

Actividad | # 2| Primeros

Comandos de Shell

Sistemas Operativos 1

Ingeniería en Desarrollo
de Software



TUTOR: Urbano Francisco Ortega Rivera

ALUMNO: Jenifer Ortiz Vazquez

FECHA: 27/12/2023

INDICE

1. Introducción.....	3
2. Descripción.....	4
3. Justificación.....	5
4. Desarrollo.....	6
5. Conclusión.....	12
6. Referencias.....	13

INTRODUCCION

En esta actividad estaremos conociendo algunos comandos que podemos utilizar en un Shell. Existen muchos comandos que podemos utilizar, cada uno con una función específica, sin embargo en nuestra siguiente actividad estaremos ejecutando solamente diez de ellos, conoceremos algunos muy básicos, pero no menos importantes ya que cada uno cumple con una función relevante para poder hacer un uso adecuado del sistema con el que estamos trabajando y para nuestro conocimiento.

Dichos comandos los estaremos ejecutando en nuestra terminal y anexando en este documento dando una breve descripción de la función que realizan y cuál es la forma correcta de utilizarlo para que nos de algún resultado o la función deseada. También podremos observar que, en ocasiones, dependiendo la función que busquemos, se pueden utilizar dos o más comandos en una misma función, para poder realizar nuestra operación requerida.

Todo esto lo podremos visualizar en algunas imágenes tomadas desde nuestra pantalla Shell para comprender de una mejor forma como se pueden utilizar en nuestro sistema operativo.

DESCRIPCION

Un Shell es un programa que toma comandos del teclado para dárselos a un sistema operativo, el cual los pueda ejecutar.

El Shell de Unix proporciona una interfaz que permite al usuario interactuar con el sistema operativo mediante la ejecución de comandos sin embargo un Shell también es un lenguaje de programación que puede realizar funciones como, construcciones para control de flujo, alternancia, bucles, condicionales, operaciones matemáticas básicas, funciones con nombre, entre otras

Los Shells se puede usar de forma interactiva (una terminal o un emulador de terminal); y de forma no interactiva (leyendo comandos desde un archivo).

El kernel representa el nivel más bajo del software, fácilmente reemplazable, que interactúa con el hardware de tu computadora. Es responsable de interconectar todas las aplicaciones que se ejecutan en el modo de usuario hasta el hardware físico, permite que los procesos obtengan información entre si mediante la comunicación entre procesos

Podemos encontrar tres tipos de kernel:

Micronúcleo, el cual se enfoca en administrar solo lo que tiene que hacer, tienen portabilidad, ocupan poco espacio

Monolítico, es lo opuesto a los microkernel ya que también incluye controladores de dispositivo, administración del sistema de archivos y llamadas al servidor del sistema. Tiende a ser mejor para acceder al hardware y realizar múltiples tareas

Híbrido, este es la combinación del micronúcleo y monolítico.

JUSTIFICACION

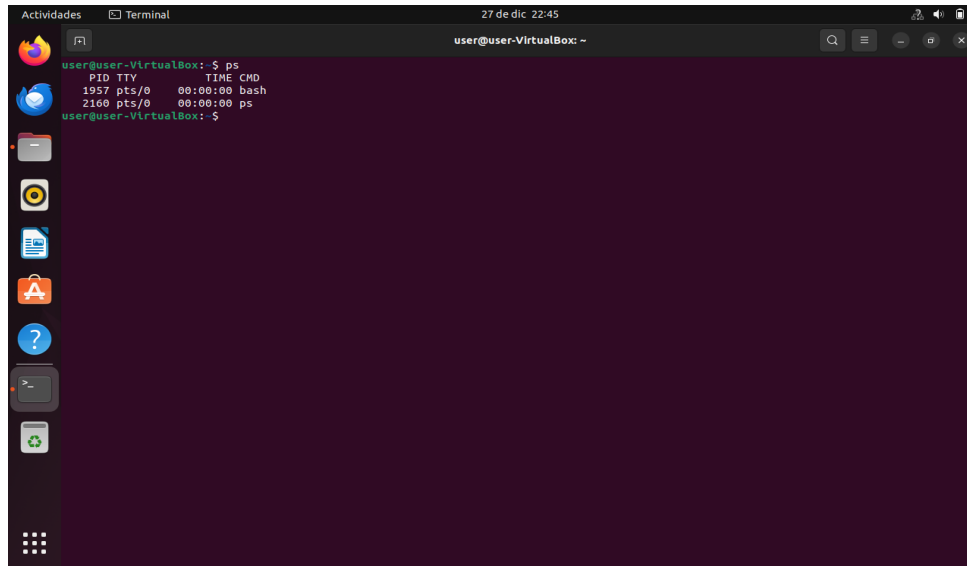
Es muy importante conocer los diferentes tipos de comandos que hay y como se aplican en nuestro Shell, ya que es una forma muy completa para poder interactuar con nuestro sistema operativo de una forma más fácil, ya que desde una misma pantalla podemos realizar múltiples procesos como lo son, creación de carpetas, descarga de archivos, modificación de documentos, etc.

