

Actividad # 3 |

Comandos para el hardware

Sistemas Operativos I

Ingeniería en Desarrollo de
Software



TUTOR: Francisco Ortega Rivera

ALUMNO: Cristopher Eduardo Ramírez Calvillo

FECHA:06/05/2024

Indice

Introducción.....	3
Descripción.....	4
Justificación.....	5
Desarrollo.....	6
Etapa 1.....	6
Etapa 2.....	13
Etapa 3.....	19
Conclusión.....	24
Referencias.....	25

Introducción

Una de las tareas fundamentales al trabajar con sistemas operativos basados en Linux es la gestión y reconocimiento del hardware del sistema. Afortunadamente, Ubuntu ofrece una variedad de comandos a través de la terminal que permite a los usuarios obtener información detallada sobre los componentes de su equipo.

Para aquellos interesados en explorar y entender el hardware de su sistema existen comandos como **lshw**, **lscpu**, **free**, entre otros que proporcionan datos específicos del equipo. Son herramientas esenciales para los administradores de sistemas y entusiastas de la tecnología que buscan maximizar el rendimiento del hardware o solucionar problemas relacionados con el mismo.

Estos comandos son accesibles desde la terminal y pueden ser usados sin necesidad de instalar software adicional, lo que permite la flexibilidad y potencia del sistema operativo.

En resumen. Conocer y utilizar estos comandos es crucial para cualquier usuario de Ubuntu que quiera tener un control completo sobre su sistema y asegurarse de que todos los componentes estén en funcionamiento correcto y sean reconocidos por el sistema operativo.

Descripción

Imaginemos que estamos interesados en la mejora de la seguridad del sistema operativo. Para ello es fundamental conocer el hardware sobre el cual se ejecutan los procesos de seguridad. Los comandos de Ubuntu para la inspección del hardware son herramientas valiosas en este contexto. Por ejemplo, comandos como **lsusb** es esencial para identificar dispositivos USB conectados, lo que es crucial para detectar dispositivos no autorizados o potencialmente maliciosos. Es de suma importancia darle el tiempo necesario al aprendizaje de estos comandos sobre todo si está en tus planes estudiar o dedicarte laboralmente a esto, entender el hardware del sistema operativo va más allá de la simple curiosidad; es una cuestión de eficiencia y rendimiento. Al conocer a fondo los componentes que tienes puedes tomar decisiones informadas sobre actualizaciones, compatibilidad de software y resolución de problemas. Poder evaluar el rendimiento del sistema y adaptar configuraciones para optimizar el uso de los recursos puede llevar a una experiencia de usuario más fluida y productiva.

Justificación

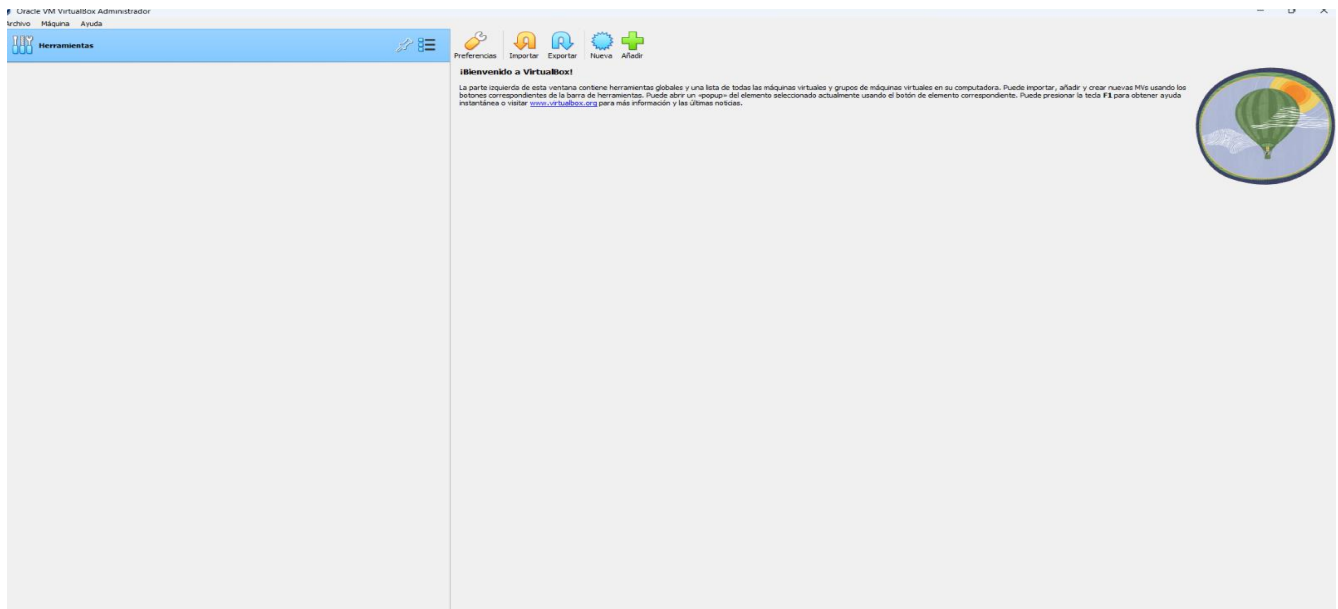
Saber cómo emplear los comandos para conocer el hardware en Ubuntu es importante por varias razones:

1. Diagnóstico y solución de problemas: Conocer los detalles del hardware puede ayudar a diagnosticar problemas. Por ejemplo, si una aplicación no funciona correctamente, podría deberse a la falta de recursos del sistema o a un hardware incompatible.
2. Optimización del rendimiento: Los comandos permiten verificar el uso de recursos en tiempo real, como la CPU o la memoria RAM, lo que es esencial para optimizar el rendimiento del sistema.
3. Actualizaciones y mejoras: Antes de realizar actualizaciones o mejoras del sistema, es crucial conocer las especificaciones actuales del hardware para asegurarse de que los nuevos componentes sean compatibles.
4. Seguridad: Algunos comandos pueden revelar información sobre dispositivos que podrían ser vulnerables a ataques, permitiendo tomar medidas preventivas.
5. Administración de sistemas: Para los administradores de sistemas, es fundamental conocer la configuración del hardware para gestionar eficientemente los recursos y planificar futuras expansiones y mantenimientos.

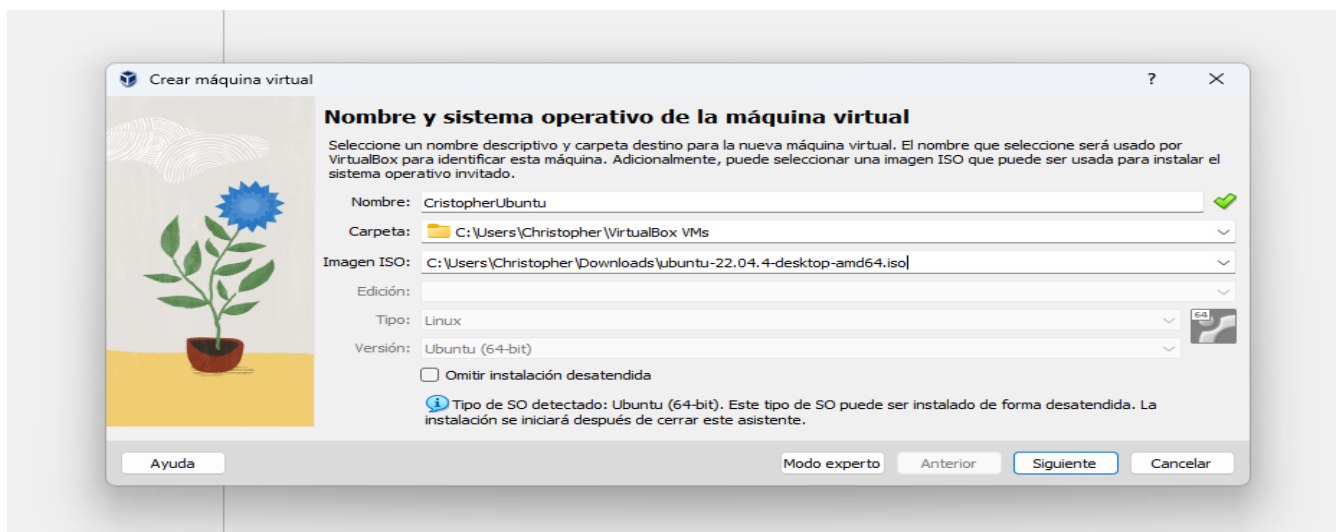
Desarrollo

Etapa 1

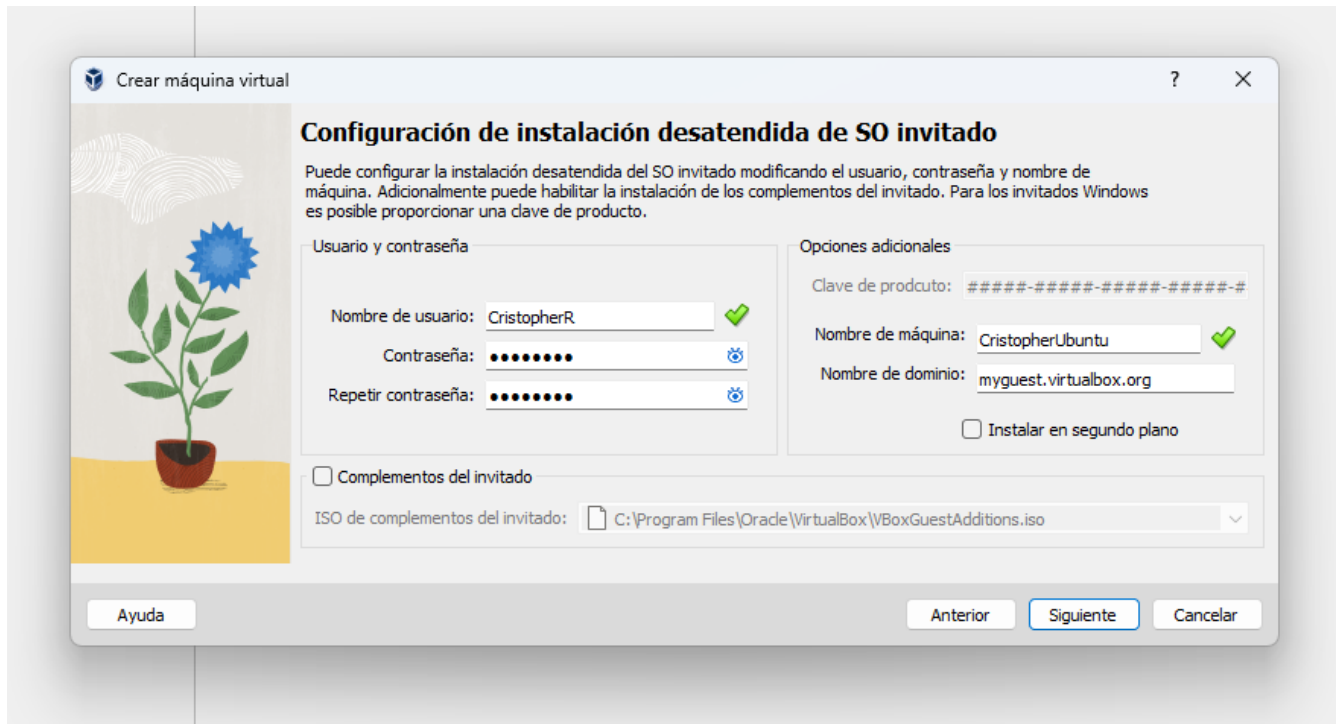
Vamos a ver parte del proceso de instalación y configuración, por lo menos del SO Ubuntu



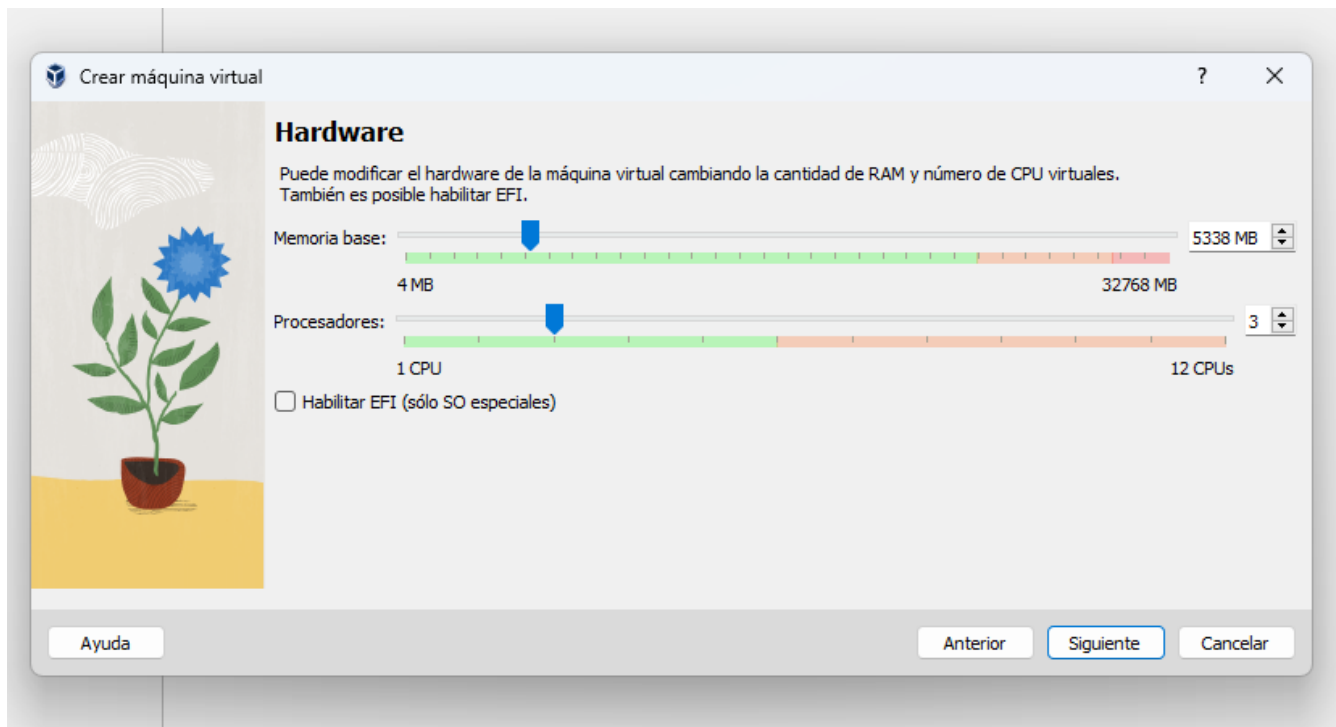
Iniciamos con la apertura de VirtualBox, en la parte superior media nos arroja varias opciones, daremos clic en "nueva" para poder agregar una nueva máquina virtual.



