Grupo de Investigación Salud Pública

Introducción a Linux para Bioinformática

Manual de Bioinformática

Héctor Fabio Espitia-Navarro

Universidad de Nariño
FUNDADA EN 1904

Universidad de Nariño
ACREDITADA DE ALTA CALIDAD
RESOLUCIÓN MEN 10567 - MAYO 23 DE 2017

CESUN
CENTRO DE ESTUDIOS EN SALUD
Universidad de Nariño

Universidad de **Nariño**
FUNDADA EN 1904

Universidad de **Nariño**
ACREDITADA DE ALTA CALIDAD
RESOLUCIÓN MEN 10567 - MAYO 23 DE 2017

CESUN
CENTRO DE ESTUDIOS EN SALUD
Universidad de **Nariño**

INTRODUCCIÓN A LINUX PARA BIOINFORMÁTICA

Manual de Bioinformática

INTRODUCCIÓN A LINUX PARA BIOINFORMÁTICA

Manual de Bioinformática

Héctor Fabio Espitia-Navarro

Universidad de Nariño
FUNDADA EN 1904

Universidad de Nariño
ACREDITADA DE ALTA CALIDAD
RESOLUCIÓN MEN 10567 – MAYO 23 DE 2017

CESUN
CENTRO DE ESTUDIOS EN SALUD
Universidad de Nariño

Introducción a Linux para Bioinformática: Manual de Bioinformática

Diagramación y diseño: Héctor Fabio Espitia-Navarro

Fecha de publicación: 2023
San Juan de Pasto, Nariño, Colombia

Tabla de contenidos

Bienvenid@

Este es “Introducción a Linux para Bioinformática”, un manual creado con la finalidad de proveer conceptos básicos del sistema operativo (SO) Linux y su uso, orientados a la ejecución de tareas de bioinformática. El manual cubre las tareas más básicas de Linux comúnmente utilizadas en análisis básicos de bioinformática, como el uso de la línea de comandos para la manipulación de archivos, directorios y texto.

Por qué leer este manual

Aunque existen herramientas gráficas para bioinformática, el uso de sistemas operativos parecidos a Unix, tal como Mac o Linux, es el estándar actual para practicar la bioinformática. La mayoría de herramientas bioinformáticas se han desarrollado como programas basados en línea de comandos que se ejecutan en SOs como Linux. Así, la utilización de estas herramientas depende de las habilidades del usuario en la línea de comandos.

Este manual proveerá los conceptos y habilidades básicas para sacar provecho de Linux como una plataforma poderosa para el análisis y la administración de datos biológicos. Al terminar el manual, el lector podrá instalar Linux, estará familiarizado con el entorno de trabajo de línea de comandos, podrá manipular archivos, directorios y textos, y realizará comandos útiles para tareas básicas de bioinformática.

Prerrequisitos

EL manual está destinado para ser ejecutado en el sistema Operativo (SO) Linux. Específicamente, se usó la distribución Ubuntu Linux 20.04 para desarrollar este manual. En el Capítulo 2 econtrará las instrucciones para instalar Linux.

Convenciones del manual

Tipografía

Las siguientes convenciones tipográficas son usadas en este manual:

- *Cursivas*. Se usan para indicar:
  - Palabras escritas en idiomas diferentes del español (p. ej. *prompt*, o *forward*)
  - Expresiones aritméticas (p. ej. $Q \geq 20$).
- **Negrillas**. Se usan para representar:
  - Teclas (p. ej. **ENTER**)
  - Conceptos que se definen por primera vez
- `Fuente monoespaciada`. Se usan en:
  - Nombres de archivos y directorios (p. ej. `/ruta/al/archivo.txt`, o `Documentos`)
  - Nombres especiales (p. ej. ambientes conda: `base`)
  - Nombre de comandos (p. ej. `ls`).
  - Bloques de órdenes de línea de comandos y su salida. Por ejemplo:

```
ls -l /home
```

```
total 4
drwxr-xr-x 39 hector hector 4096 mar 28 15:59 hector
```

- Hipervínculos (links): enlaces a partes de este documento (internos) o a otros sitios Web (externos).

Bloques especiales

Textos con información de especial atención se presentan en bloques (recuadros) especiales que se diferencian por su color. Existen cinco tipos de bloques especiales:

Este es un ejemplo de una nota.

Importante

Este es un texto que corresponde a una información importante.

Tip

Así se ven los consejos o trucos

Precaución

Este bloque denota una información de precaución

Advertencia

Este bloque denota una información de advertencia.

1. Bioinformática y Linux

1.1. Qué es bioinformática

La bioinformática puede ser entendida como un disciplina que combina técnicas de múltiples campos de conocimiento para interpretar datos biológicos. Campos como la biología, ciencias de la computación, matemáticas y estadística son usados en la bioinformática para analizar datos biológicos, mayoritariamente secuencias de ADN o proteínas, y extraer conocimiento de ellos.

Aunque la bioinformática ha estado presente durante los últimos 60 años, se desarrolló fuertemente durante los años 80, y debido a los grandes avances tecnológicos en computación y secuenciación de ADN, la bioinformática ha tenido un creciente auge y demanda durante los últimos 15 años aproximadamente. Actualmente las nuevas tecnologías de secuenciación producen grandes volúmenes de datos, los cuales solo pueden ser analizados eficientemente a través de los computadores, aplicando técnicas de los campos del conocimiento mencionados.

Las ciencias ómicas, como la genómica (estudio de los genomas), la transcriptómica (estudio de los transcriptomas), y la proteómica (estudio de las proteínas), han tenido un gran avance gracias a la bioinformática. Las ómicas han aportado entendimiento de los procesos biológicos involucrados en áreas de interés científico, como enfermedades, medicina personalizada, o mejoramiento de cultivos.

1.2. Qué es Linux

Se puede definir Linux como una familia de sistemas operativos (SOs) de distribución libre y código abierto (*open-source*[1]) basados en el núcleo o *kernel* del mismo nombre. Linux, es tal vez el SO *open-source* más conocido y usado actualmente en mundo. Linux es usado en diversos dispositivos y para variadas aplicaciones, desde teléfonos celulares, automóviles,

[1]Wikipedia - Software libre y de código abierto: https://es.wikipedia.org/wiki/Software_libre_y_de_c%C3%B3digo_abierto

refrigeradores, televisores, hasta computadores de escritorio, la mayoría de los servidores que soportan la Internet, y los 500 más poderosos supercomputadores del mundo[2].

Dada su fiabilidad y flexibilidad, Linux ha sido adoptado por la comunidad científica al rededor del mundo. Campos como la física de partículas, la astronomía, y las ciencias biomédicas usan Linux como parte de sus flujos de trabajo de investigación y desarrollo. La bioinformática no es la excepción en cuanto al uso de Linux en ambientes científicos, pues actualmente se desarrolla casi exclusivamente en Linux.