Esto nos facilita la realización de muchas operaciones, pero para ello debemos de conocer con exactitud como utilizar todos esos comandos, por eso en esta actividad estaremos ejecutando algunos de ellos, ya que si desconocemos su función podríamos llegar a dañar algo, o si no son escritos correctamente no podremos realizar ningún movimiento.

Nosotros seremos los responsables de indicarle a nuestro dispositivo lo que tiene que realizar mediante los comandos que utilizemos, es por eso que es muy importante saber qué es lo que queremos hacer y cómo llevarlo a cabo, para realizar un buen trabajo utilizando todas las herramientas proporcionadas.

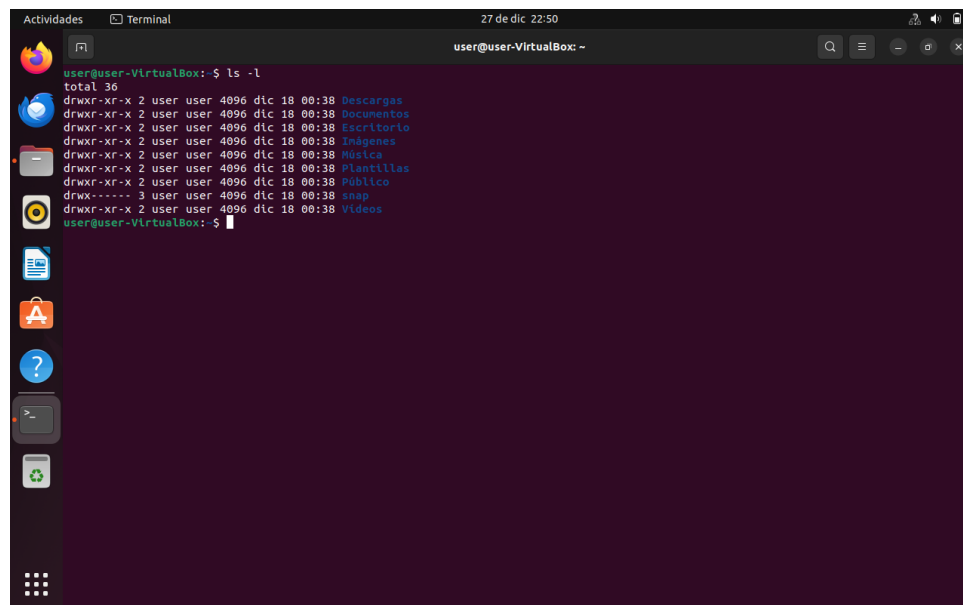
DESARROLLO

1.El comando *ps* (estado de procesos) nos muestra todos los procesos en ejecución en el sistema. Nos muestra el identificador único del proceso (PID), tipo de terminal (TTY), el tiempo de funcionamiento (TIME), y el comando que lanza el proceso (CMD).



```
user@user-VirtualBox: ~  
$ ps  
  PID TTY          TIME CMD  
 1957 pts/0    00:00:00 bash  
 2160 pts/0    00:00:00 ps  
user@user-VirtualBox: ~$
```

2.Con el comando *ls* podemos ver el contenido de un directorio, este comando tiene muchas opciones adicionales un ejemplo de ellas es al agregar *ls -l* donde nos enlistara de una forma larga y detallada los permisos del archivo, propietario, grupo, tamaño, y fecha hora de creación.