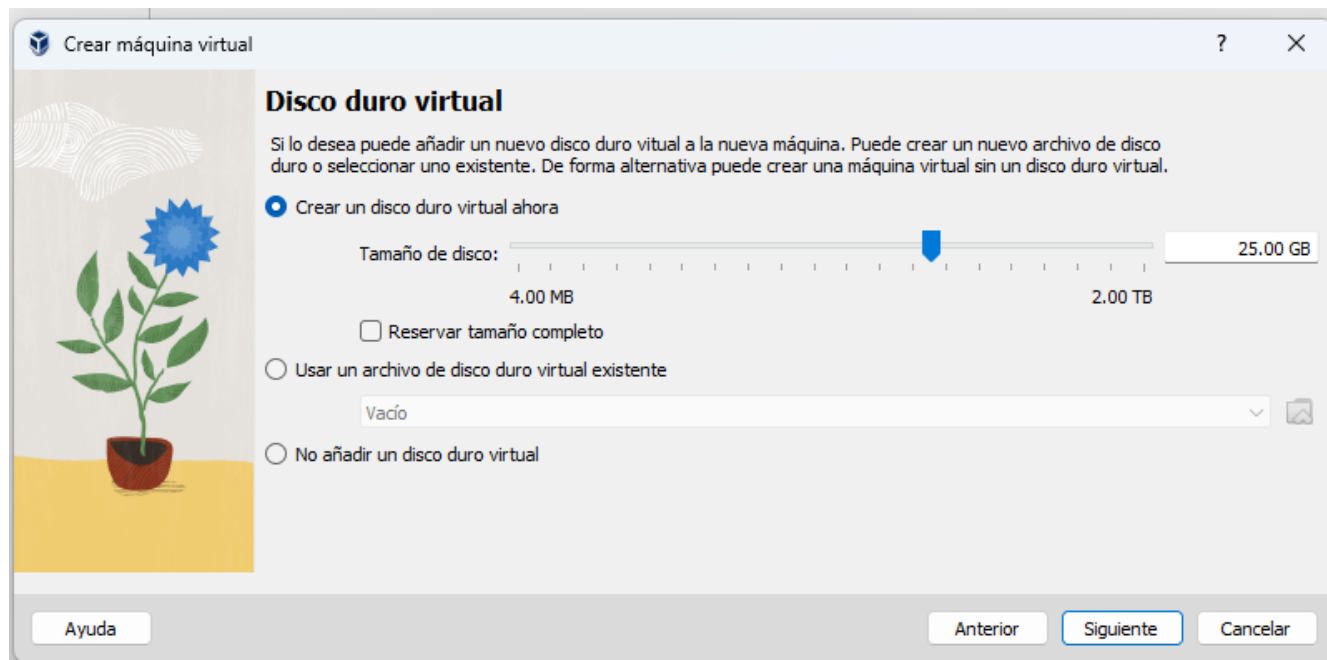
Nos abrirá una pantalla donde podremos cambiar el nombre y carpeta de la nueva máquina virtual al igual que la imagen ISO del SO que vamos a probar, esta última ya tendríamos que tenerla descargada previamente.



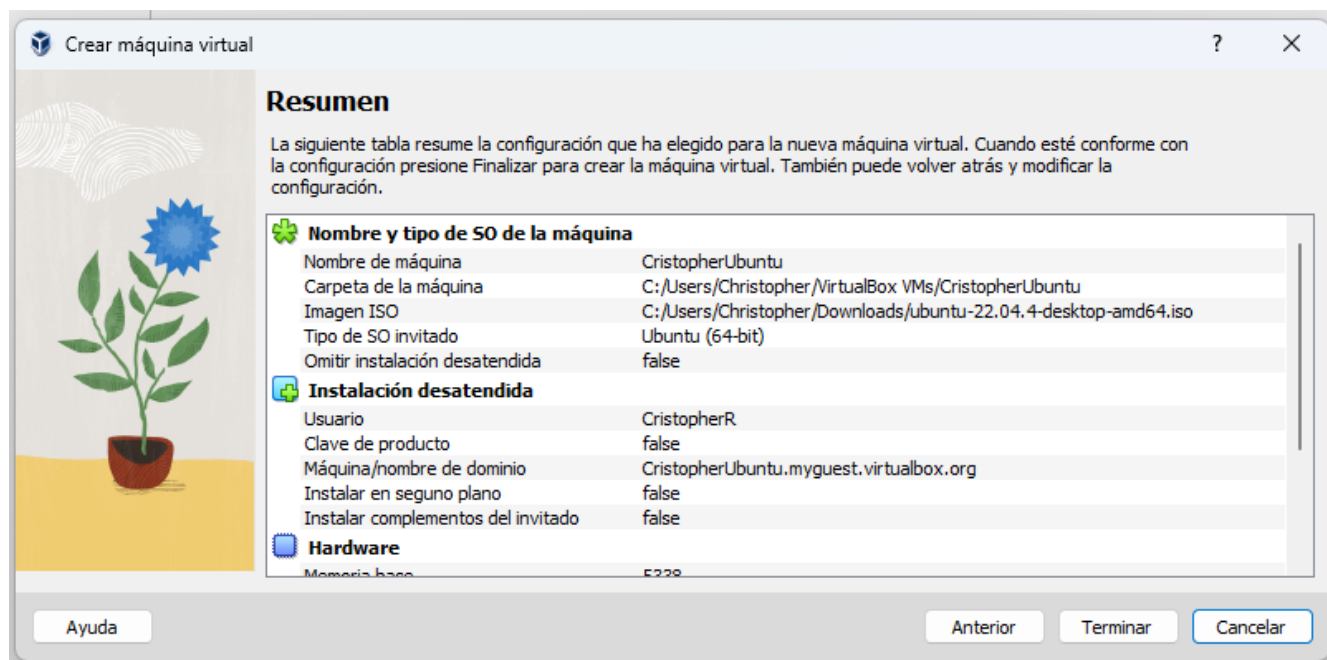
Por seguridad, modificaremos tanto el nombre de usuario como la contraseña de acceso, muy importante recordarla.



Le daremos potencia a esta preciosidad virtual, los parámetros vienen por defecto, pero si te lo puedes permitir y le vas a dar uso constante, que valga la pena.

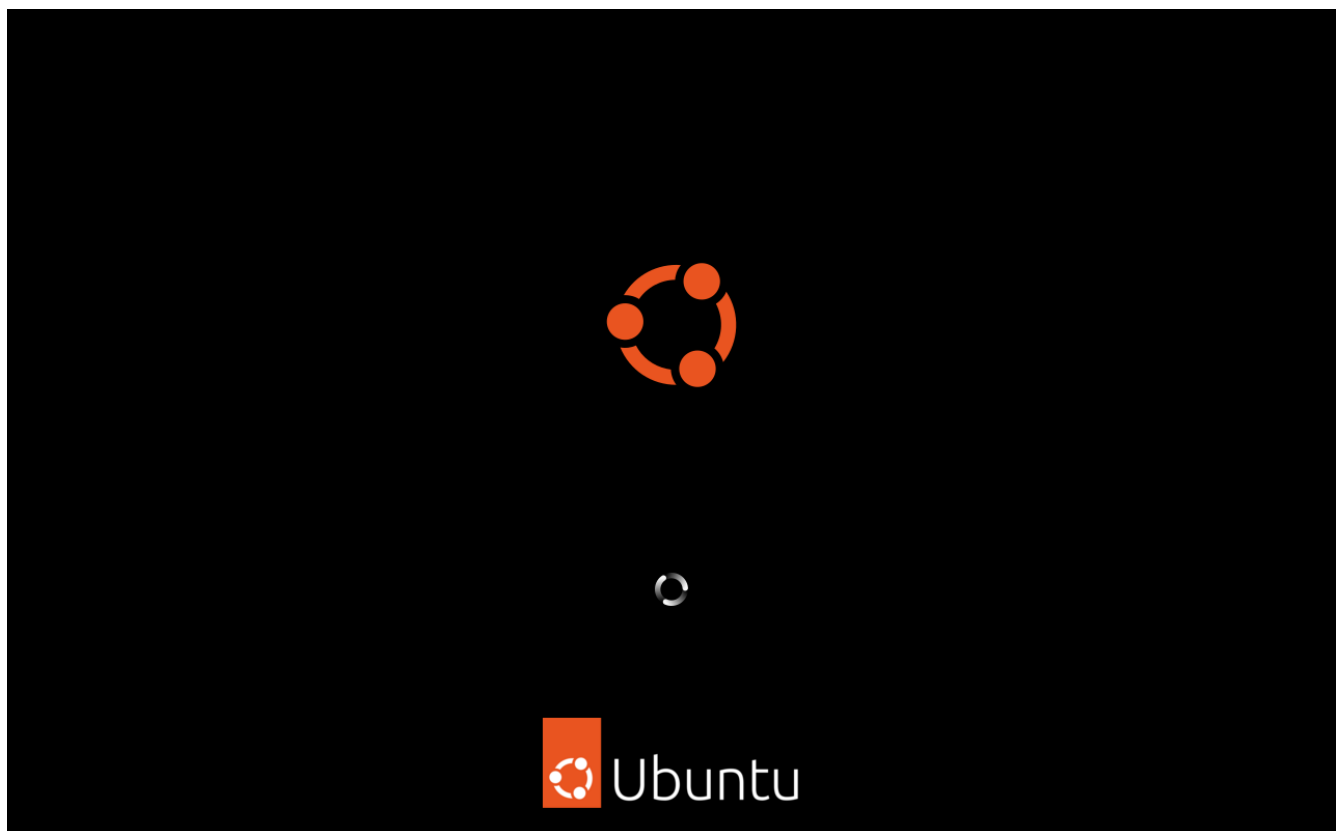
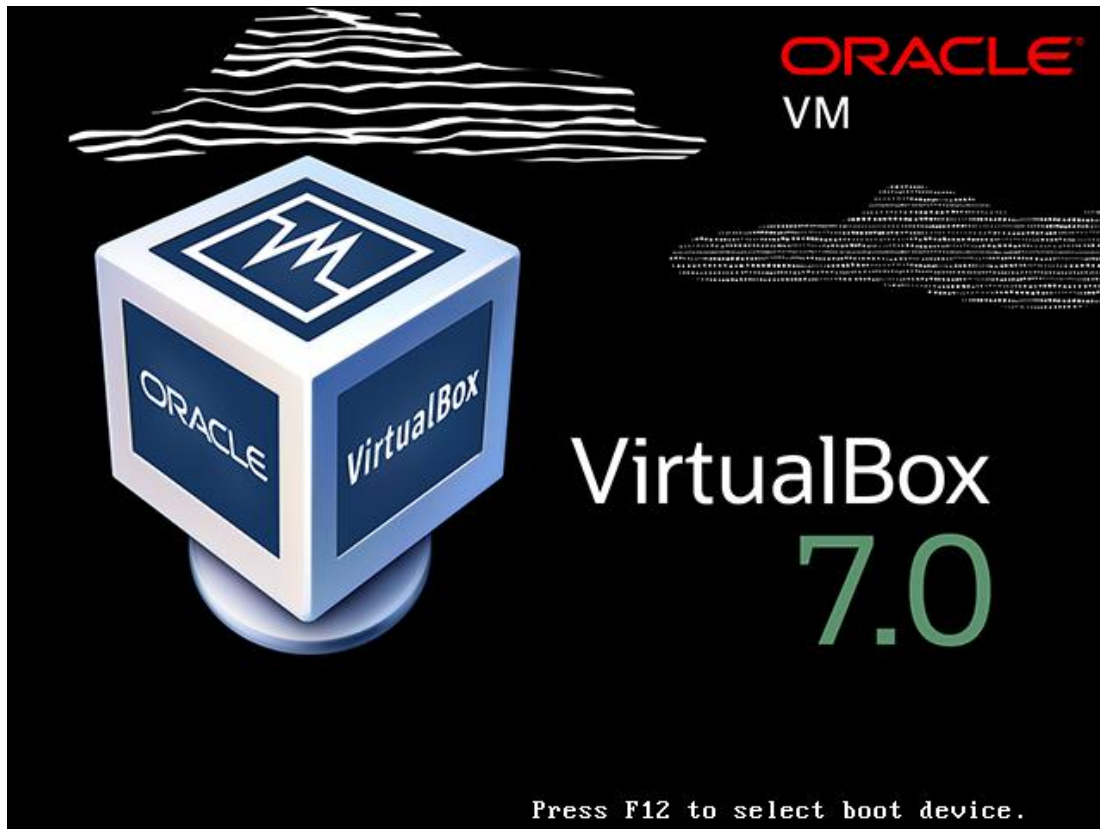


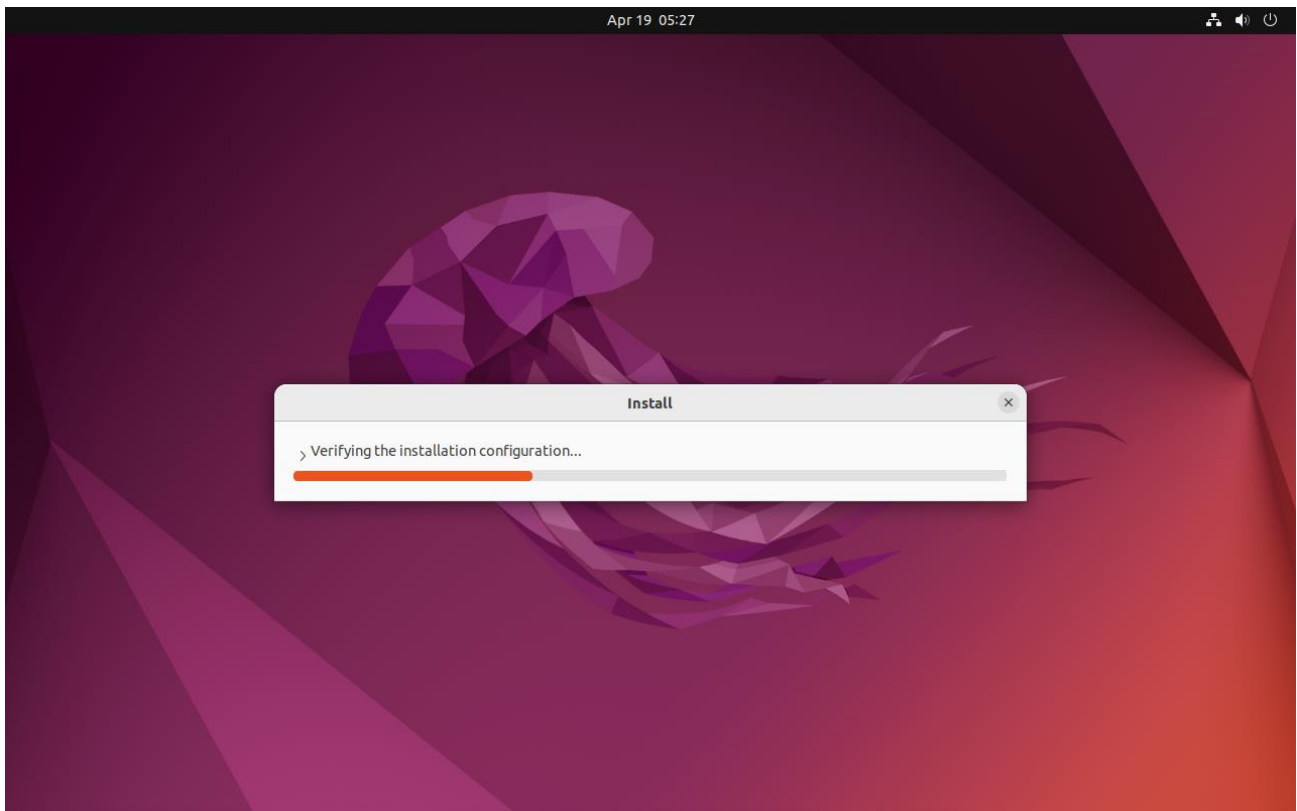
Se recomienda también el almacenamiento, aunque depende mucho del uso que se pretenda dar a la máquina virtual.