1.3. Por qué usar Linux para Bionformática

- Es estable y rápido, apto tanto para computadores de escritorio y portátiles, como para servidores y ambientes distribuidos.
- Es libre: no cuesta nada y puede ser modificado seg según las necesidades del usuario.
- Es POSIX (del inglés *Portable Operating System Interface*) lo cual asegura compatibilidad e interoperablidad en diferentes sustemas de cómputo.
- La mayoría de software especializado en bioinformática está disponible en Linux.
- Trae listos múltiples programas para procesar archivos.
- Se ha convertido en el SO estándar *de facto* para Bioinformática.
- Fácil creación de tuberías o *pipelines* para análisis. Aquí la salida de un programa es la entrada del siguiente, por lo que se dice que son ensamblados como tuberías.

[2] *Top 500 - Operating system Family / Linux* (estadísticas hasta 2022): https://www.top500.org/statistics/details/osfam/1/

2. Linux

Para ser exactos Linux no es un sistema operativo (SO) *per se*, es un núcleo o *kernel* usado en muchos SOs. Sin embargo, de manera genérica los SOs que usan este *kernel* toman el nombre Linux. Un *kernel* es la base del SO que tiene control absoluto sobre todos los dispositivos o *hardware* de un computador (memoria, procesador, disco, etc.).

El *kernel* Linux fue creado por Linus Torvalds como una alternativa del SO Unix. Después de ser liberado en septiembre de 1991, el *kernel* Linux se convirtió rápidamente en la base del SO GNU, un proyecto que buscaba reemplazar al SO Unix. Dado que Linux tiene características parecidas al SO Unix, se considera como un sistema "similar a Unix" (Unix-*like*), también conocido como UN*X o *nix. Los SOs (o distribuciones) Linux empaquetan el *kernel* junto con otros componentes como bibliotecas, el sistema gráfico y de ventanas, y programas (editores de texto, ofimática, administrador de archivos, etc.). Se estima que actualmente existen más de 300 distribuciones Linux mantenidas activamente (SUSE 2021), entre las que se encuentran Ubuntu, Debian, Fedora, y Mint como algunas de las más populares.

2.1. Instalación entorno Linux

2.2. Instalación nativa

El entorno ideal Linux es aquel en donde se ejecuta de manera nativa en la máquina. La gran ventaja de ejecutar un SO nativamente es que tiene control exclusivo de todos los recursos del computador (memoria RAM, almacenamiento en disco, y núcleos de procesamiento), por lo que puede explotar al máximo las características de la máquina. La desventaja es que el usuario requiere tener destrezas informáticas medias o avanzadas para su instalación. Esto es válido para cualquier SO, bien sea Linux, Windows, o macOS, por ejemplo. Linux puede ser instalado como único SO o junto con otros SOs (por ejemplo Windows) en la máquina, pero en todo caso, solo un SO puede ser ejecutado a la vez.

En aras del tiempo y la practicidad, no se tratará aquí la instalación de Linux de manera nativa. Sin embargo, en caso de que el lector quiera explorar esta opción, se recomienda

leer la guía oficial de Ubuntu Linux "*Install Ubuntu desktop*" en inglés, o la guía "Como instalar Ubuntu 20.04 (u otra distro Linux) en tu equipo".

2.2.1. Máquina virtual

Una opción para instalar Linux es usar una máquina virtual. Una máquina virtual es una instancia simulada de un computador con todos sus recursos. Para nuestros propósitos y suponiendo que usted cuenta con un computador con Windows, la instancia simulada o máquina virtual será Linux. Esta máquina virtual se ejecutará en su equipo físico con Windows a través de un programa de virtualización, que en este caso será VirtualBox[1]. En el Apéndice A se describe todo el proceso paso a paso para instalar Linux como una máquina virtual usando VirtualBox.

2.2.2. Windows WSL

Otra opción práctica es instalar el Subsistema de Windows para Linux (WSL del inglés *Windows Subsystem for Linux*) si se cuenta con una máquina con Windows 10/11. El WSL es una característica de Windows 10/11 que permite ejecutar programas de Linux en Windows de manera nativa (Microsoft 2021). Aunque esta instalación es práctica, puede dar problemas en algunos usos específicos. Sin embargo, para el objetivo de este manual la instalación de Windows WSL es perfectamente válida y funcional. Puede instalar Linux con WSL siguiendo el artículo "Instalación de Linux en Windows con WSL".

2.3. La línea de comandos

La línea de comandos es una interfaz de solo texto que permite la comunicación del usuario con el computador. La comunicación se realiza a través de la entrada de órdenes llamadas "comandos", las cuales son interpretadas por el *shell*. El *shell* es un programa que actúa como interfaz entre el usuario y el *kernel*, esto es, recibe los comandos, los pasa al *kernel*, recibe la respuesta del *kernel*, y finalmente la pasa al usuario.

La línea de comandos es la principal herramienta utilizada para hacer bioinformática en Linux, y se accede mediante el "emulador de la terminal" o simplemente "terminal". La terminal es un programa que permite la entrada de comandos en un ambiente gráfico (una ventana), y su apariencia y características dependen del SO. En este manual se usará Ubuntu Linux 20.04 para todos los ejemplos presentados (ver Apéndice A para instalar Ubuntu).

[1] VirtualBox, sitio Web: https://www.virtualbox.org

Para abrir la terminal en Ubuntu basta con hacer clic en el botón Aplicaciones en el *dash* (barra lateral), y a continuación escribir "terminal" en la barra de búsqueda. La Figura 2.1 y Figura 2.2 muestran los pasos para abrir la terminal.

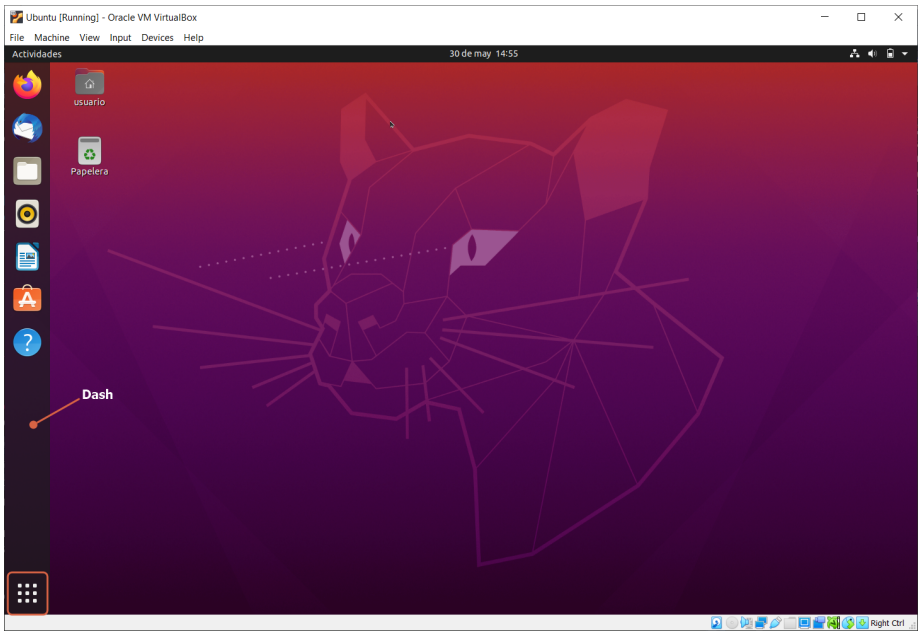


Figura 2.1.: Escritorio de Ubuntu Linux 20.04. El botón Aplicaciones (recuadro rojo), ubicado en la parte inferior del *dash* (barra lateral) es parecido al botón Inicio de Windows, y permite buscar e iniciar programas.