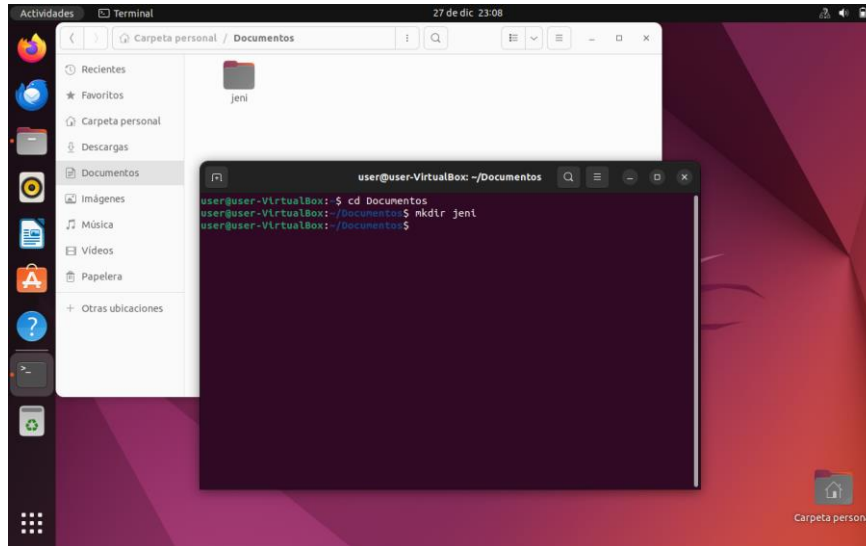


```
user@user-VirtualBox: ~$ ls -l  
total 36  
drwxr-xr-x 2 user user 4096 dic 18 00:38 Descargas  
drwxr-xr-x 2 user user 4096 dic 18 00:38 Documentos  
drwxr-xr-x 2 user user 4096 dic 18 00:38 Escritorio  
drwxr-xr-x 2 user user 4096 dic 18 00:38 Imágenes  
drwxr-xr-x 2 user user 4096 dic 18 00:38 Música  
drwxr-xr-x 2 user user 4096 dic 18 00:38 Plantillas  
drwxr-xr-x 2 user user 4096 dic 18 00:38 Público  
drwx----- 3 user user 4096 dic 18 00:38 snap  
drwxr-xr-x 2 user user 4096 dic 18 00:38 Videos  
user@user-VirtualBox: ~$
```

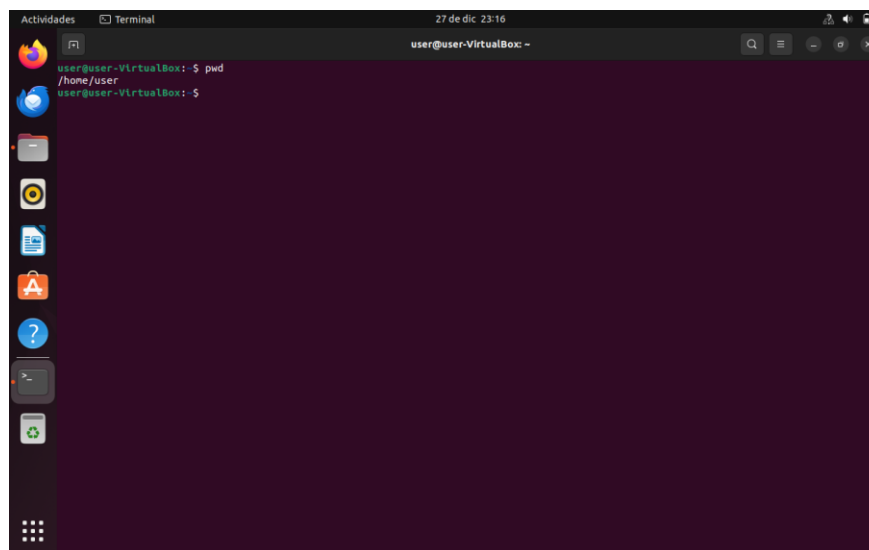
3. Para navegar por los archivos y directorios de Linux usamos *cd* donde se acompaña del nombre o ruta donde queremos ir.