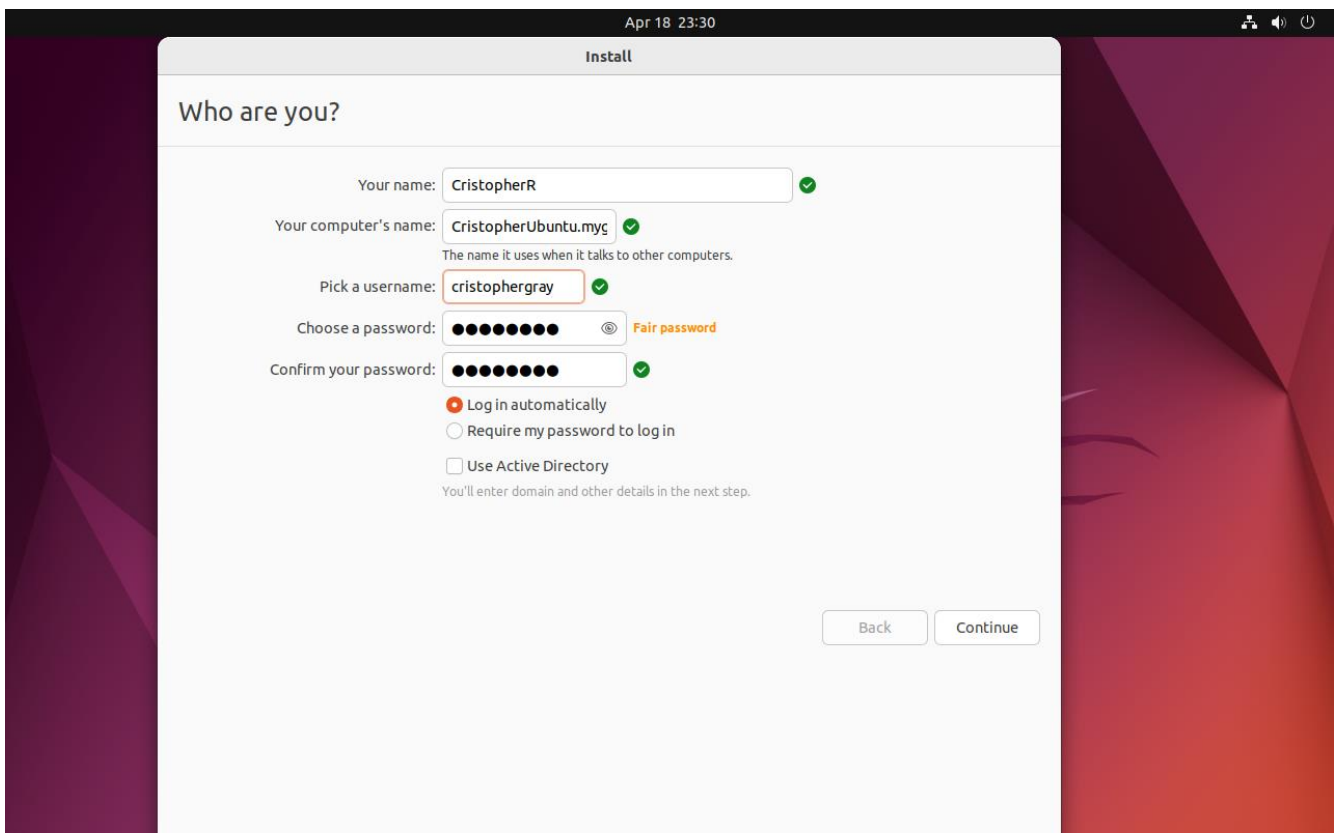
Terminando la configuración básica nos va a dar un resumen antes de la creación de la máquina virtual, vamos a verificar que todo esté en orden y como queremos.

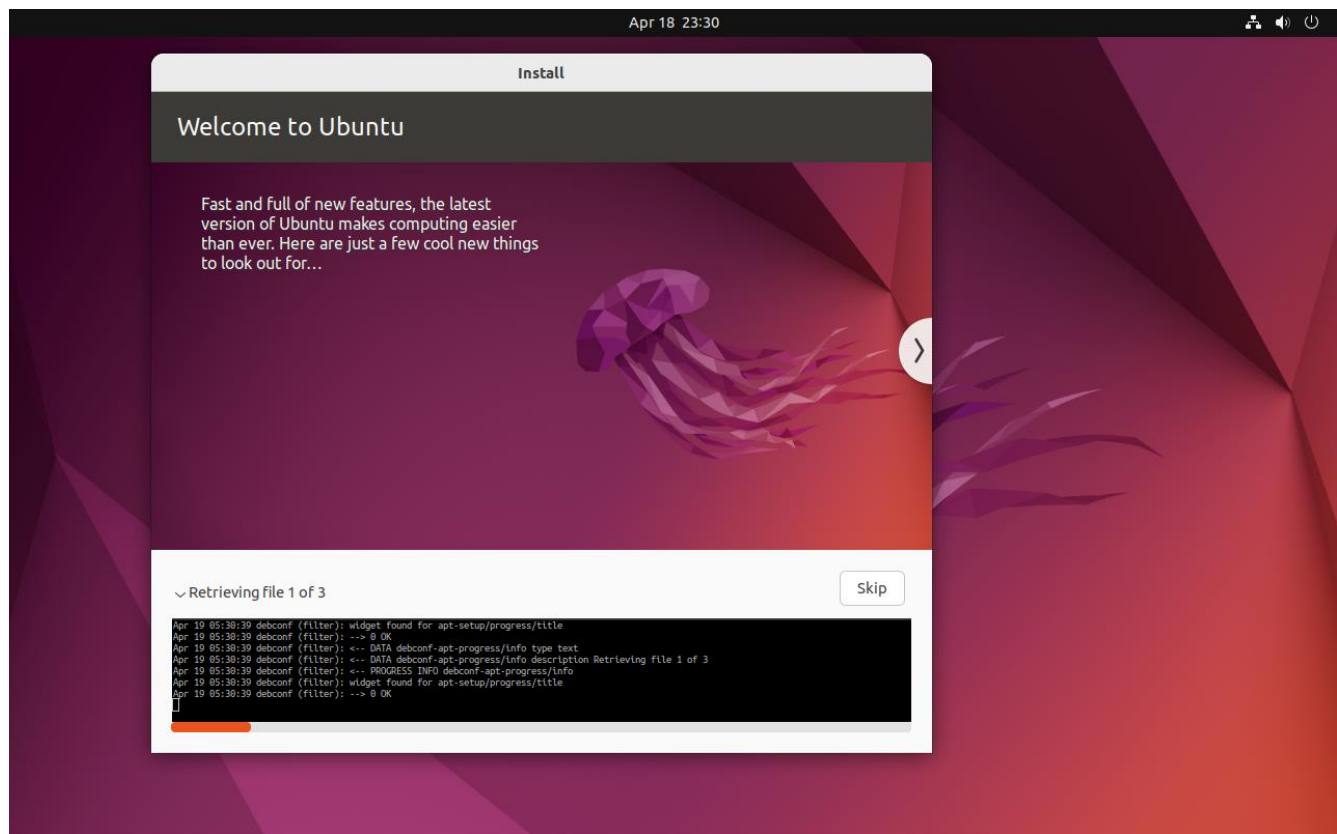
Siguiendo el proceso mayormente lo que sigue es esperar...



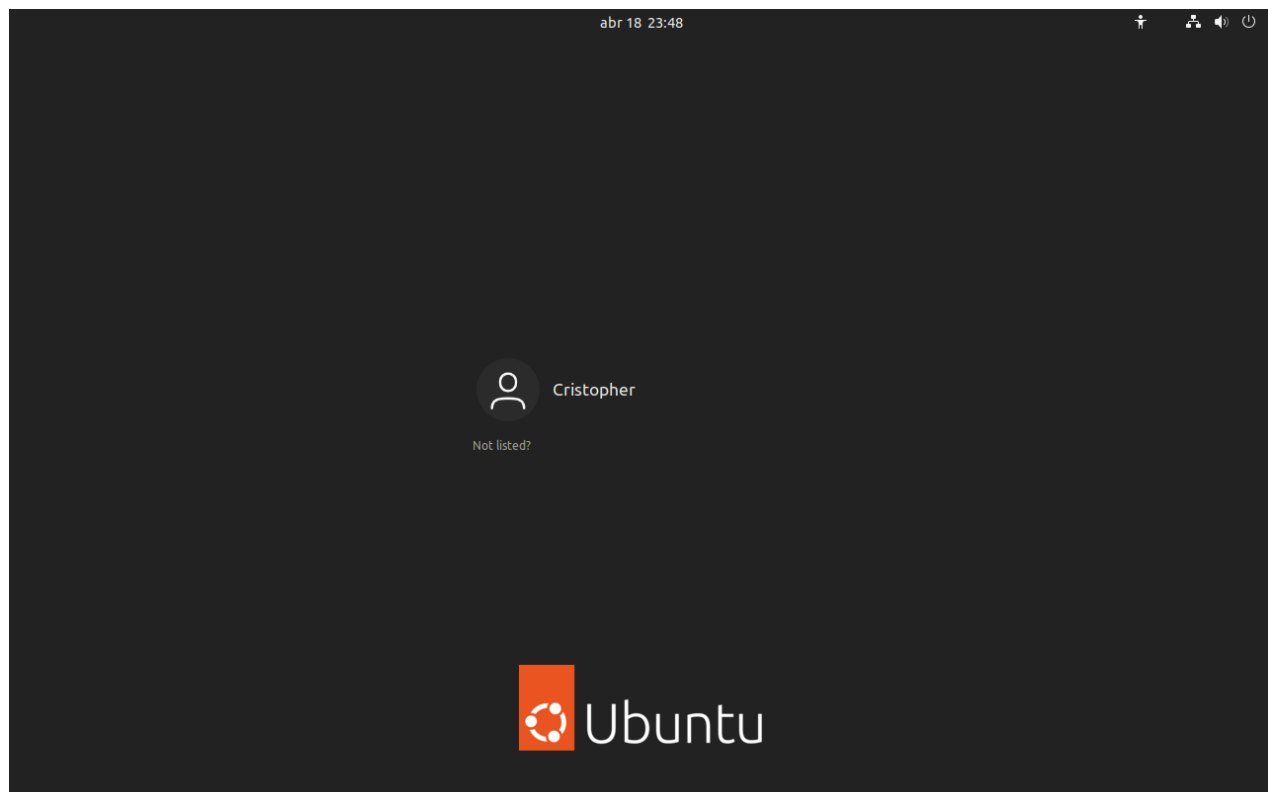


Iniciara el proceso de instalación de las configuraciones previamente seleccionadas.



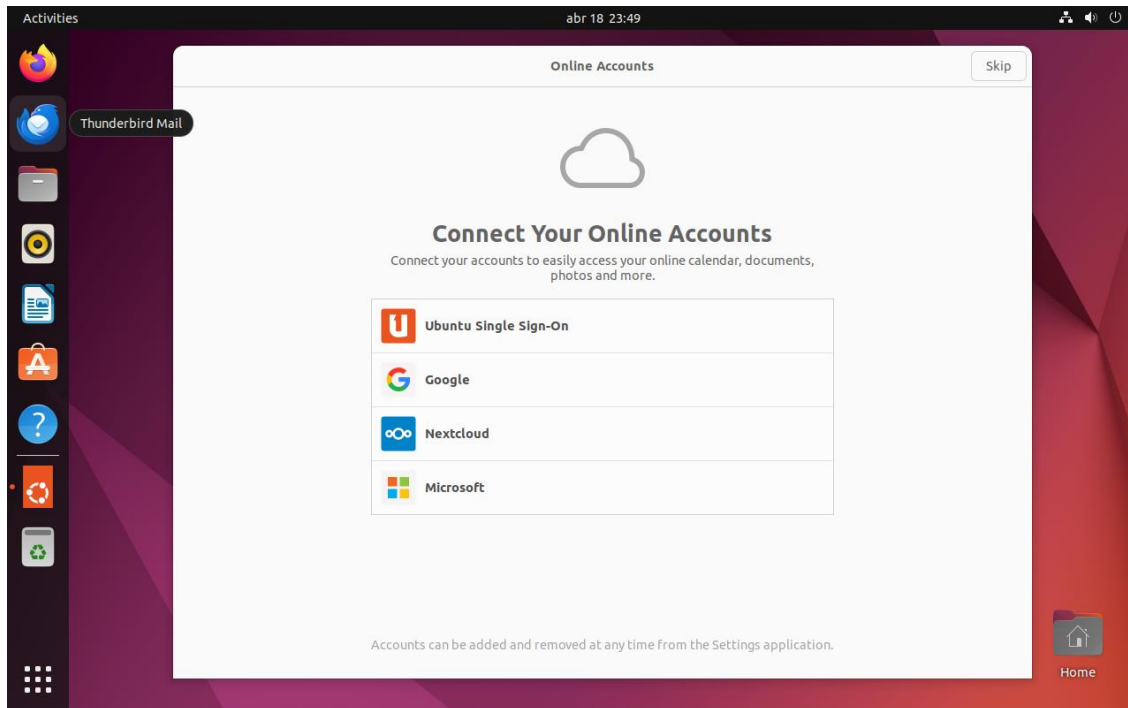


Ya casi terminamos...