Al hacer clic en el ícono de la terminal, la ventana de la terminal se inicia (Figura 2.3).

Lo primero que verá en la línea de comandos de la terminal será una línea llamada el *command prompt* (también, solo *prompt*) o símbolo del sistema que luce parecido a esto:

```
hector@Ubuntu:~$
```

En el ejemplo anterior del símbolo del sistema, se pueden identificar los siguientes elementos:

- `hector`: el nombre del usuario actual
- `Ubuntu`: el nombre de la máquina
- `~`: El directorio actual. En `bash`, el *shell* o intérprete de comandos por defecto, la virgulilla (`~`), es un carácter especial que equivale (o "expande") a la ubicación del

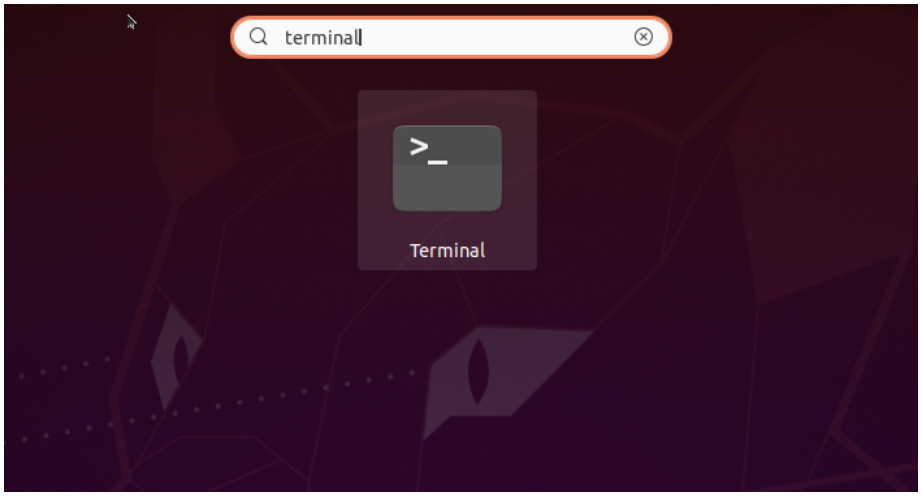


Figura 2.2.: Barra de búsqueda Ubuntu Linux 20.04. La barra de búsqueda permite buscar cualquier programa instalado en Ubuntu. En este caso se muestra el resultado después de buscar la terminal.

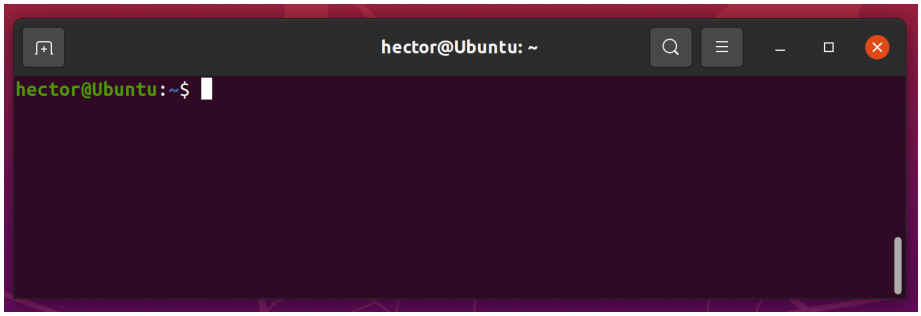


Figura 2.3.: La terminal de Ubuntu Linux 20.04.

directorio *home*, en este caso `/home/hector`[2].

En adelante, se asume que todos los comandos descritos en este texto serán ejecutados a través de la terminal de Ubuntu.

2.4. Los comandos

Los comandos son programas que realizan tareas específicas en el sistema. Así, existe un comando para cada acción que el usuario desea ejecutar en el SO: uno para listar los archivos en un directorio (`ls`) , uno para crear directorios (`mkdir`), uno para copiar archivos (`cp`), etc.

Cada comando se ejecuta en la terminal invocando el nombre respectivo, escrito todo en minúsculas, y a continuación presionando al tecla **Enter**. Por ejemplo, para conocer el directorio actual, digite el comando `pwd` y a continuación presione **Enter**. El resultado del comando `pwd` muy seguramente será algo como `/home/NOMBRE_DE_USUARIO`, en donde `NOMBRE_DE_USUARIO` corresponderá al usuario actual que está usando la terminal (Figura 2.4).

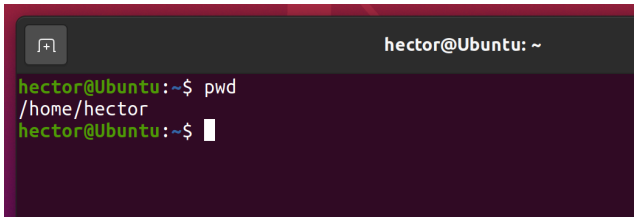


Figura 2.4.: Ejecución del comando `pwd`.

2.4.1. Anatomía de los comandos

Los comandos se componen de tres partes: el nombre, las opciones, y los argumentos:

```
nombre_comando [OPCIONES] ARG_1 ... ARG_N
```

La linea anterior es una representación de la estructura de los comandos. Como se dijo, el nombre (`nombre_comando`) sirve para invocar el comando y siempre será estará ubicado en primer lugar.

[2]Para saber más acerca del directorio *home* consulte la Sección 3.1.1.

> **Nota**
>
> En la estructura o sintaxis de los comandos, los corchetes `[]` significan que los argumentos son opcionales, por eso las opciones se presentaron como `[OPCIONES]`. Algunas veces, puede ver los argumentos obligatorios entre símbolos de mayor y menor que: `<ARG_1>`

Las opciones (`[OPCIONES]`) se pueden agregar (o no, pues son opcionales) para modificar el comportamiento de un comando. Las opciones consisten en palabras (una o más letras) que se escriben siempre después del nombre del comando y que van precedidas de uno o dos guiones (`-` o `--`), p. ej. `comando -o` o `comando --opcion`. En algunos casos, las opciones requieren valores, p. ej. `comando --opcion valor`.

Los argumentos (`ARG_1 ... ARG_N`) son valores (uno o más) que se pasan al comando y que, en algunos casos, son requeridos (obligatorios) para que el comando funcione.