4. El comando *mkdir* crea directorios uno o varios y establece permisos.

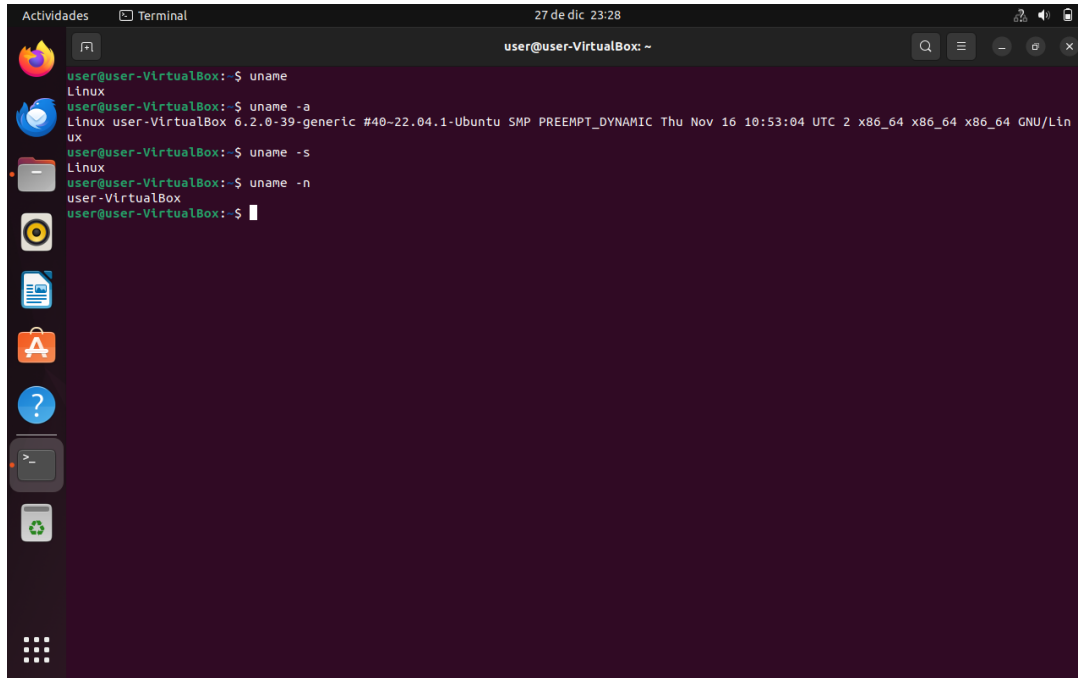
En nuestra pantalla podemos observar que una vez dentro de nuestra carpeta se creó otra utilizando los comandos anteriormente mencionados y de fondo se puede apreciar la validación de la misma.