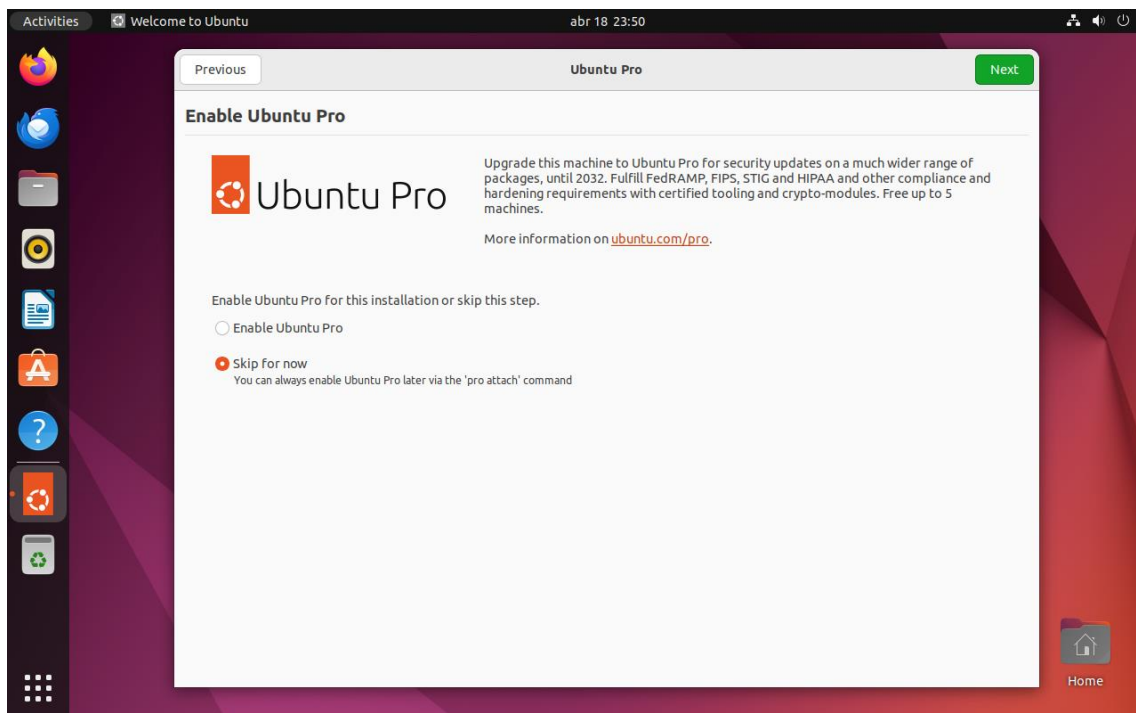


Como ya lo tenía previamente instalado ya me arroja en automático una sesión para poder continuar el

proceso.



Junto con esto, nos da la opción de vincular alguna cuenta.

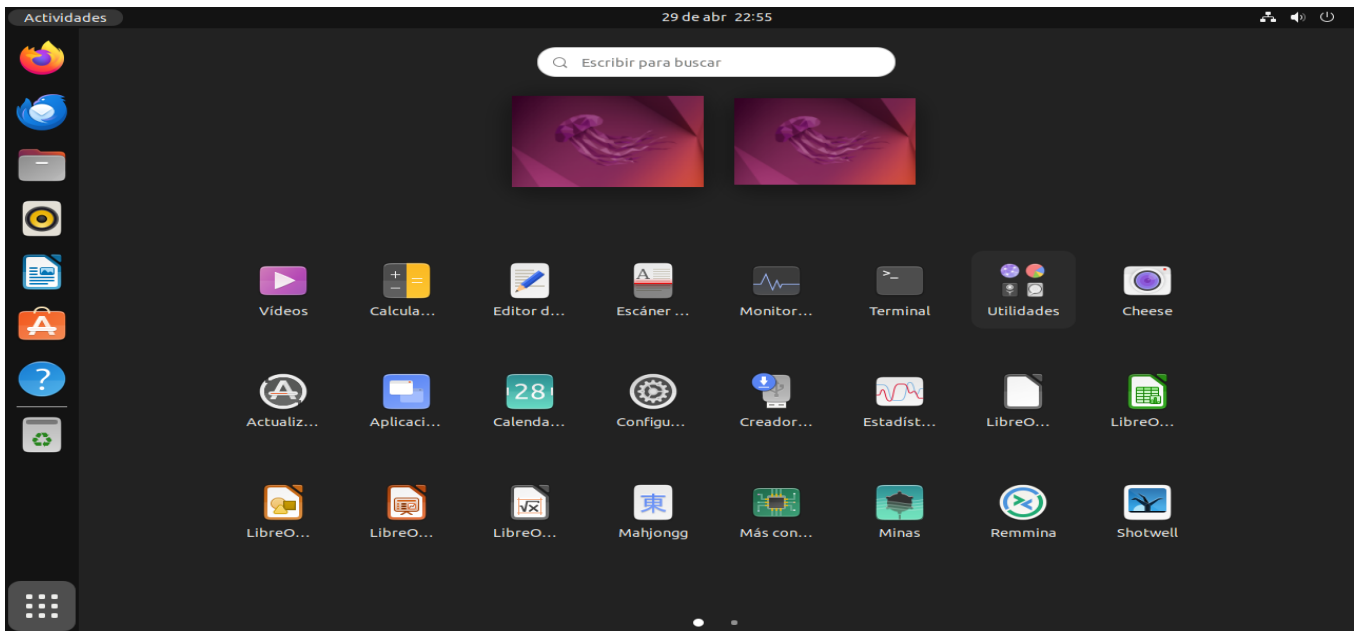


Y también poder adquirir la versión Pro.

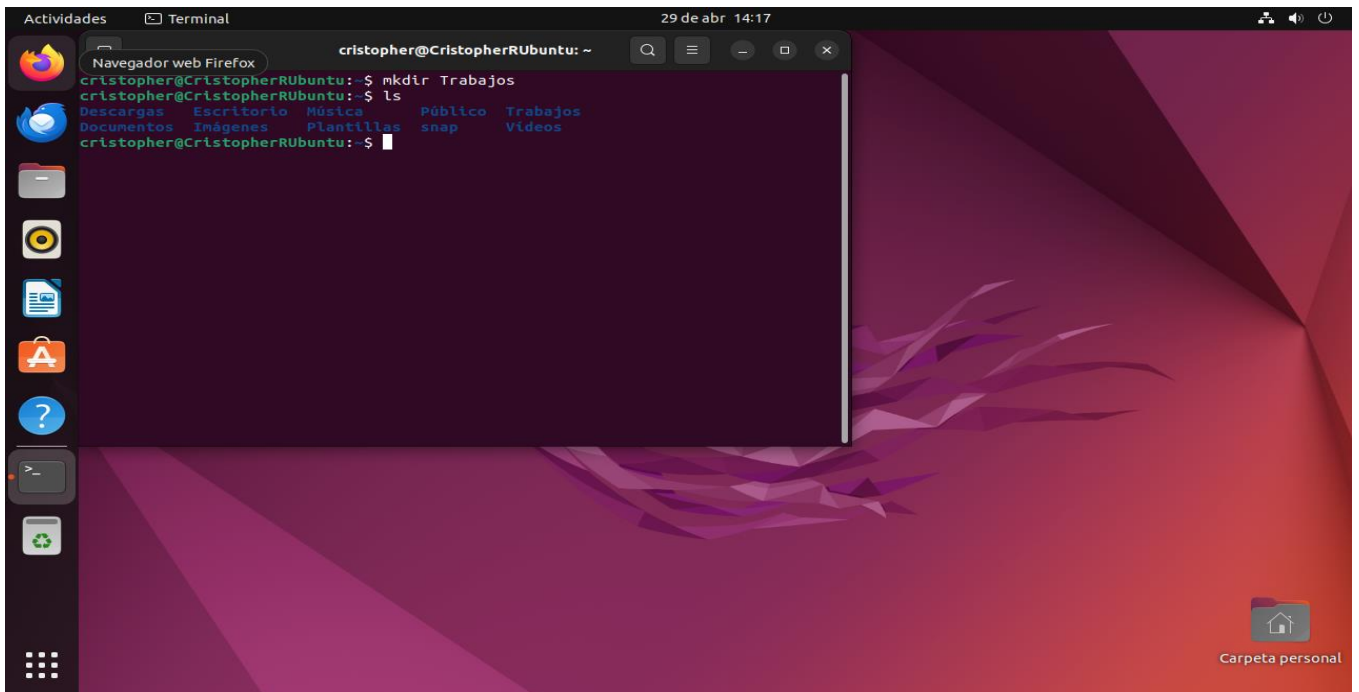
Terminado el proceso ya tendremos nuestra máquina virtual completamente funcional.

Etapa 2

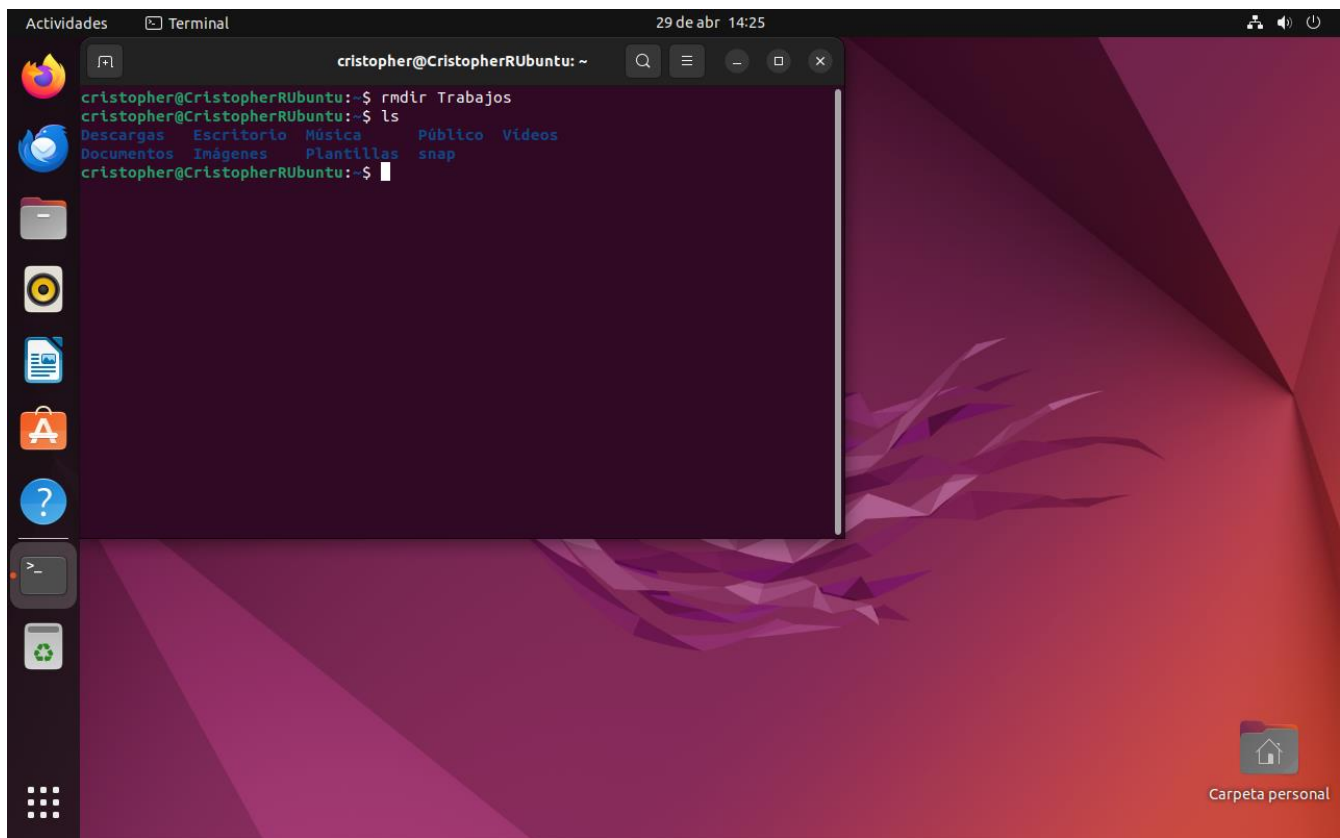
Para esta parte del trabajo iniciaremos abriendo la terminal desde el menú.