2.4.2. Ayuda de los comandos

Un aspecto clave para el uso de los comando es la ayuda de cada uno. Aunque existen decenas de comandos en Linux y su uso depende en gran medida de la buena memoria del usuario para recordarlos, no es necesario recordar cada detalle acerca de los comandos. Solo basta con recordar el nombre del comando y siempre se podrá consultar su ayuda abreviada o extendida.

2.4.2.1. La ayuda

La ayuda es un mensaje que provee la información más básica a cerca del uso de un comando, por lo que se le conoce como ayuda corta o abreviada. El mensaje de ayuda generalmente describe la estructura o manera de usarlo (*usage*), la función que realiza el comando, y las opciones y argumentos disponibles.

En la mayoría de los casos la ayuda se consulta agregando la opción `--help` al comando. Por ejemplo, para consultar la ayuda del comando `mkdir`, basta con ejecutar:

```
mkdir --help
```

```
Modo de empleo: mkdir [OPCIÓN]... DIRECTORIO...
Crea los DIRECTORIO(s), si no existen ya.
```

```
Los argumentos obligatorios para las opciones largas son también
 ↪  obligatorios
para las opciones cortas.
  -m, --mode=MODE   establece los permisos (como en chmod), en lugar
                    de  a=rwx - umask
  -p, --parents     no hay error si existen, crea los directorios padres
   ↪  en
                    caso necesario
  -v, --verbose     muestra un mensaje por cada directorio creado
  -Z                  establece el contexto de seguridad SELinux de cada
                        directorio creado al tipo predeterminado
      --context[=CTX]  como -Z, o si se especifica CTX entonces
       ↪  establece el
                        contexto de seguridad SELinux o SMACK a CTX
      --help     muestra esta ayuda y finaliza
      --version  informa de la versión y finaliza

ayuda en línea sobre GNU coreutils:
 ↪  <https://www.gnu.org/software/coreutils/>
Report mkdir translation bugs to <https://translationproject.org/team/>
Full documentation at: <https://www.gnu.org/software/coreutils/mkdir>
or available locally via: info '(coreutils) mkdir invocation'
```

> **Advertencia**
>
> Si usa `--help` con el comando `pwd` en Ubuntu, tal vez obtendrá un error. Esto se debe a que Ubuntu usa por defecto una versión distinta del comando `pwd`, la cual no acepta esta opción. Para que funcione correctamente debe ejecutar `/usr/bin/pwd --help`.

2.4.2.2. El manual

Cuando se requiere ver todas las opciones y argumentos de un comando junto con información detallada de las características, uso, y ejemplos, se puede consultar el manual o ayuda extendida, siempre que esté disponible. El manual se consulta mediante el comando `man` seguido del nombre del comando de cual se desea conocer más información. Por ejemplo, para consultar el manual del comando `mkdir`, se ejecuta el la orden:

```
man mkdir
```

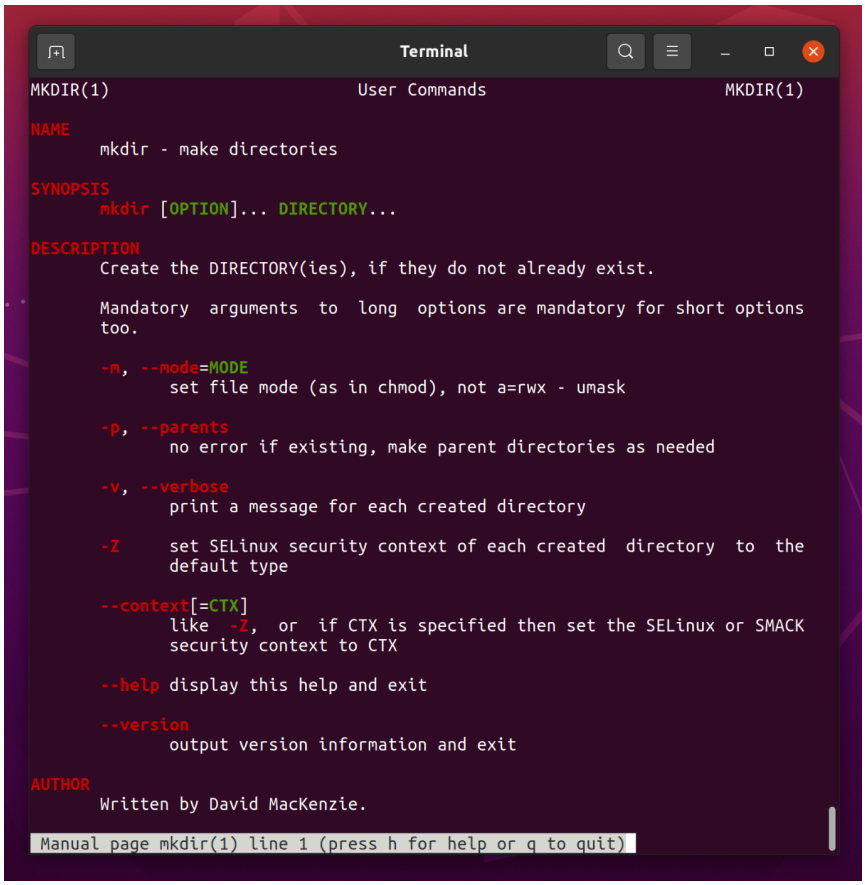


Figura 2.5.: Manual del comando `mkdir` en la terminal, generado con la orden `man mkdir`.

El manual es desplegado en un modo especial que permite navegar a través del contenido usando las teclas de dirección (↑/↓) o de avance página (**RePág**/**AvPág** o **PageUp**/**PageDown** en inglés). Como verá, el manual es mucho más extenso comparado con el mensaje de ayuda del comando (Figura 2.5).

Para salir de este modo y regresar a la línea de comandos se presiona la tecla **Q**.

2.5. Lecturas adicionales

- *What is Linux and Why There are 100's of Linux Distributions?*

- *Wikipedia - Linux*
- *Why do Bioinformaticians Avoid Using Windows?*
- *Install WSL*
- *An Introduction to the Linux Terminal*

3. Administración de archivos y directorios

3.1. Sistema de archivos de Linux

El sistema de archivos de Linux se organiza de manera jerárquica, en donde todos los directorios y archivos descienden de un elemento común llamado directorio raíz (*root* en inglés) el cual se representa con el carácter de barra inclinada `/` (*slash* en inglés). En la Figura 3.1 Se puede observar la estructura del sistema de archivos de Linux. Observe cómo todos los elementos se desprenden del directorio raíz (`/`), es decir que están contenidos en él.

Como se puede observar en la Figura 3.1, hay directorios que aparecen con una barra inclinada en medio, p. ej. `usr/bin`. Esta estructura es llamada ruta o *path*, y describe la ruta o dirección hasta la ubicación de un elemento del sistema de archivos del SO. En el caso anterior, podemos decir que `bin` se encuentra dentro de `usr`. Cada elemento se separa por medio de una barra inclinada (`/`), de manera parecida a las rutas en Windows que utilizan barras invertidas (`\`, p. ej. `C:\Users\MiUsurio`). En la Tabla 3.1 se encuentran descritos los directorios más relevantes de Linux.