5. Utilizamos este comando *pwd* para encontrar la ruta del directorio de trabajo actual

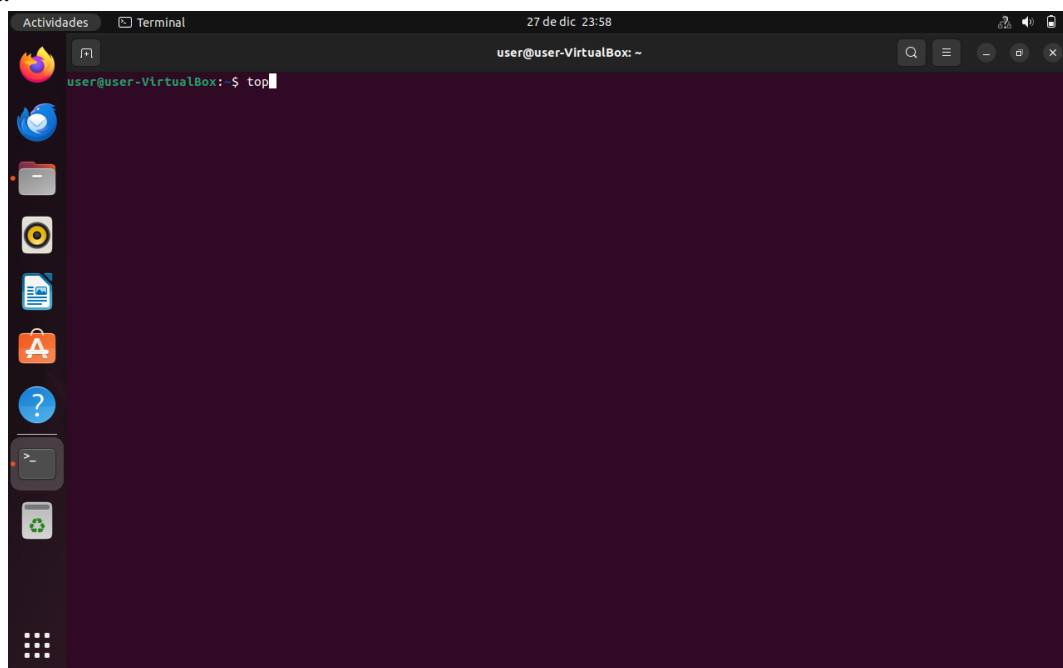


6.El comando *uname* imprime información detallada sobre el sistema Linux y el hardware. Se puede complementar con *-a* (imprime toda la información del sistema), *-s* (imprime el nombre del núcleo) y *-n* (imprime el nombre del host del nodo del sistema).



```
user@user-VirtualBox: ~  
user@user-VirtualBox:~$ uname  
Linux  
user@user-VirtualBox:~$ uname -a  
Linux user-VirtualBox 6.2.0-39-generic #40~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Thu Nov 16 10:53:04 UTC 2 x86_64 x86_64 x86_64 GNU/Linux  
user@user-VirtualBox:~$ uname -s  
Linux  
user@user-VirtualBox:~$ uname -n  
user-VirtualBox  
user@user-VirtualBox:~$
```

7.Comando *top* este nos mostrara todos los procesos en ejecución y una vista dinámica en tiempo real del sistema actual, nos resume la utilización de recursos desde la CPU hasta el uso de memoria.



```
user@user-VirtualBox: ~  
user@user-VirtualBox:~$ top
```


