.3 MB)  TX bytes:63008 (63.0 KB)  
  
root@jvelasquezl-VirtualBox:~# ip route  
default via 10.0.2.2 dev enp0s3  proto static  metric 100  
10.0.2.0/24 dev enp0s3  proto kernel  scope link  src 10.0.2.15  metric 100  
169.254.0.0/16 dev enp0s3  scope link  metric 1000  
root@jvelasquezl-VirtualBox:~#
```

6. Definir los permisos de archivos

En esta sección se consultan y definen permisos sobre el archivo Python generado ejecutando `ls -l myPythonScript.py`

Utilizando el comando **chmod** podremos agregar el indicador ejecutable en el archivo `myPythonScript.py`. Esto de la siguiente forma **chmod +x myPythonScript.py**

Con esto los permisos de nuestro archivo se modifican y permiten al grupo de usuario, grupos y otros usuarios poder ejecutar el archivo.



```
root@jvelasquezl-VirtualBox: ~  
root@jvelasquezl-VirtualBox:~# ls -l myPythonScript.py  
-rwxr-xr-x 1 root root 91 mar  6 11:50 myPythonScript.py  
root@jvelasquezl-VirtualBox:~# ./myPythonScript.py  
a  
bye bye  
root@jvelasquezl-VirtualBox:~#
```

Nota: los permisos de archivos se dividen en cuatro secciones diferentes. La primera sección es “-” al comienzo de la línea, que indica si es un archivo normal, un directorio o un archivo enlazado. “-” indica que es un archivo normal. Las tres secciones siguientes constan de tres caracteres específicos que indican los permisos. Las tres secciones se refieren al propietario, el grupo y otros usuarios. Este archivo presenta permisos de lectura y escritura para el propietario (rw-), permisos de lectura para el grupo (r--) y permisos de lectura para otros usuarios (r--).

7. Muestre la memoria, la información de uso del disco y el apagado del sistema

Utilice el comando **free** lo usamos para verificar el uso de la memoria.

Con el comando **free -h** podemos visualizar la memoria en valores de listado “legibles para el ser humano” con Mbytes en lugar de bytes únicamente.

Usando los comandos **df -h** podemos verificar la cantidad de espacio libre en el disco en los sistemas de archivos.

Finalmente, con el comando **sudo shutdown -h now** el sistema se apagará de forma inmediata, para la ejecución de este comando se debe hacer mediante privilegios altos asignados al usuario, por lo tanto, para este caso al ingresar **sudo** ejecutamos el comando como usuario root del sistema

```
root@jvelasquezl-VirtualBox: ~
root@jvelasquezl-VirtualBox:~# ls -l myPythonScript.py
-rwxr-xr-x 1 root root 91 mar  6 11:50 myPythonScript.py
root@jvelasquezl-VirtualBox:~# ./myPythonScript.py
a
bye bye
root@jvelasquezl-VirtualBox:~# free
              total        used        free      shared  buff/cache   available
Memoria:    1008840       575048       95428        2984       338364       267164
Swap:       998396       218324       780072
root@jvelasquezl-VirtualBox:~# free -h
              total        used        free      shared  buff/cache   available
Memoria:    985M         561M         93M        2,9M       330M       260M
Swap:       974M         213M         761M
root@jvelasquezl-VirtualBox:~# df
S.ficheros  bloques de 1K  Usados  Disponibles  Uso%  Montado en
udev         473888         0       473888      0% /dev
tmpfs        100884         3652       97232      4% /run
/dev/sda1    9204224  4538812  4174816     53% /
tmpfs        504420         252       504168      1% /dev/shm
tmpfs         5120           4         5116      1% /run/lock
tmpfs        504420         0       504420      0% /sys/fs/cgroup
tmpfs        100884         120       100764      1% /run/user/1000
root@jvelasquezl-VirtualBox:~# df -h
S.ficheros  Tamaño Usados  Disp  Uso%  Montado en
udev         463M      0      463M   0% /dev
tmpfs         99M     3,6M    95M    4% /run
/dev/sda1     8,8G   4,4G   4,0G   53% /
tmpfs         493M   252K   493M    1% /dev/shm
tmpfs         5,0M   4,0K   5,0M    1% /run/lock
tmpfs         493M      0   493M    0% /sys/fs/cgroup
tmpfs         99M    120K    99M    1% /run/user/1000
root@jvelasquezl-VirtualBox:~# sudo shutdown -h now
sudo: shutdown: orden no encontrada
root@jvelasquezl-VirtualBox:~# sudo shutdown -h now
```

CONCLUSIONES

Después de realizar la actividad es posible concluir que:

- A pesar de que Linux tiene un entorno gráfico y puede ser más amigable de utilizar, la forma de sacarle un mayor provecho a este sistema operativo es utilizando la ventana Terminal y ejecutar comandos de órdenes
- En Terminal se pueden ejecutar comandos básicos como crear un archivo txt, y comandos complejos como crear y ejecutar un programa en Python
- Linux nos permite consumir servicios, interactuar con lenguajes de programación.