Tabla 3.1.: Directorios m'as importantes del sistema de archivos de Linux.

Directorio	Descripción
`bin`	Continene archivos ejecutables, también llamados binarios (binaries), esenciales del SO.
`boot`	Contiene archivos necesarios para el arranque (boot) del SO.
`dev`	Este directorio contiene archivos de los dispositivos de la máquina en uso por el SO, icluidos los discos internos o externos montados.
`home`	Contiene los directorios de los usuarios del SO. Más información en la Sección 3.1.1.
`lib`	Este folder contiene archivos de bibliotecas (libraries) requeridos para el funcionamiento de distintos programas.
`mnt`	Este directorio es usado para montar temporalmente dispositivos de almacenamiento como discos duros externos.

Directorio	Descripción
`opt`	Aquí se almacenan arhivos de programas opcionales (optional), los cuales no vienen incluidos con la distribución Linux y son instalados independientemente por los usuarios. Aquí se instalan por ejemplo programas como Google Chrome.
`root`	Existe un usuario diferente de todos los demás, el root. Este usuario tiene control absoluto sobre los procesos y archivos del SO, por eso se puede decir que tiene "superpoderes". Este directorio es su home, y como puede verse, no está dentro del directorio /home como el resto de los usuarios.
`sbin`	Como en el caso del directorio /bin, contiene archivos binarios pero destinados a ser usados por el usuario root.
`tmp`	Aquí se almacenan archivos temporales usados por el SO y programas. El contenido de este archivo se elimina cada vez que el SE se reinicia.
`usr`	Este directorio es de sólo lectura y contiene aplicaciones y archivos que son usados y compartidos por todos los usuarios del sistema.
`var`	Parecido a /usr pero se pude escribir en él. Contiene archivos de historial del sistema (logs).

> **Nota**
>
> En Linux todo elemento en el sistema de archivos es un archivo, incluso los directorios. Así, cada archivo puede ser de tres tipos: regular, directorio, o especial (enlaces simbólicos, de bloque, entre otros).

3.1.1. El directorio personal *home*

Cada usuario tiene un directorio personal el cual se denomina de manera genérica como *home*. El directorio *home* de cada usuario siempre está ubicado dentro del directorio del sistema `/home`. El nombre del directorio *home* de un usuario siempre corresponde a su nombre de usuario. Por ejemplo, mi nombre de usuario es hector, así entonces mi directorio *home* lleva el nombre `hector`. El directorio *home* contiene todos los archivos de un usuario y solo es accesible por dicho usuario. Cuando un usuario abre una terminal, el directorio actual de inicio es su directorio *home*. Como se dijo en la Sección 2.3, el directorio *home* se representa en la línea de comandos con con el carácter `~`. (virgulilla).

Figura 3.1.: Estructura del sistema de archivos de Linux generado con el comando `tree`. La opción `-L 1` hace que `tree` muestre los directorios y archivos del nivel 1 contenidos en el directorio raíz (`/`). Observe los elementos con el símbolo `->`. Estos elementos son enlaces simbólicos, algo parecido a los enlaces directos en Windows. En los enlaces simbólicos de la figura, el nombre (parte izquierda) está enlazado o apunta a una ubicación destino (parte derecha). Por ejemplo, en `bin -> usr/bin`, `bin` es un enlace (o apunta) a la ubicación `usr/bin`. En la figura, cada elemento tiene un color determinado según su tipo: directorios en azul, enlaces simbólicos en cyan (aguamarina), y archivos en blanco.

3.1.2. Rutas absolutas y relativas

En Linux, frecuentemente se requiere especificar la ubicación de archivos o directorios, por ejemplo para pasar un archivo a un comando para que sea procesado, o simplemente para navegar por distintos directorios desde la terminal. La ubicación de un archivo o directorio en Linux se especifica por medio de una ruta (*path*), la cual puede ser entendida como la "dirección" donde reside dicho archivo o directorio. Los elementos de una ruta que describen las ubicaciones, siempre están separados por barras inclinadas `/`. Una ruta puede ser de dos tipos: absoluta o relativa.

Una **ruta absoluta** es aquella que especifica la ubicación desde la raíz del sistema de archivos, es decir, el directorio *root* representado con la barra inclinada (`/`). Por ejemplo, en mi directorio *home* existe otro directorio llamado "Documentos". Dentro de este último directorio tengo un documento de texto llamado `mi_texto.txt`. La ruta absoluta a este archivo será entonces `/home/hector/Documentos/mi_texto.txt`. Observe que la ruta describe todos los directorios desde la raíz `/` hasta llegar al archivo en cuestión.

Una **ruta relativa** es aquella que especifica la ubicación de un archivo o directorio, con base en la ubicación actual ("relativa" a la ubicación actual). Siguiendo con el ejemplo anterior del archivo `mi_texto.txt`, suponga que acaba de abrir la terminal y por tanto se encuentra ubicado en su directorio *home*. La ruta relativa de este archivo será `Documentos/mi_texto.txt`. Observe que la ruta describe la ubicación partiendo de su ubicación actual (*home*), ignorando los directorios padre que lo contienen (`/home/hector`). Suponga ahora que está ubicado dentro del directorio `Documentos`; la ruta relativa del archivo en este caso sería `mi_texto.txt`, es decir, el nombre del archivo. Siguiendo la regla de la ruta relativa, se especifica la ubicación del archivo con base en la ubicación actual e ignorando los directorios que lo contienen (`/home/hector/Documentos`).

Para entender las rutas absolutas y relativas, se puede pensar en una analogía para ubicar objetos dentro de su casa. Suponga que desea especificar la ubicación de un bolígrafo que se encuentra sobre su escritorio. La ruta absoluta para ubicar el bolígrafo requiere que se especifique toda la dirección empezando desde la ciudad (ciudad, barrio, intersección de calle y carrera (o similar), número de la casa, habitación, escritorio, bolígrafo): `/cali/san_fernando/calle4b_cra36/casa00/hab01/escritorio_cafe/boligrafo_azul`. Ahora suponga que usted se encuentra en su casa; en este caso la ruta relativa al bolígrafo sería `hab01/escritotio_cafe/boligrafo_azul`, es decir, para ubicar el bolígrafo, no es necesario especificar la ciudad, barrio, intersección, y casa. Si usted estuviese sentado al frente de su escritorio, la ruta relativa sería aún más corta: `boligrafo_azul`.

3.2. Comandos

Ahora se explicarán los comando más comunes para la gestión de archivos y directorios en Linux. La siguiente tabla muestra el Uso de los comando que serán explicados. Note que hay tres columnas: el comando, nemotecnia, y la descripción. En la columna nemotecnia se presenta la regla para recordar más fácilmente el nombre del comando, que generalmente es una abreviación en inglés de la acción/tarea que realiza dicho comando.