comando *top*

```

top - 23:58:13 up 1:24, 1 user, load average: 0.37, 0.13, 0.03
Tareas: 166 total, 2 ejecutar, 164 hibernar, 0 detener, 0 zombie
%Cpu(s): 1.7 us, 0.7 sy, 0.0 ni, 97.3 id, 0.3 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 3907.9 total, 2151.8 libre, 717.3 usado, 1038.9 búfer/caché
MiB Intercambio: 2810.0 total, 2810.0 libre, 0.0 usado, 2933.4 dispon Mem

  PID  USUARIO  PR  NI  VIRT  RES  SHR  S  %CPU  %MEM  HORA+ ORDEN
1443 user      20   0 3755208 347548 138276 S   1.0  8.7  1:58.95  gnome-shell
1622 user      20   0 318824 12196 7296 S   0.3  0.3  0:03.14  ibus-daemon
2505 user      20   0 567668 55340 42264 S   0.3  1.4  0:00.39  gnome-terminal-
2530 user      20   0 15824 4224 3456 R   0.3  0.1  0:00.01  top
1 root    20   0 166676 11732 8276 S   0.0  0.3  0:01.14  systemd
2 root    20   0 0 0 0 S   0.0  0.0  0:00.00  kthreadd
3 root    0 -20 0 0 0 I   0.0  0.0  0:00.00  rcu_gp
4 root    0 -20 0 0 0 I   0.0  0.0  0:00.00  rcu_par_gp
5 root    0 -20 0 0 0 I   0.0  0.0  0:00.00  slub_flushwq
6 root    0 -20 0 0 0 I   0.0  0.0  0:00.00  netns
8 root    0 -20 0 0 0 I   0.0  0.0  0:00.00  kworker/0:0H-kblockd
10 root   0 -20 0 0 0 I   0.0  0.0  0:00.00  mm_percpu_wq
11 root   20  0 0 0 0 I   0.0  0.0  0:00.00  rcu_tasks_kthread
12 root   20  0 0 0 0 I   0.0  0.0  0:00.00  rcu_tasks_rude_kthread
13 root   20  0 0 0 0 I   0.0  0.0  0:00.00  rcu_tasks_trace_kthread
14 root   20  0 0 0 0 S   0.0  0.0  0:00.24  ksoftirqd/0
15 root   20  0 0 0 0 R   0.0  0.0  0:01.26  rcu_preempt
16 root   rt  0 0 0 S   0.0  0.0  0:00.05  migration/0
17 root   -51  0 0 0 0 S   0.0  0.0  0:00.00  idle_inject/0
19 root   20  0 0 0 0 S   0.0  0.0  0:00.00  cpuhp/0
20 root   20  0 0 0 0 S   0.0  0.0  0:00.00  kdevtmpfs
21 root   0 -20 0 0 0 I   0.0  0.0  0:00.00  inet_frag_wq
22 root   20  0 0 0 0 S   0.0  0.0  0:00.00  kauditd
23 root   20  0 0 0 0 S   0.0  0.0  0:00.00  khungtaskd
25 root   20  0 0 0 0 S   0.0  0.0  0:00.00  oom_reaper
26 root   0 -20 0 0 0 I   0.0  0.0  0:00.00  writeback
28 root   20  0 0 0 0 S   0.0  0.0  0:00.51  kcompactd0
29 root   25  5 0 0 0 S   0.0  0.0  0:00.00  ksmd
30 root   39 19 0 0 0 S   0.0  0.0  0:00.00  khugepaged
31 root   0 -20 0 0 0 I   0.0  0.0  0:00.00  kintegrityd
32 root   0 -20 0 0 0 I   0.0  0.0  0:00.00  biodbg
33 root   0 -20 0 0 0 I   0.0  0.0  0:00.00  blkcg_punt_bio
34 root   0 -20 0 0 0 I   0.0  0.0  0:00.00  tpm_dev_wq
  
```

8.El comando *kill* es utilizado para terminar manualmente un programa que no responde, pero para ello debemos de conocer su número de identificación de proceso (PID), donde para conocer el PID se ejecuta el comando *ps*

```

user@user-VirtualBox: ~$ ps u
USER      PID  %CPU  %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
user      1293  0.0   0.1 165184  6272 tty2      Ssl+  22:34   0:00 /usr/libexec/gdm-wayland-session env GNOME_SHELL_SESSION_MODE=ubu
user      1305  0.0   0.3 225836 15872 tty2      Sl+   22:34   0:00 /usr/libexec/gnome-session-binary --session=ubuntu
user      2295  0.0   0.1 13744  5120 pts/0    Ss   23:09   0:00 bash
user      2400  0.5   0.1 15824  4224 pts/0    S+   23:29   0:02 top
user      2414  0.0   0.1 13616  5120 pts/1    Ss   23:32   0:00 bash
user      2436  0.0   0.0 15364  3456 pts/1    R+   23:37   0:00 ps u

user@user-VirtualBox: ~$ kill -9 2400
  