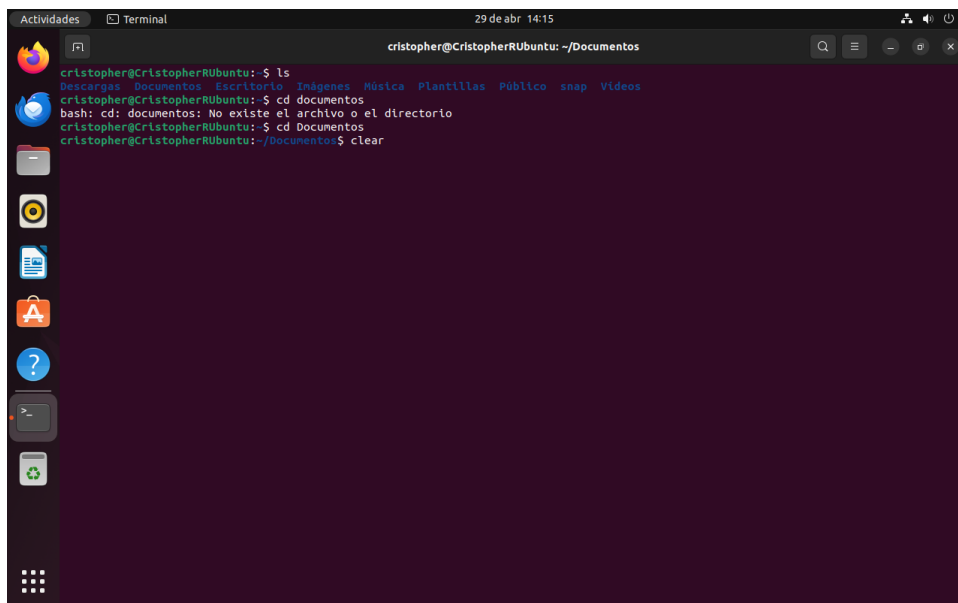
Vamos a ver algunos de los comandos básicos de Ubuntu y su funcionalidad



Empezamos con el comando **mkdir** el cual nos permite crear directorios (carpetas) vacías, en conjunto usamos el comando **ls** para poder verificar la lista de directorios existentes.

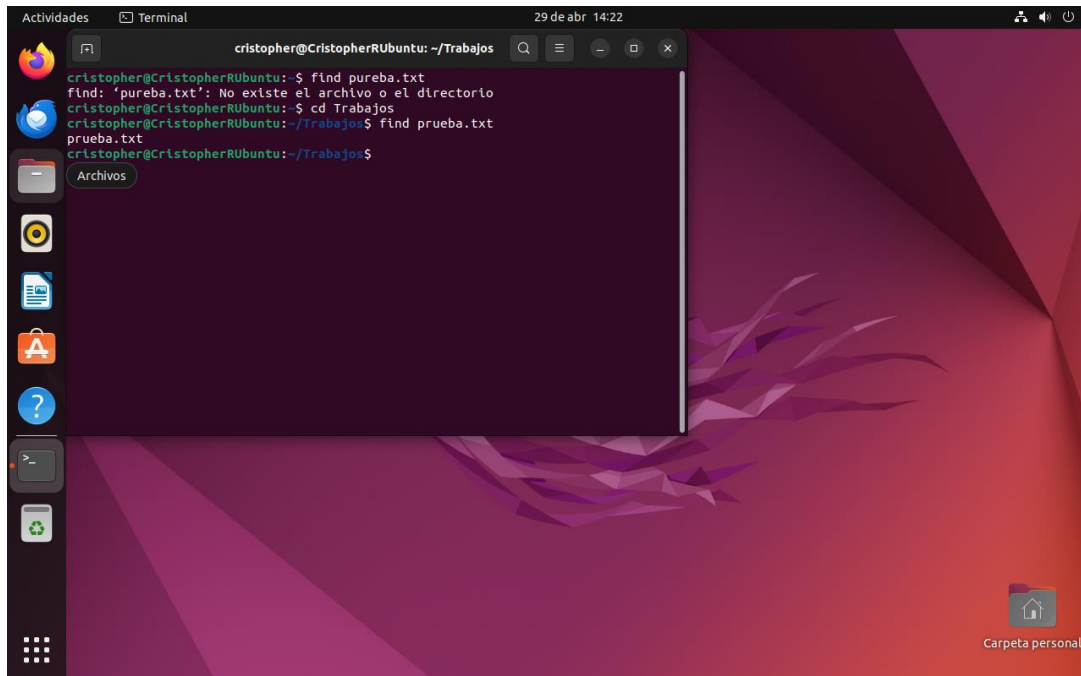


Si no requieres alguno de los directorios existentes o que hayas creado en algún momento, usaremos el comando **rmdir** para eliminarlo.



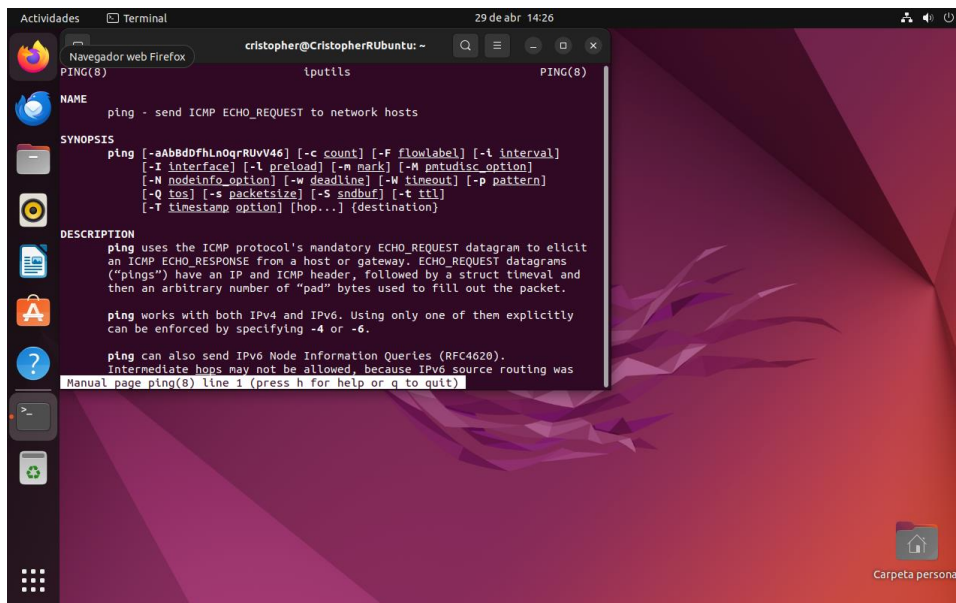
Para movernos entre los directorios usaremos el comando **cd**, muy importante escribir tal cual el nombre del directorio al cual deseamos ir, de lo contrario nos arroja que el archivo o directorio no

existe.



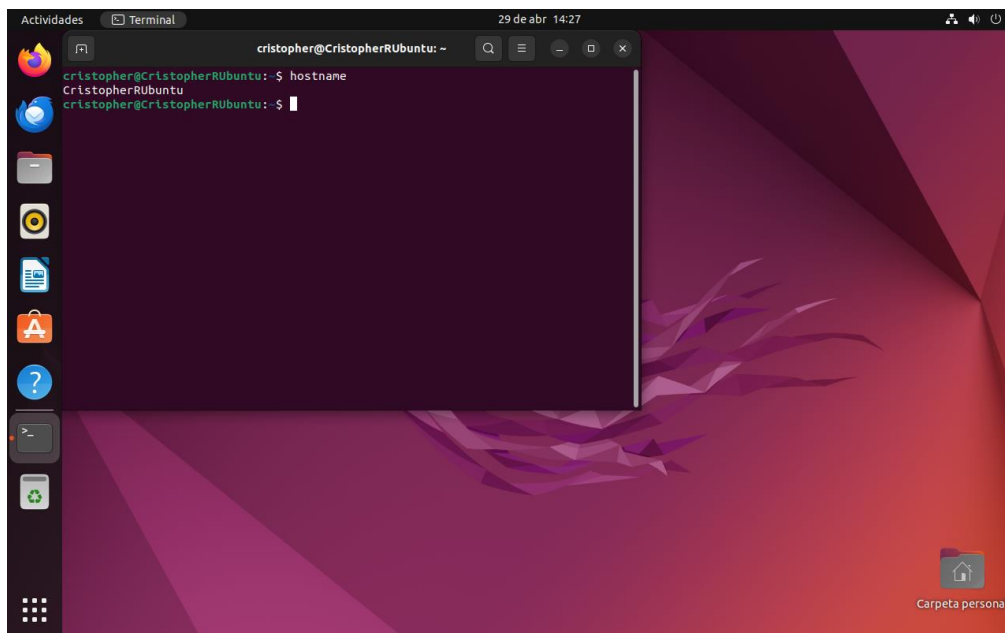
```
crisopher@CristopherRUbuntu: ~/Trabajos
crisopher@CristopherRUbuntu:~$ find pureba.txt
find: 'pureba.txt': No existe el archivo o el directorio
crisopher@CristopherRUbuntu:~$ cd Trabajos
crisopher@CristopherRUbuntu:~/Trabajos$ find prueba.txt
prueba.txt
crisopher@CristopherRUbuntu:~/Trabajos$
```

Por otro lado, si no encontramos donde dejamos algún archivo o directorio utilizaremos el comando **find**.

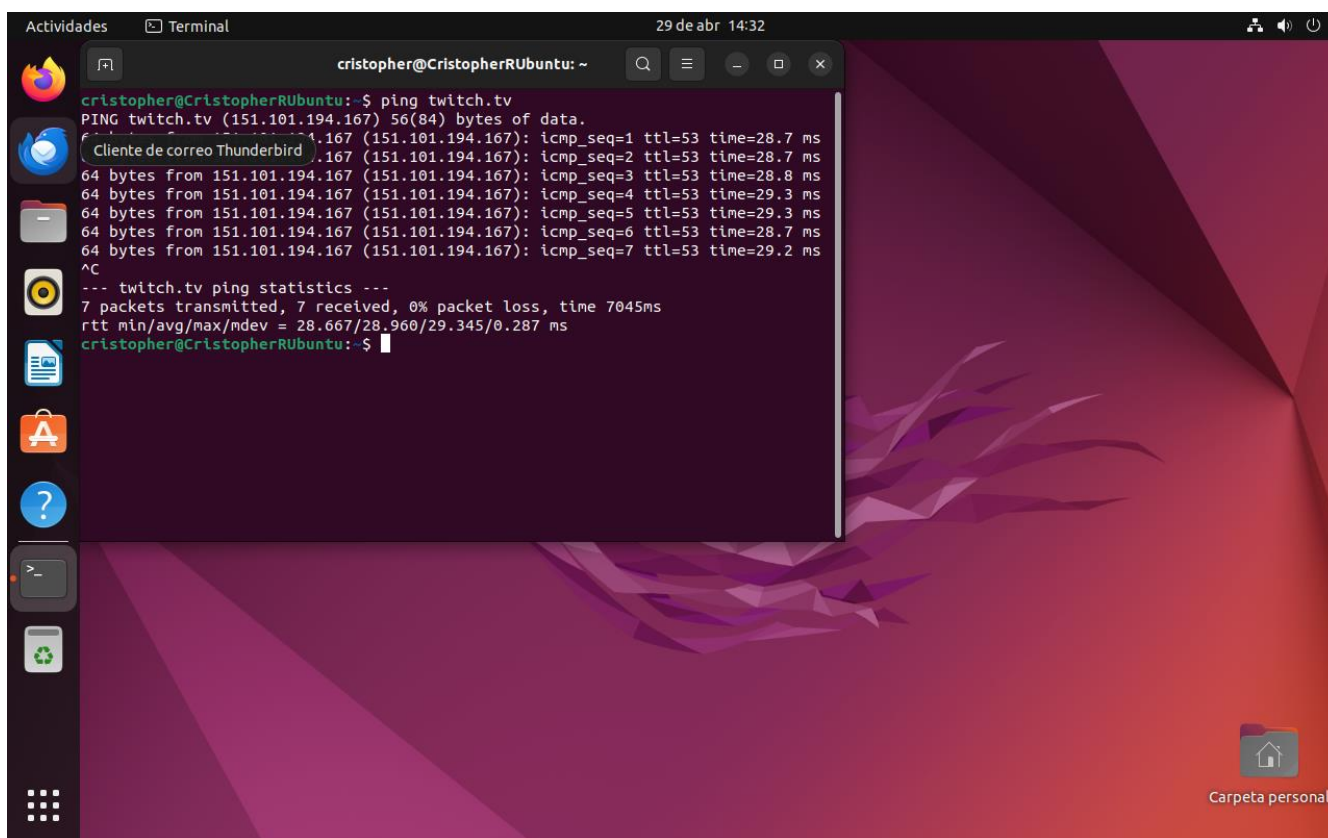


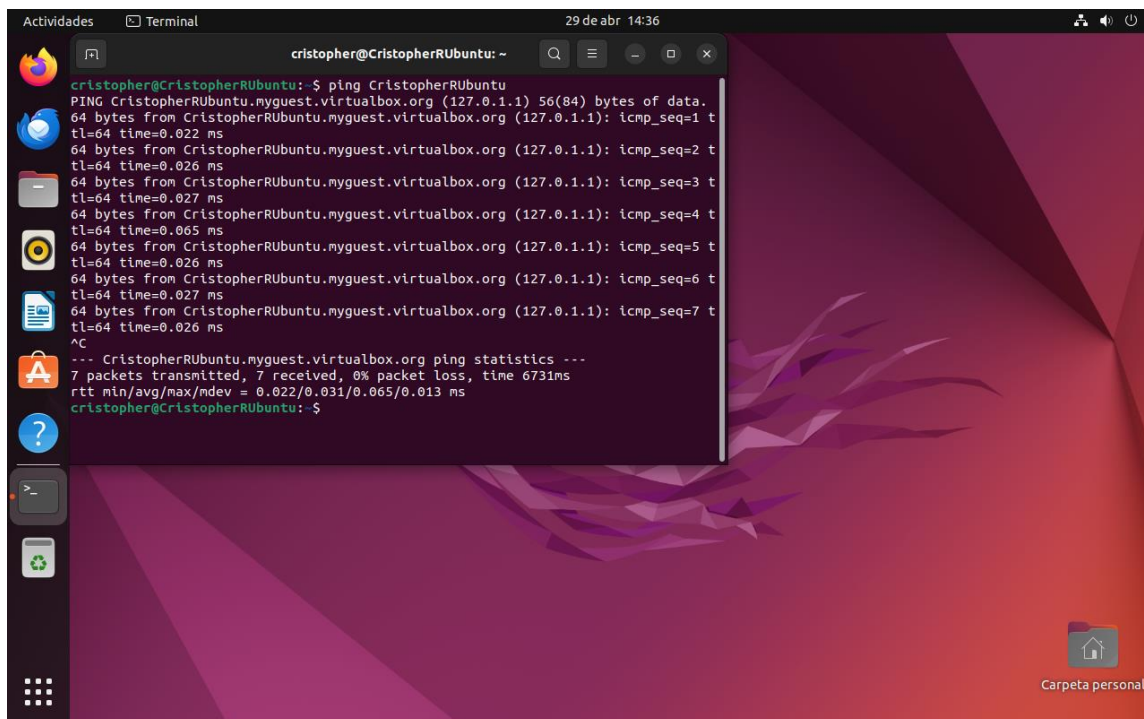
```
crisopher@CristopherRUbuntu: ~
PING(8)
NAME
    ping - send ICMP ECHO_REQUEST to network hosts
SYNOPSIS
    ping [-aAbBdDfHlNoqrRUVvV46] [-c count] [-F flowlabel] [-i interval]
        [-I interface] [-l preload] [-m mark] [-M mtu] [-n nodisc_option]
        [-M nodeinfo_option] [-w deadline] [-W timeout] [-p pattern]
        [-Q tos] [-s packetsize] [-S sndbuf] [-t ttl]
        [-T timestamp_option] [hop...] {destination}
DESCRIPTION
    ping uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit
    an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams
    ("pings") have an IP and ICMP header, followed by a struct timeval and
    then an arbitrary number of "pad" bytes used to fill out the packet.
    ping works with both IPv4 and IPv6. Using only one of them explicitly
    can be enforced by specifying -4 or -6.
    ping can also send IPv6 Node Information Queries (RFC4620).
    Intermediate hops may not be allowed, because IPv6 source routing was
    Manual page ping(8) line 1 (press h for help or q to quit)
```

Si queremos saber de la funcionalidad de alguno de los comandos va a ser tan sencillo de usar el comando **man** seguido del nombre del comando del cual queremos información.