Tabla 3.2.: Comandos para adminstración de archivos y directorios.

Comando	Nemotecnia	Descripción
`pwd`	***p**rint **w**orking **d**irectory*	Muestra el directorio de trabajo
`ls`	***l**i**s**t*	Lista (muestra) los archivos de un directorio
`cd`	***c**hange **d**irectory*	Cambia la ubicación actual de trabajo a otro directorio
`touch`	tocar	Crea un archivo vacío (sin contenido). Realmente modifica la fecha y hora de modificación (timestamp) de un archivo, pero usualmente se usa con el proósito descrito al principio.
`mkdir`	***m**a**k**e **dir**ectory*	Crea ("hace o fabrica" del inglés *make*) un directorio
`cp`	***c**o**p**y*	Copia archivos o directorios
`mv`	***m**o**v**e*	Mueve archivos o directorios de un logar a otro. También se usa para renombrar.
`cat`	*con**cat**enate*	Muestra el contenido de uno o varios archivos. También se usa para concatenar (pegar) varios arcivos en uno solo.

> **!** Importante
>
> A partir de aquí se espera que usted ejecute los comandos en su terminal. La práctica es fundamental para aprender los comandos y sentirse cómodo con el uso de la línea de comandos.

3.2.1. pwd

> **Uso**

```
pwd
```

Imprime el directorio de trabajo, es decir la ubicación actual en la línea de comandos.

Como se vio en el capítulo anterior, recién se inicia la terminal, el directorio de trabajo es el *home* (directorio personal) del usuario:

```
pwd
```

```
/home/hector
```

Observe que el resultado de `pwd` es una ruta absoluta, pues empieza desde el directorio raíz (`/`)

3.2.2. ls

Uso

```
ls [OPCIONES] [DIRECTORIO]
```

Lista el contenido del(de los) directorio(s) especificado(s) mediante `[DIRECTORIO]`. Opciones útiles:

- `-l`: Formato largo (*long*)
- `-a`: Lista todos (*all*) los archivos, incluso los ocultos
- `-h`: Con `-l` y `-s` imprime los tamaños de archivo en unidades comprensibles para humanos.

Si no se especifica el directorio, `ls` mostrará el contenido del directorio actual de trabajo, que en este caso es el *home*:

```
ls
```

```
bin
Descargas
Documentos
Escritorio
```

```
Imágenes
mambaforge
Música
Plantillas
projects
proyectos
Público
R
snap
```

Ahora se le ordena a `ls` mostrar los archivos en `/usr`:

```
ls /usr
```

```
bin
games
include
lib
lib32
lib64
libexec
libx32
local
sbin
share
src
```

Observe la diferencia en el formato de salida cuando se usa la opción `-l`:

```
ls -l -h ~
```

```
total 52K
drwxrwxr-x  2 hector hector 4,0K mar 24 16:08 bin
drwxr-xr-x  6 hector hector 4,0K mar 28 12:28 Descargas
drwxr-xr-x  2 hector hector 4,0K ene 18 20:29 Documentos
drwxr-xr-x  2 hector hector 4,0K ene  1 17:28 Escritorio
drwxrwxr-x  2 hector hector 4,0K mar 27 14:40 Imágenes
drwxrwxr-x 19 hector hector 4,0K mar 28 11:26 mambaforge
drwxrwxr-x  2 hector hector 4,0K mar 27 14:40 Música
drwxr-xr-x  2 hector hector 4,0K ene  1 17:28 Plantillas
```

```
drwxrwxr-x  9 hector hector 4,0K mar 27 13:10 projects
drwxrwxr-x  3 hector hector 4,0K mar 28 10:43 proyectos
drwxrwxr-x  2 hector hector 4,0K mar 27 14:40 Público
drwxrwxr-x  3 hector hector 4,0K mar 27 14:41 R
drwx------  5 hector hector 4,0K mar  5 17:14 snap
```

En el comando anterior además de especificar el directorio (*home* o `~`). Se usaron las opciones `-l` y `-h`. Como puede ver hay nueve columnas con información diferente para cada elemento de la lista. Veamos las partes de la fila del elemento `Documentos`:

- `drwxr-xr-x`: cadena que muestra los permisos de archivo[1] en cuatro partes:
  - `d`: tipo de archivo. Aquí la `d` significa que se trata de un directorio;
  - `rwx`: permisos del propietario (*owner*), `r` lectura (*read*), `w` escritura (*write*), y `x` ejecución (*execution*)
  - `r-x`: permisos del grupo (*group*) al que pertenece el propietario. Note que aparece `-` en lugar de `w`, es decir que no hay permiso de escritura.
  - `r-x`: permisos para otros (*other*) usuarios diferentes del propietario
- `2`: número de enlaces duros (*hard links*[2])
- `hector`: el propietario del archivo/directorio
- `hector`: el grupo al que pertenece el propietario
- `4,0K`: el tamaño del archivo/directorio. En este caso se muestra en Kilobytes gracias a la opción `-h`
- `ene 1 17:28`: la fecha (mes, día y hora) de creación/modificación del archivo
- `Documentos`: el nombre del elemento, en este caso un directorio

Existen archivos ocultos en Linux los cuales no son visibles normalmente con `ls` o en el explorador gráfico de archivos. Generalmente los archivos ocultos corresponden a archivos de configuración, y por esta razón, conviene tenerlos "ocultos" para protegerlos de modificaciones o eliminación accidentales. Para visualizarlos se usa la opción `-a` con `ls`:

```
ls -a
```

```
.
..
.bash_history
.bash_logout
.bashrc
```

[1] Permisos de archivos y directorios: https://www.linuxtotal.com.mx/index.php?cont=info_admon_011
[2] Wikipedia - Enlace duro: https://es.wikipedia.org/wiki/Enlace_duro

```
.bashrc.bak
bin
.cache
.common.config.sh
.conda
.conda.old
.config
.customrc.sh
Descargas
Documentos
.dotfiles
.dotnet
Escritorio
.fonts
.gitconfig
.gnupg
Imágenes
.java
.local
mambaforge
.mozilla
Música
.ncbi
.parallel
.pki
Plantillas
.profile
projects
proyectos
Público
.r
R
.Rhistory
.shutter
snap
.ssh
.sudo_as_admin_successful
.thunderbird
.TinyTeX
.vboxclient-clipboard.pid
.vscode
```

```
.vscode-R
.wget-hsts
.zcompdump
.zlogin
.zlogout
.zprezto
.zpreztorc
.zprofile
.zshenv
.zsh_history
.zshrc
```

Como puede ver, existen varios archivos ocultos, entre los que se destacan `.profile` y `.bashrc` que contienen configuración de la sesión de Bash, el intérprete de comandos (*shell*) por defecto de Ubuntu Linux.