```

```
user@user-VirtualBox: ~  
$ top  
top - 27 de dic 23:59  
PID USUARIO PR NI VIRT RES SHR S %CPU %MEM HORA+ ORDEN  
1443 user 20 0 3755208 347548 138276 S 1.3 8.7 2:00.65 gnome-shell  
2530 user 20 0 18824 4224 3456 R 0.3 0.1 0:00.10 top  
1 root 20 0 166676 11732 8276 S 0.0 0.3 0:01.15 systemd  
2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd  
3 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_gp  
4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_par_gp  
5 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 slub_flushwq  
6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 netns  
8 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0H-kblockd  
10 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 mm_percpu_wq  
11 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_kthread  
12 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_rude_kthread  
13 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_trace_kthread  
14 root 20 0 0 0 0 S 0.0 0.0 0:00.25 ksoftirqd/0  
15 root 20 0 0 0 0 R 0.0 0.0 0:01.27 rcu_preempt  
16 root rt 0 0 0 0 S 0.0 0.0 0:00.05 migration/0  
17 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/0  
19 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/0  
20 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kdevtmpfs  
21 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 ln timer  
22 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kauditd  
23 root 20 0 0 0 0 S 0.0 0.0 0:00.00 khungtaskd  
25 root 20 0 0 0 0 S 0.0 0.0 0:00.00 oom reaper  
26 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 writeback  
28 root 20 0 0 0 0 S 0.0 0.0 0:00.51 kcompactd0  
29 root 25 5 0 0 0 S 0.0 0.0 0:00.00 ksm  
30 root 39 19 0 0 0 S 0.0 0.0 0:00.00 khugepaged  
31 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kintegrityd  
32 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kblockd  
33 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 blkcg_punt_bio  
34 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 tpm_dev_wq  
35 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 ata_sff  
36 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 nd  
Terminado (killed)  
user@user-VirtualBox: $
```

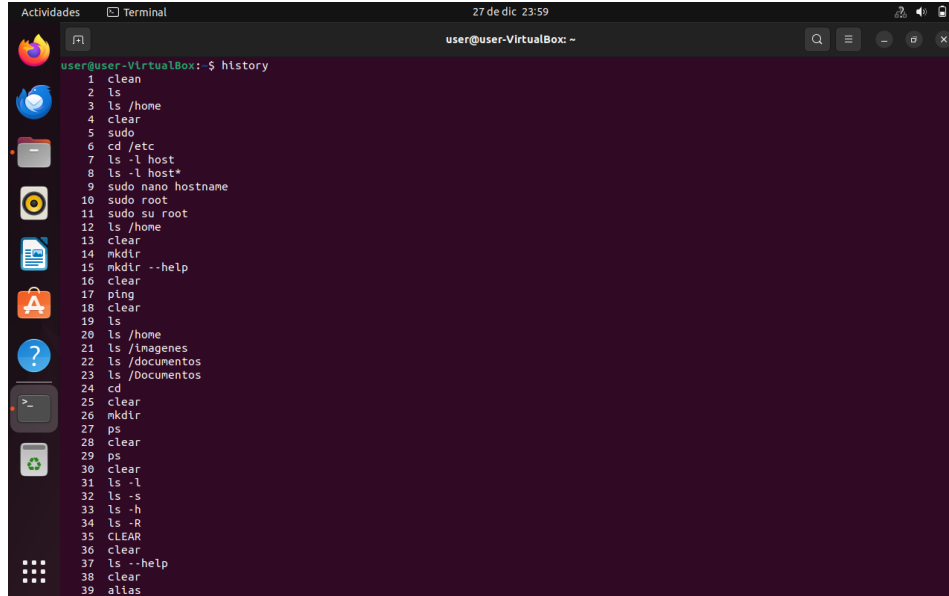
9. Otro comando utilizado es *man* que proporciona un manual de usuario de cualquier comando o utilidad que pueda ejecutar en terminal, incluyendo el nombre, la descripción y las opciones.

```
user@user-VirtualBox: ~  
$ man man  
Terminado (killed)  
user@user-VirtualBox: $
```



```
user@user-VirtualBox: ~  
MAN(1) Utilidades de paginador del manual MAN(1)  
NOMBRE man - interfaz de los manuales de referencia del sistema  
SINOPSIS man [opciones de man] [[sección] página ...] ...  
man -k [opciones de apropos] regexp ...  
man -K [opciones de man] [sección] term ...  
man -f [whatis opciones] página ...  
man -l [opciones de man] archivo ...  
man -W [-W [opciones de man] página ...]  
DESCRIPCIÓN man es el paginador de manuales del sistema. Cada argumento de página dado a man normalmente es el nombre de un  
programa, utilidad o función. La página de manual asociada con cada uno de estos argumentos es, pues, encontrada y  
mostrada. Si se proporciona una sección, man mirará solo en esa sección del manual. La acción predeterminada es buscar  
en todas las secciones disponibles siguiendo un orden predefinido (véase DEFAULTS), y mostrar solo la primera página  
encontrada, incluso si la página existe en varias secciones.  
  
La tabla de abajo muestra los números de sección del manual seguidos por los tipos de página que contienen.  
  
1 Programas ejecutables u órdenes de la shell  
2 Llamadas al sistema (funciones proporcionadas por el núcleo)  
3 Llamadas a biblioteca (funciones dentro de bibliotecas de programa)  
4 Archivos especiales (normalmente se encuentran en /dev)  
5 Formatos de archivo y convenios, p.e. /etc/passwd  
6 Juegos  
7 Miscelánea (incluidos paquetes de macros y convenios), p.e. man(7), groff(7), man-pages(7)  
8 Órdenes de administración del sistema (normalmente solo para root)  
9 Rutinas del núcleo [No estándar]  
  
Una página de manual contiene varias secciones.  
  
Nombres de sección convencionales: NOMBRE, SINOPSIS, CONFIGURACIÓN, DESCRIPCIÓN, OPCIONES, ESTADO DE SALIDA,  
VALOR DEVUELTO, ERRORES, ENTORNO, ARCHIVOS, VERSIONES, CONFORME A, NOTAS, DEFECTOS, EJEMPLO, AUTORES, y VÉASE TAMBIÉN.  
  
Los siguientes convenios se aplican a la sección SINOPSIS y pueden utilizarse como guía en otras secciones.  
  
Manual page man(1) line 1 (press h for help or q to quit)
```