El **hostname** nos brindara información sobre el usuario que está usando el sistema.

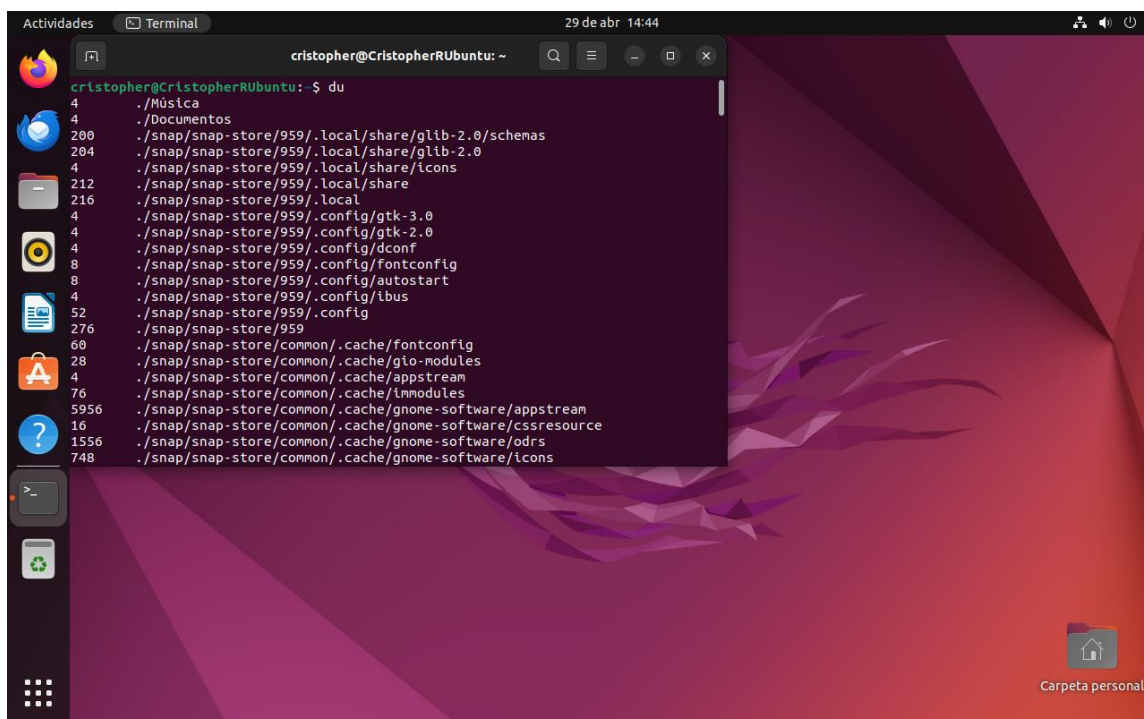




A screenshot of a Linux desktop environment with a terminal window open. The terminal shows the execution of a `ping` command against `CristopherRUbuntu.myquest.virtualbox.org` (IP: 127.0.1.1). The output displays seven successful ping requests, each with a 64-byte payload and a response time of approximately 0.022 to 0.026 ms. Below the individual pings, a summary line indicates that 7 packets were transmitted, 7 were received, and there was 0% packet loss, with an average round-trip time of 0.022 ms.

```
cristopher@CristopherRUbuntu: ~  
$ ping CristopherRUbuntu  
PING CristopherRUbuntu.myquest.virtualbox.org (127.0.1.1) 56(84) bytes of data:  
64 bytes from CristopherRUbuntu.myquest.virtualbox.org (127.0.1.1): icmp_seq=1 t  
tl=64 time=0.022 ms  
64 bytes from CristopherRUbuntu.myquest.virtualbox.org (127.0.1.1): icmp_seq=2 t  
tl=64 time=0.026 ms  
64 bytes from CristopherRUbuntu.myquest.virtualbox.org (127.0.1.1): icmp_seq=3 t  
tl=64 time=0.027 ms  
64 bytes from CristopherRUbuntu.myquest.virtualbox.org (127.0.1.1): icmp_seq=4 t  
tl=64 time=0.065 ms  
64 bytes from CristopherRUbuntu.myquest.virtualbox.org (127.0.1.1): icmp_seq=5 t  
tl=64 time=0.026 ms  
64 bytes from CristopherRUbuntu.myquest.virtualbox.org (127.0.1.1): icmp_seq=6 t  
tl=64 time=0.027 ms  
64 bytes from CristopherRUbuntu.myquest.virtualbox.org (127.0.1.1): icmp_seq=7 t  
tl=64 time=0.026 ms  
^C  
--- CristopherRUbuntu.myquest.virtualbox.org ping statistics ---  
7 packets transmitted, 7 received, 0% packet loss, time 6731ms  
rtt min/avg/max/mdev = 0.022/0.031/0.065/0.013 ms  
cristopher@CristopherRUbuntu: $
```

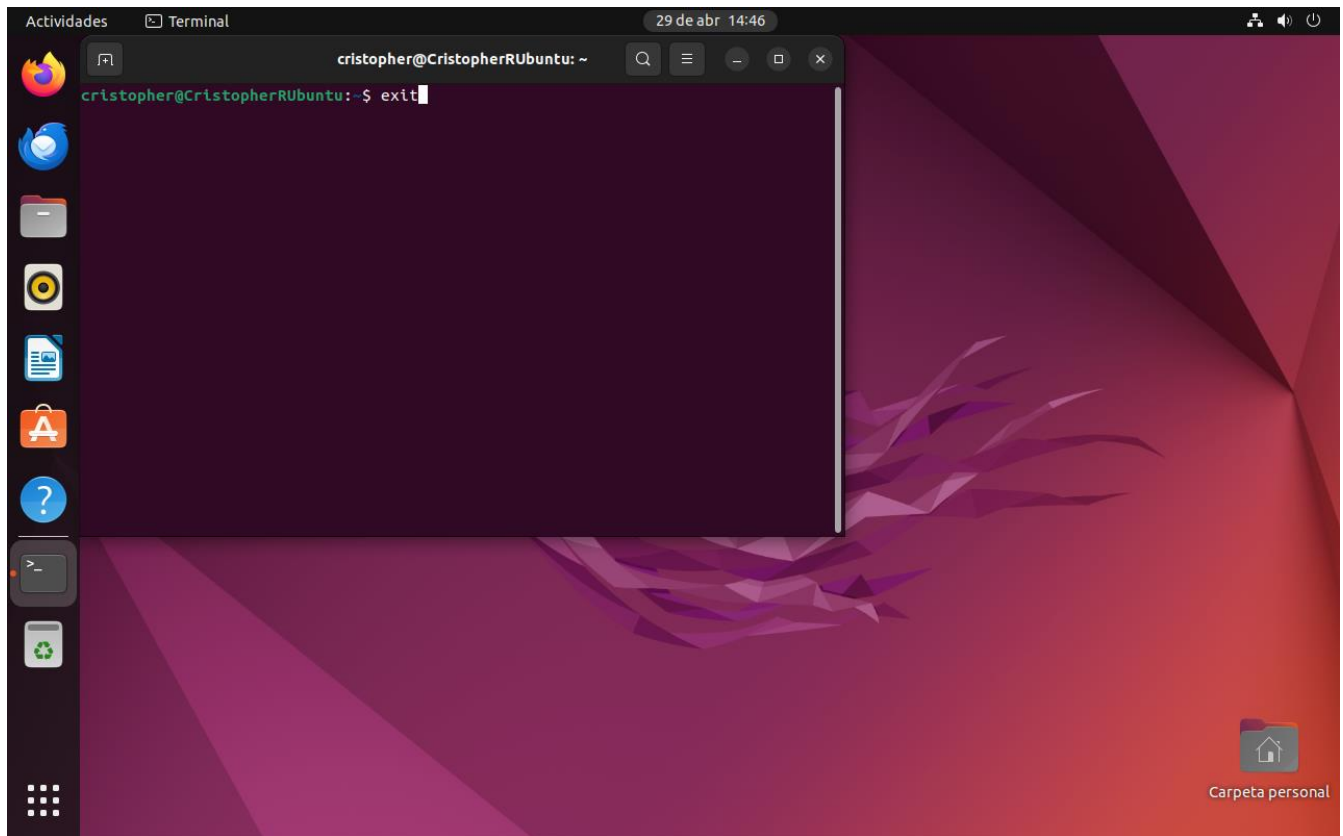
El comando **ping** nos brinda información de la conexión a un servidor por medio del tiempo que tarda la información de ida y vuelta, a su vez también nos da información de la conexión a internet.



A screenshot of a Linux desktop environment with a terminal window open. The terminal shows the execution of the `du` command, which displays the disk usage of various files and directories. The output lists several paths, including `./Música`, `./Documentos`, and various snap store directories, along with their respective disk usage in bytes.

```
cristopher@CristopherRUbuntu: ~  
$ du  
4      ./Música  
4      ./Documentos  
200    ./snap/snap-store/959/.local/share/glib-2.0/schemas  
204    ./snap/snap-store/959/.local/share/glib-2.0  
4      ./snap/snap-store/959/.local/share/icons  
212    ./snap/snap-store/959/.local/share  
216    ./snap/snap-store/959/.local  
4      ./snap/snap-store/959/.config/gtk-3.0  
4      ./snap/snap-store/959/.config/gtk-2.0  
4      ./snap/snap-store/959/.config/dconf  
8      ./snap/snap-store/959/.config/fontconfig  
8      ./snap/snap-store/959/.config/autostart  
4      ./snap/snap-store/959/.config/ibus  
52     ./snap/snap-store/959/.config  
276    ./snap/snap-store/959  
60     ./snap/snap-store/common/.cache/fontconfig  
28     ./snap/snap-store/common/.cache/gio-modules  
4      ./snap/snap-store/common/.cache/appstream  
76     ./snap/snap-store/common/.cache/ibus  
5956   ./snap/snap-store/common/.cache/gnome-software/appstream  
16     ./snap/snap-store/common/.cache/gnome-software/cssresource  
1556   ./snap/snap-store/common/.cache/gnome-software/odrs  
748    ./snap/snap-store/common/.cache/gnome-software/icons
```

Entre los archivos creados todo genera espacio de almacenamiento el cual podemos visualizar con el comando **du**.

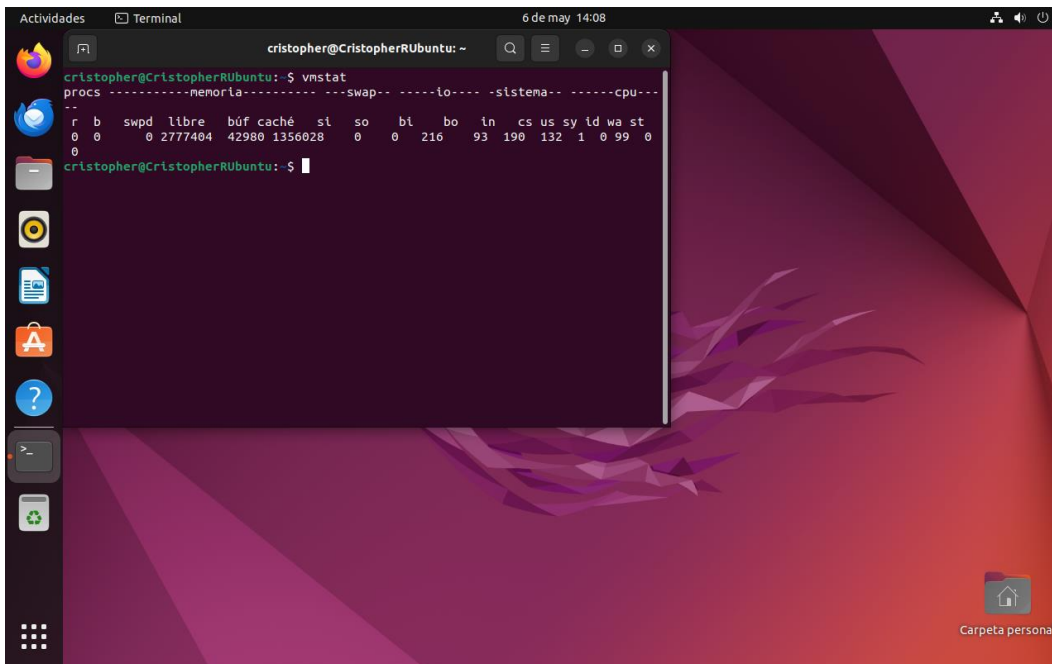


Al finalizar, podemos usar el comando **clear** para limpiar todo y volver al inicio de la terminal, si hemos concluido con el trabajo podemos usar la "tachita" que se encuentra en la parte superior derecha de la terminal o en todo caso el comando **exit**.

Nota importante, todos los comandos tienen que ser escritos en minúsculas, por otro lado, archivos y directorios que estemos empleando se tiene que escribir tal cual se encuentran en existencia de lo contrario no podremos seguir con el proceso.