> **!** Importante
>
> Note los dos primeros elementos en la salida del comando `ls -a`: `.` (punto) y `..` (punto punto). Todo directorio en Linux tiene estos dos elementos, los cuales son directorios. El `.` representa el mismo directorio listado, y el `..` representa el directorio padre, es decir el directorio que contiene al directorio listado.

Se pueden combinar varias opciones en en una sola "palabra", es decir con un solo `-` al principio, para modificar el comportamiento de los comandos de varias maneras. Por ejemplo en el siguiente comando se usan las opciones `-lhtr` para forzar a `ls` a mostrar el formato largo (`-l`), con tamaños de archivo entendibles por el humano (`-h`), ordenados por fecha de creación/modificación (`-t`), de manera inversa (`-r`), es decir, del más antiguo al más reciente. Además, se usa una ruta absoluta para especificar el directorio que se desea listar:

> **!** Importante
>
> Usted deberá cambiar `hector` por su propio nombre de usuario en este y el resto de ejemplos que involucren el directorio *home*.

```
ls -lhtr /home/hector
```

```
total 52K
drwxr-xr-x  2 hector hector 4,0K ene  1 17:28 Plantillas
```

```
drwxr-xr-x  2 hector hector 4,0K ene  1 17:28 Escritorio
drwxr-xr-x  2 hector hector 4,0K ene 18 20:29 Documentos
drwx------  5 hector hector 4,0K mar  5 17:14 snap
drwxrwxr-x  2 hector hector 4,0K mar 24 16:08 bin
drwxrwxr-x  9 hector hector 4,0K mar 27 13:10 projects
drwxrwxr-x  2 hector hector 4,0K mar 27 14:40 Público
drwxrwxr-x  2 hector hector 4,0K mar 27 14:40 Música
drwxrwxr-x  2 hector hector 4,0K mar 27 14:40 Imágenes
drwxrwxr-x  3 hector hector 4,0K mar 27 14:41 R
drwxrwxr-x  3 hector hector 4,0K mar 28 10:43 proyectos
drwxrwxr-x 19 hector hector 4,0K mar 28 11:26 mambaforge
drwxr-xr-x  6 hector hector 4,0K mar 28 12:28 Descargas
```

Observe el siguiente ejemplo. Aquí se usan las mismas opciones (`-lhrt`) y se especifica el directorio actual por medio del punto (`.`). Verá que el resultado es idéntico al ejemplo inmediatamente anterior dado que la ubicación actual es el *home*:

```
ls -lhrt .
```

```
total 52K
drwxr-xr-x  2 hector hector 4,0K ene  1 17:28 Plantillas
drwxr-xr-x  2 hector hector 4,0K ene  1 17:28 Escritorio
drwxr-xr-x  2 hector hector 4,0K ene 18 20:29 Documentos
drwx------  5 hector hector 4,0K mar  5 17:14 snap
drwxrwxr-x  2 hector hector 4,0K mar 24 16:08 bin
drwxrwxr-x  9 hector hector 4,0K mar 27 13:10 projects
drwxrwxr-x  2 hector hector 4,0K mar 27 14:40 Público
drwxrwxr-x  2 hector hector 4,0K mar 27 14:40 Música
drwxrwxr-x  2 hector hector 4,0K mar 27 14:40 Imágenes
drwxrwxr-x  3 hector hector 4,0K mar 27 14:41 R
drwxrwxr-x  3 hector hector 4,0K mar 28 10:43 proyectos
drwxrwxr-x 19 hector hector 4,0K mar 28 11:26 mambaforge
drwxr-xr-x  6 hector hector 4,0K mar 28 12:28 Descargas
```

Ahora veamos el contenido del directorio oculto `.local` ubicado en el *home* especificando la ruta absoluta:

```
ls /home/hector/.local
```

```
share
```

En el directorio `.local` solo existe el directorio `share`.

Ahora veamos el contenido del mismo directorio `.local` pero usando la ruta relativa desde el *home*:

```
ls .local
```

```
share
```

Como puede observar solo cambia el tipo de ruta que se pasa a `ls` pero el resultado es el mismo.

3.2.3. touch

Uso

```
touch [OPCIONES] ARCHIVO
```

Actualiza la fecha y hora de acceso y modificación de `ARCHIVO` a la hora actual. Si `ARCHIVO` no existe, `touch` creará el archivo vacío.

Generalmente `touch` se usa para crear archivos vacíos:

```
touch mi_archivo.txt
```

Como `mi_archivo.txt` no existe, `touch` lo crea. Verifiquemos con `ls` que `mi_archivo.txt` fue creado:

```
ls -lh --time-style +%H:%M:%S
```

```
total 52K
drwxrwxr-x  2 hector hector 4,0K 16:08:05 bin
drwxr-xr-x  6 hector hector 4,0K 12:28:34 Descargas
drwxr-xr-x  2 hector hector 4,0K 20:29:56 Documentos
drwxr-xr-x  2 hector hector 4,0K 17:28:15 Escritorio
drwxrwxr-x  2 hector hector 4,0K 14:40:26 Imágenes
drwxrwxr-x 19 hector hector 4,0K 11:26:33 mambaforge
-rw-rw-r--  1 hector hector    0 16:00:39 mi_archivo.txt
```

```
drwxrwxr-x  2 hector hector 4,0K 14:40:26 Música
drwxr-xr-x  2 hector hector 4,0K 17:28:15 Plantillas
drwxrwxr-x  9 hector hector 4,0K 13:10:50 projects
drwxrwxr-x  3 hector hector 4,0K 10:43:50 proyectos
drwxrwxr-x  2 hector hector 4,0K 14:40:26 Público
drwxrwxr-x  3 hector hector 4,0K 14:41:33 R
drwx------  5 hector hector 4,0K 17:14:01 snap
```

> **Nota**
>
> Se usó la opción `--time-style +%H:%M:%S` con `ls` para que se muestre la hora (`+%H`), los minutos (`%M`) y los segundos (`%S`) en la hora de modificación de los archivos.

Ahora `mi_archivo.txt` aparece en el listado con tamaño de archivo 0 (vacío; columna 4).

Si usa `touch` nuevamente sobre el mismo archivo, se actualizará la hora de modificación pues el archivo ya existe previamente:

```
touch mi_archivo.txt
ls -lh --time-style +%H:%M:%S
```

```
total 52K
drwxrwxr-x  2 hector hector 4,0K 16:08:05 bin
drwxr-xr-x  6 hector hector 4,0K 12:28:34 Descargas
drwxr-xr-x  2 hector hector 4,0K 20:29:56 Documentos
drwxr-xr-x  2 hector hector 4,0K 17:28:15 Escritorio
drwxrwxr-x  2 hector hector 4,0K 14:40:26 Imágenes
drwxrwxr-x 19 hector hector 4,0K 11:26:33 mambaforge
-rw-rw-r--  1 hector hector    0 16:00:44 mi_archivo.txt
drwxrwxr-x  2 hector hector 4,0K 14:40:26 Música
drwxr-xr-x  2 hector hector 4,0K 17:28:15 Plantillas
drwxrwxr-x  9 hector hector 4,0K 13:10:50 projects
drwxrwxr-x  3 hector hector 4,0K 10:43:50 proyectos
drwxrwxr-x  2 hector hector 4,0K 14:40:26 Público
drwxrwxr-x  3 hector hector 4,0K 14:41:33 R
drwx------  5 hector hector 4,0K 17:14:01 snap
```

Puede ver que la hora de modificación es más reciente (columna 5) que la mostrada cuando se verificó la creación del archivo.