10.El comando *history*, este nos enlistara hasta 500 comandos ejecutados previamente, reuniéndolos sin necesidad de volver a entrar.



```
user@user-VirtualBox: ~  
$ history  
1 clear  
2 ls  
3 ls /home  
4 clear  
5 sudo  
6 cd /etc  
7 ls -l host  
8 ls -l host*  
9 sudo nano hostname  
10 sudo root  
11 sudo su root  
12 ls /home  
13 clear  
14 mkdir  
15 mkdir --help  
16 clear  
17 ping  
18 clear  
19 ls  
20 ls /home  
21 ls /imagenes  
22 ls /documentos  
23 ls /Documentos  
24 cd  
25 clear  
26 mkdir  
27 ps  
28 clear  
29 ps  
30 clear  
31 ls -l  
32 ls -s  
33 ls -h  
34 ls -R  
35 CLEAR  
36 clear  
37 ls --help  
38 clear  
39 alias
```

CONCLUSION

Esta actividad es muy útil ya que con ella pudimos conocer de qué forma se utilizan los comandos, cual es la forma correcta de ejecutarlos o con que otras funciones se pueden combinar para realizar alguna acción específica y dentro de nuestro sistema operativo cual es la función que realizan.

Esto nos ayuda para realizar todo tipo de actividades desde las más básicas hasta otras más complejas, en donde algunas de ellas nos permiten realizarlas de una forma más rápida, también nos hace un menor consumo de recursos y en general todo esto nos permite tener un mejor control de nuestro sistema operativo.

En nuestra materia es muy básico aprender todos los comandos que se puedan implementar para poder realizar sin problema las actividades futuras que se puedan presentar, ya que sería una limitación para comprender lo que estamos realizando.

REFERENCIAS

Deyimar A. (oct 23, 2023) Comandos básicos de Linux que todo usuario debe saber.

[40 Comandos básicos de Linux que todo usuario debe saber \(hostinger.es\)](#)