Etapa 3

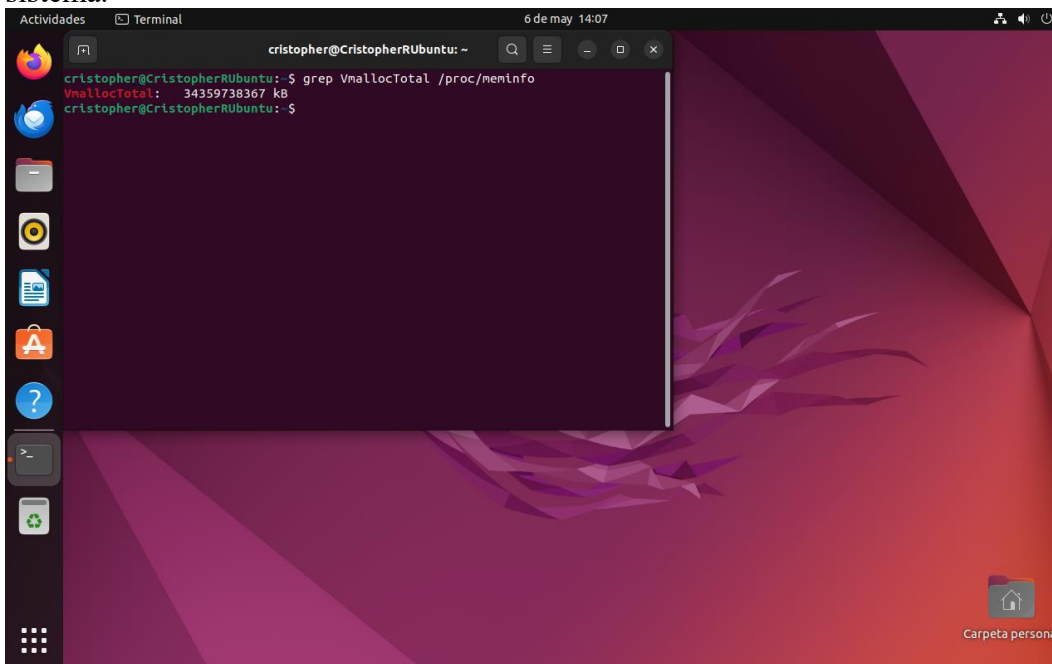
Para el proyecto final vamos a utilizar algunos de los comandos enfocados al reconocimiento de hardware del equipo, por ejemplo:



A screenshot of a Linux desktop environment with a terminal window open. The terminal shows the command `vmstat` being executed, displaying system statistics in a table format. The desktop background is a red and purple geometric pattern, and the left sidebar contains various application icons.

```
cristopher@CristopherRUbuntu: ~  
cristopher@CristopherRUbuntu:~$ vmstat  
procs -----memoria----- --swap-- -----lo---- -sistema-- -----cpu---  
..  
r b swpd libre búf caché st so bi bo in cs us sy id wa st  
0 0 0 2777404 42980 1356028 0 0 0 216 93 190 132 1 0 99 0  
0  
cristopher@CristopherRUbuntu:~$
```

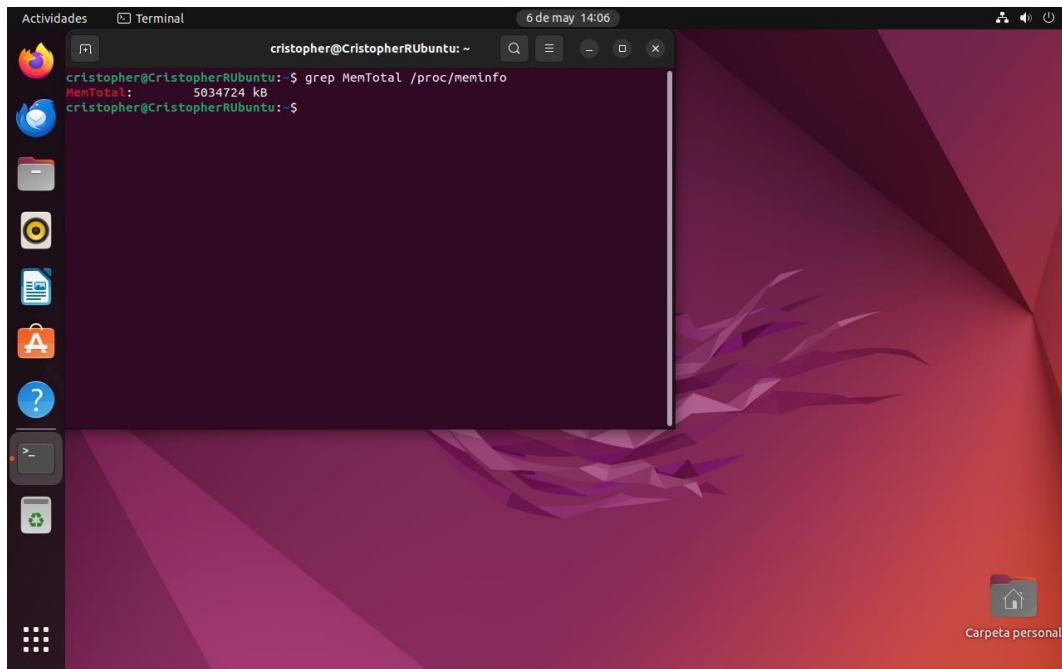
El comando **vmstat** que nos brinda información acerca de las estadísticas de la memoria virtual del sistema.



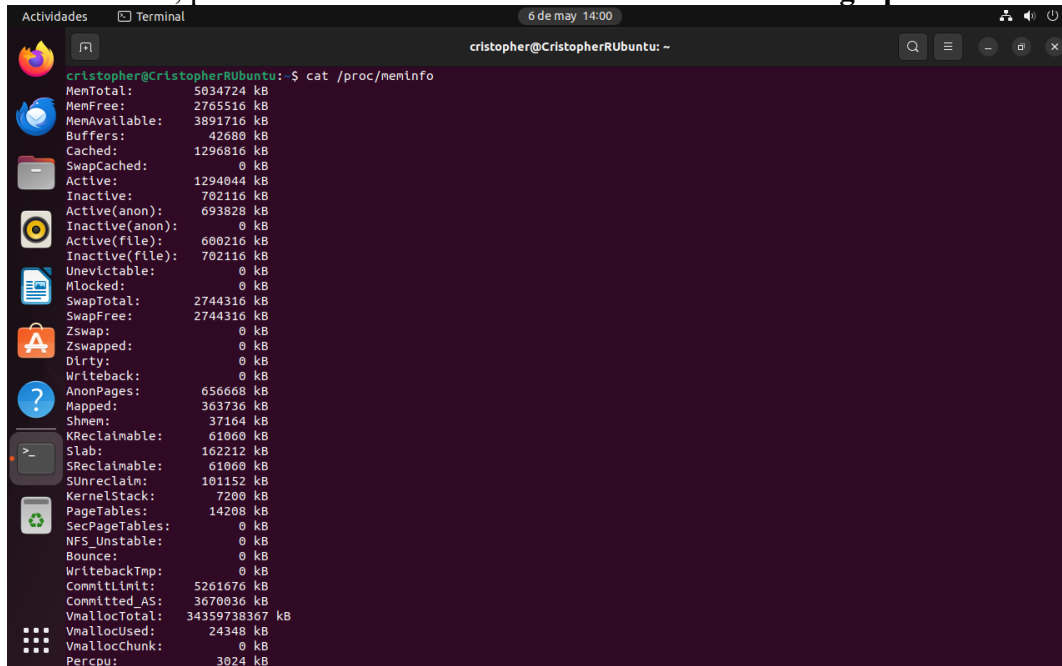
A screenshot of a Linux desktop environment with a terminal window open. The terminal shows the command `grep VmallocTotal /proc/meminfo` being executed, which outputs the total virtual memory size. The desktop background is a red and purple geometric pattern, and the left sidebar contains various application icons.

```
cristopher@CristopherRUbuntu: ~  
cristopher@CristopherRUbuntu:~$ grep VmallocTotal /proc/meminfo  
VmallocTotal: 34359738367 kB  
cristopher@CristopherRUbuntu:~$
```

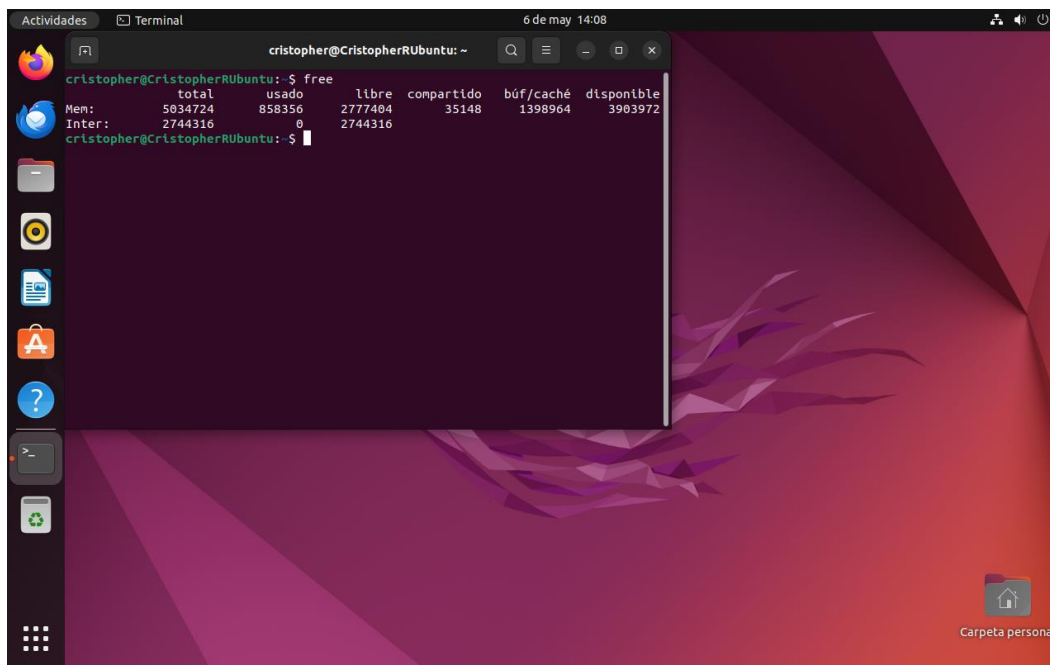
grep VmallocTotal nos dará la información de la memoria virtual del archivo.



Por otro lado, para saber la memoria física del archivo usaremos **grep MemTotal**

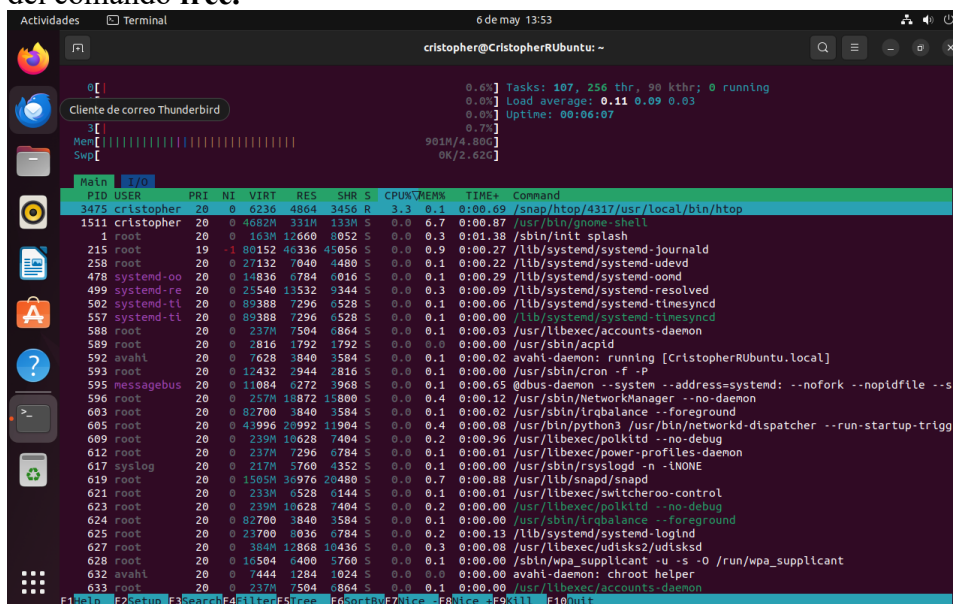


Para conocer toda la información relacionada a la memoria usaremos el comando **cat** con el archivo `/proc/meminfo`.



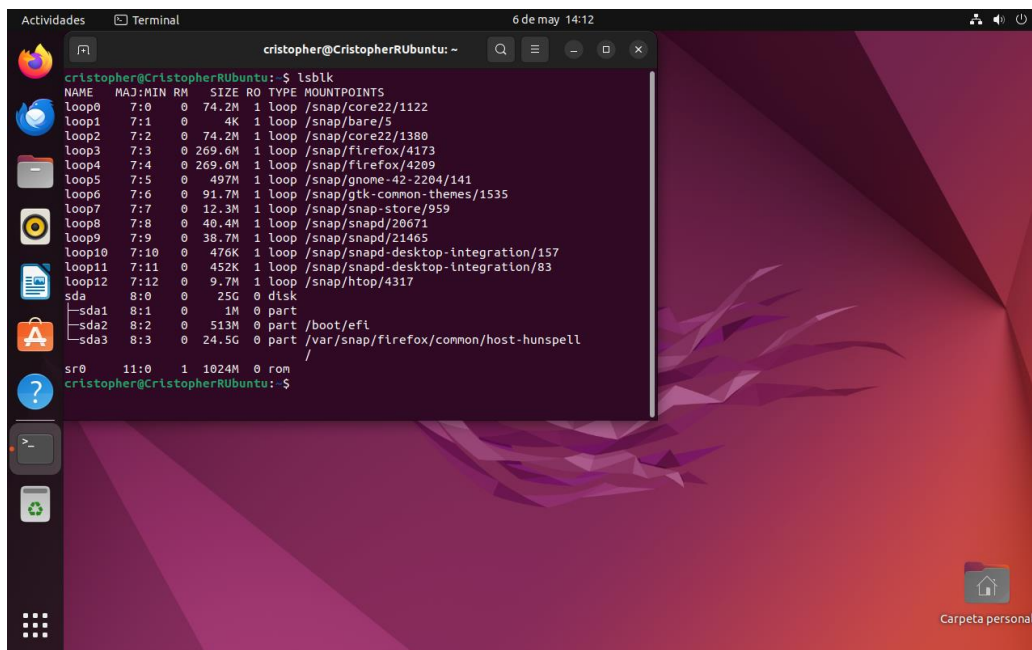
```
cristopher@CristopherRUbuntu: ~  
cristopher@CristopherRUbuntu:~$ free  
              total        usado      libre  compartido    búf/caché    disponible  
Mem:          5034724      858356      2777404      35148      1398964      3903972  
Inter:         2744316           0      2744316  
cristopher@CristopherRUbuntu:~$
```

Para conocer la saturación de la memoria (cuanto tenemos libre y también lo utilizado) dispondremos del comando **free**.



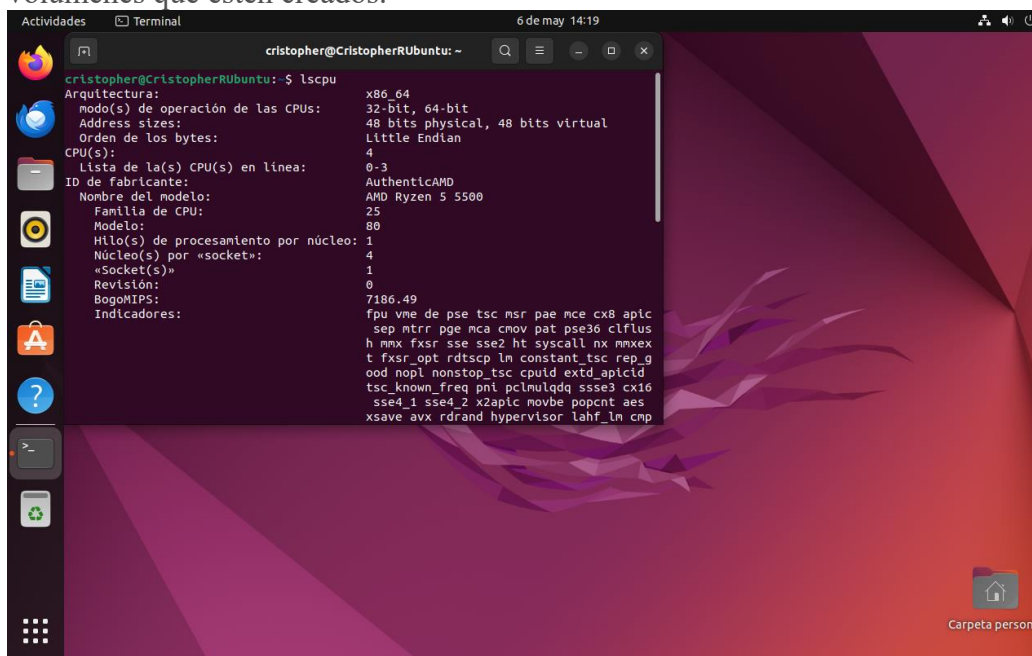
```
cristopher@CristopherRUbuntu: ~  
0.6% Tasks: 107, 256 thr, 90 kthr; 0 running  
0.0% Load average: 0.11 0.09 0.03  
0.0% Uptime: 00:00:07  
0.7%  
Mem[|||||] 901M/4.80G  
Swp[|||||] 0K/2.62G  
Main | T/O  
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command  
3475 cristopher 20 0 6236 4864 3456 R 3.3 0.1 0:00.69 /snap/htop/4317/usr/local/bin/htop  
1511 cristopher 20 0 4682M 331M 133M S 0.0 6.7 0:00.87 /usr/bin/gnome-shell  
1 root 20 0 163M 12660 8052 S 0.0 0.3 0:01.38 /sbin/init splash  
215 root 19 -1 80152 46396 45056 S 0.0 0.9 0:00.27 /lib/systemd/systemd-journald  
258 root 20 0 27132 7040 4480 S 0.0 0.1 0:00.22 /lib/systemd/systemd-udev  
478 systemd-oo 20 0 14836 6784 6016 S 0.0 0.1 0:00.29 /lib/systemd/systemd-oond  
499 systemd-re 20 0 25540 13532 9344 S 0.0 0.3 0:00.09 /lib/systemd/systemd-resolved  
502 systemd-tl 20 0 89388 7296 6528 S 0.0 0.1 0:00.06 /lib/systemd/systemd-timesyncd  
557 systemd-tl 20 0 89388 7296 6528 S 0.0 0.1 0:00.00 /lib/systemd/systemd-timesyncd  
580 root 20 0 237M 7504 6864 S 0.0 0.1 0:00.03 /usr/libexec/accounts-daemon  
589 root 20 0 2816 1792 1792 S 0.0 0.0 0:00.00 /usr/sbin/acpid  
592 avahi 20 0 7628 3840 3584 S 0.0 0.1 0:00.02 avahi-daemon: running [CristopherRUbuntu.local]  
593 root 20 0 12432 2944 2816 S 0.0 0.1 0:00.00 /usr/sbin/cron -f -P  
595 messagebus 20 0 11084 6272 3968 S 0.0 0.1 0:00.65 @dbus-daemon --system --address=systemd: --nofork --nopidfile --s  
596 root 20 0 257M 18872 15800 S 0.0 0.4 0:00.12 /usr/sbin/NetworkManager --no-daemon  
603 root 20 0 82700 3840 3584 S 0.0 0.1 0:00.02 /usr/sbin/irqbalance --foreground  
605 root 20 0 43996 20992 11984 S 0.0 0.4 0:00.08 /usr/bin/python3 /usr/bin/networkd-dispatcher --run-startup-trigg  
609 root 20 0 239M 16628 7404 S 0.0 0.2 0:00.96 /usr/libexec/polkitd --no-debug  
612 root 20 0 237M 7296 6784 S 0.0 0.1 0:00.01 /usr/libexec/power-profiles-daemon  
617 syslog 20 0 217M 5760 4352 S 0.0 0.1 0:00.00 /usr/sbin/rsyslogd -n -lNONE  
619 root 20 0 1505M 36976 20480 S 0.0 0.7 0:00.88 /usr/lib/snapd/snapd  
621 root 20 0 233M 6528 6144 S 0.0 0.1 0:00.01 /usr/libexec/switcheroo-control  
623 root 20 0 239M 16628 7404 S 0.0 0.2 0:00.00 /usr/libexec/polkitd --no-debug  
624 root 20 0 82700 3840 3584 S 0.0 0.1 0:00.00 /usr/sbin/irqbalance --foreground  
625 root 20 0 23700 8036 6784 S 0.0 0.2 0:00.13 /lib/systemd/systemd-logind  
627 root 20 0 384M 12868 10436 S 0.0 0.3 0:00.08 /usr/libexec/udisks2/udisksd  
628 root 20 0 16504 6400 5760 S 0.0 0.1 0:00.00 /sbin/wpa_supplicant -u -s -O /run/wpa_supplicant  
632 avahi 20 0 7444 1284 1024 S 0.0 0.0 0:00.00 avahi-daemon: chroot helper  
633 root 20 0 237M 7504 6864 S 0.0 0.1 0:00.00 /usr/libexec/accounts-daemon  
F1?oP F2Setup F3Search F4Filter F5Free F6SortBy F7Nice F8Nice F9Kill F10Quit
```

htop es el que usaremos como visor de procesos interactivos y poder gestionar los recursos.



```
crisopher@CristopherRUbuntu: ~  
$ lsblk  
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS  
loop0       7:0    0 74.2M 1 loop /snap/core22/1122  
loop1       7:1    0    4K 1 loop /snap/bare/5  
loop2       7:2    0 74.2M 1 loop /snap/core22/1380  
loop3       7:3    0 269.6M 1 loop /snap/firefox/4173  
loop4       7:4    0 269.6M 1 loop /snap/firefox/4209  
loop5       7:5    0 497M 1 loop /snap/gnome-42-2204/141  
loop6       7:6    0 91.7M 1 loop /snap/gtk-common-themes/1535  
loop7       7:7    0 12.3M 1 loop /snap/snap-store/959  
loop8       7:8    0 40.4M 1 loop /snap/snapd/20671  
loop9       7:9    0 38.7M 1 loop /snap/snapd/21465  
loop10      7:10   0 476K 1 loop /snap/snapd-desktop-integration/157  
loop11      7:11   0 452K 1 loop /snap/snapd-desktop-integration/83  
loop12      7:12   0   9.7M 1 loop /snap/htop/4317  
sda         8:0    0   25G 0 disk  
├─sda1      8:1    0    1M 0 part  
├─sda2      8:2    0 513M 0 part /boot/efi  
└─sda3      8:3    0 24.5G 0 part /var/snap/firefox/common/host-hunspell/  
sr0        11:0    1 1024M 0 rom  
crisopher@CristopherRUbuntu: $
```

El comando **lsblk** nos entrega detalles de los dispositivos de bloque permitiendo ver todos los dispositivos de bloque de Linux incluyendo los discos duros, las unidades flash, las particiones y los volúmenes que estén creados.



```
crisopher@CristopherRUbuntu: ~  
$ lscpu  
Arquitectura:          x86_64  
modo(s) de operación de las CPUs: 32-bit, 64-bit  
Address sizes:         48 bits physical, 48 bits virtual  
Orden de los bytes:    Little Endian  
CPU(s):                4  
Lista de la(s) CPU(s) en línea: 0-3  
ID de fabricante:      AuthenticAMD  
Nombre del modelo:     AMD Ryzen 5 5500  
Familia de CPU:        25  
Modelo:                80  
Hilo(s) de procesamiento por núcleo: 1  
Núcleo(s) por «socket»: 4  
«Socket(s)»:           1  
Revisión:              0  
BogoMIPS:              7186.49  
Indicadores:           fpu vme de pse tsc msr pae mce cx8 apic  
                        sep mtrr pge mca cmov pat pse36 clflush  
                        h mmx fxsr sse sse2 ht syscall nx mmxex  
                        t fxsr_opt rdtscp ln constant_tsc rep_g  
                        ood nopl nonstop_tsc cpuid extd_apicid  
                        tsc_known_freq pni pclmulqdq sse3 cx16  
                        sse4_1 sse4_2 x2apic movbe popcnt aes  
                        xsave avx rdrand hypervisor lahf_lnx cnp
```

El CPU es uno de los pilares de cualquier sistema y gracias al comando **lscpu** tendremos detalles acerca de la arquitectura del procesador usado en Linux.

```
cristopher@CristopherRUbuntu:~$ lshw
AVISO: debería ejecutar este programa como superusuario.
cristopherubuntu
descripción: Computer
anchura: 64 bits
capacidades: smp vsyscall32
*-core
descripción: Motherboard
id físico: 0
*-memory
descripción: Memoria de sistema
id físico: 0
tamaño: 5248MiB
*-cpu
producto: AMD Ryzen 5 5500
fabricante: Advanced Micro Devices [AMD]
id físico: 1
información del bus: cpu@0
versión: 25.80.0
anchura: 64 bits
capacidades: fpu_exception wp vme de pse tsc msr pae mce cx8 apic
sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx mmxext f
xsr opt rdtscp x86-64 constant_tsc rep_good nopl nonstop_tsc cpuid extd_apicid t
sc_known_freq pni pclmulqdq sse3 cx16 sse4_1 sse4_2 x2apic movbe popcnt aes xsa
```

El comando **lshw** nos entrega detalles del hardware del equipo con todas sus especificaciones, este resultado no solo puede verse en pantalla, sino que podremos exportarlo en HTML, XML, JSON y

```
top - 13:52:07 up 4 min, 1 user, load average: 0.02, 0.06, 0.02
Tareas: 208 total, 2 ejecutar, 206 hibernar, 0 detener, 0 zombie
%Cpu(s): 1.5 us, 2.0 sy, 0.0 ni, 96.5 id, 0.0 wa, 0.0 hi, 0.0 st, 0.0 st
MiB Mem : 4916.7 total, 2769.7 libre, 831.3 usado, 1315.7 búfer/cache
MiB Intercambio: 2680.0 total, 2680.0 libre, 0.0 usado, 3821.5 dispon Mem

PID USUARIO PR NI VIRT RES SHR S %CPU %MEM HORA+ ORDEN
1482 cristop+ 20 0 4795048 334280 133152 S 11.5 6.6 0:04.42 gnome-shell
3025 cristop+ 20 0 558548 53744 41856 R 3.8 1.1 0:00.31 gnome-terminal-
1 root 20 0 166712 11508 8052 S 0.0 0.2 0:01.12 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
7 root 20 0 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0-events
8 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0H-events_highpri
9 root 20 0 0 0 0 I 0.0 0.0 0:00.00 kworker/0:1-rcu_gp
10 root 20 0 0 0 0 I 0.0 0.0 0:00.35 kworker/u8:0-events_power_efficient
11 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 mm_percpu_wq
12 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_kthread
13 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_rude_kthread
14 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_tasks_trace_kthread
```

más.

Para finalizar, el comando **top**, nos dará información en tiempo real acerca de los procesos la cual será en constante cambio, esto debido a que se muestra el valor en tiempo real. [OBJ]

Conclusión

Entender los comandos para el manejo del hardware en Ubuntu es crucial por diversas razones que impactan positivamente en el entorno laboral como en la vida diaria:

1. Manejo de la productividad: La capacidad de diagnosticar y comprender el estado del hardware permite una gestión más eficaz del tiempo, evitando demoras innecesarias causadas por el

desconocimiento técnico.

2. Toma de decisiones informadas: Al conocer las capacidades exactas del hardware, los usuarios pueden tomar decisiones informadas sobre las actualizaciones, mejoras o adquisiciones de nuevos equipos que se alineen a sus necesidades reales.

3. Mantenimiento proactivo: Usar los comandos para monitorear el hardware puede prevenir fallos en el sistema al detectar anticipadamente potenciales problemas.

4. Empoderamiento y confianza: La habilidad para manejar el hardware a través de comandos fomenta un sentido de empoderamiento y confianza en las capacidades propias, lo cual es esencial en un mundo tecnológico en constante cambio.

En conclusión, la competencia en el uso de comandos para el hardware en Ubuntu es una herramienta valiosa que no solo optimiza el rendimiento y la seguridad del sistema, sino que también fortalece la autonomía y la confianza del usuario en su interacción diaria con la tecnología.

Referencias

Comandos de información del sistema o hardware Linux. (s. f.). [Vídeo]. Solvetic.

https://www.solvetic.com/tutoriales/article/12933-comandos-de-informacion-del-sistema-o-hardware-linux/#google_vignette

Díaz, D. (2023, 16 junio). *Los 40 Comandos de Linux Más Utilizados que Debes Conocer*. Kinsta®.

<https://kinsta.com/es/blog/linux-comandos/>