3.2.4. cd

Uso

```
cd [DIRECTORIO]
```

Cambia el directorio de trabajo (*change directory*) a la ubicación especificada en `[DIRECTORIO]`. Si no especifica un directorio, por defecto `cd` cambiará la ubicación al directorio *home* del usuario.

Veamos algunos ejemplos del comando `cd`. Primero se verifica la ubicación actual con `pwd`:

```
pwd
```

```
/home/hector
```

Como pudo observar, `cd` no presenta en pantalla ningún resultado o mensaje asociado. El resultado solo se verifica consultado la ubicación actual, como en el anterior ejemplo con `pwd`. En Ubuntu Linux, también se puede ver la ubicación actual en el símbolo del sistema (ver Sección 2.3) justo después de los dos puntos (`:`), en este caso el *home* representado por `~`:

```
hector@Ubuntu:~$
```

Ahora se cambia la ubicación con `cd` al directorio `Descargas` con `cd` usando la ruta relativa:

```
cd Descargas
```

Su *prompt* debería indicar ahora que su directorio actual es `Descargas`:

```
hector@Ubuntu:~/Descargas$
```

Nota

Como puede ver, la ubicación en el *prompt* siempre abrevia la ruta absoluta a su directorio *home* con `~`.

Ahora, si verifica la ubicación con `pwd` obtendrá la ruta absoluta a `Descargas`:

```
pwd
```

```
/home/hector/Descargas
```

Usando `cd` sin especificar ningún directorio cambiará la ubicación al *home*:

```
# Se usa cd para ir al home
cd
# Se verifica la ubicación actual
pwd
```

```
/home/hector
```

> **Nota**
>
> Las líneas del anterior bloque de comandos que empiezan con el símbolo `#`, son comentarios. Los comentarios son líneas que nunca son ejecutadas por el *shell*, en este caso Bash, y que aportan información adicional sobre los comandos a ejecutar.

Al igual que con otros comando que reciben rutas como argumentos, `cd` también trabaja con rutas absolutas. En el siguiente ejemplo se cambia al directorio `Documentos` usando una ruta absoluta:

```
cd /home/hector/Documentos
pwd
```

```
/home/hector/Documentos
```

Ahora se cambia al directorio padre del directorio actual con `..`.

```
cd ..
pwd
```

```
/home/hector
```

```
cd /home/hector/Descargas
```

3.2.5. mkdir

Uso

```
mkdir [OPCIONES] [DIRECTORIO]
```

Crea un nuevo directorio con el nombre especificado por `[DIRECTORIO]`.
Opciones útiles:

- `-p`: Crea los directorios padre de la ruta si es necesario

Veamos algunos ejemplos del comando `mkdir`. Desde el *home*, se crea el directorio `pruebas`:

```
cd
mkdir pruebas
```

`mkdir` tampoco provee un mensaje después de crear un directorio, a menos que haya ocurrido un error. Se puede usar `ls` para verificar la creación del directorio:

```
ls
```

```
bin
Descargas
Documentos
Escritorio
Imágenes
mambaforge
mi_archivo.txt
Música
Plantillas
projects
proyectos
pruebas
Público
R
snap
```

Si intenta crear un directorio que ya existe, obtendrá un error:

```
mkdir pruebas
```

```
mkdir: no se puede crear el directorio «pruebas»: El archivo ya existe
```

Se pueden crear varios directorios a la vez con `mkdir`:

```
mkdir pruebas2 pruebas/dir1 pruebas/dir2
```

Verifiquemos la creación del directorio `pruebas2` en el *home*:

```
ls
```

```
bin
Descargas
Documentos
Escritorio
Imágenes
mambaforge
mi_archivo.txt
Música
Plantillas
projects
proyectos
pruebas
pruebas2
Público
R
snap
```

Ahora verifiquemos la creación de `dir1` y `dir2` en `pruebas`:

```
ls pruebas
```

```
dir1
dir2
```

3.2.6. cp

Uso

```
cp [OPCIONES] ORIGEN DESTINO
cp [OPCIONES] ORIGEN DIRECTORIO
```

Copia un archivo/directorio especificado en `ORIGEN` a `DESTINO`, o múltiples `ORIGEN(es)` a `DIRECTORIO`.
Opciones útiles:

- `-r`: Copia directorios recursivamente.

Copiemos el archivo `mi_archivo.txt`, creado anteriormente, al directorio `pruebas`:

```
cp mi_archivo.txt pruebas
ls pruebas
```

```
dir1
dir2
mi_archivo.txt
```

Ahora, intentemos copiar el directorio dir1 completo con su contenido usando la opción `-r` (copia recursiva) de `cp`. Primero se crea un archivo al interior de `dir1`:

```
touch pruebas/dir1/archivo2.txt
# contenido de dir1
ls pruebas/dir1
```

```
archivo2.txt
```

Ahora se copia todo el directorio `dir1` a `dir2`:

```
cp -r pruebas/dir1 pruebas/dir2
# contenido de dir2
ls pruebas/dir2
```

```
dir1
```

El directorio `dir1` dentro de `dir2` debería contener a `archivo2.txt`:

```
ls pruebas/dir2/dir1
```

```
archivo2.txt
```

3.2.7. mv

Uso

```
mv [OPCIONES] ORIGEN DESTINO
mv [OPCIONES] [ORIGEN] DIRECTORIO
```

- Uso 1: Renombra el archivo/directorio especificado en `ORIGEN` a `DESTINO`, o
- Uso 2: mueve el(los) elemento(s) especificado(s) en `[ORIGEN]` a `DIRECTORIO`.

Veamos el primer uso de `mv` para renombrar el archivo `mi_archivo.txt`:

```
mv mi_archivo.txt archivo1.txt
```

Ahora verifiquemos el cambio de nombre:

```
ls
```

```
archivo1.txt
bin
Descargas
Documentos
Escritorio
Imágenes
mambaforge
Música
Plantillas
projects
proyectos
pruebas
pruebas2
```

```
Público
R
snap
```

Efectivamente el archivo que aparece ahora es `archivo1.txt`.

Veamos ahora el segundo uso de `mv` para mover el archivo `archivo1.txt` al directorio `pruebas`:

```
mv archivo1.txt pruebas
```

Verifiquemos que `archivo1.txt` aparece dentro de pruebas:

```
echo "*** Verificando contenido de home ***"
ls ~
echo "*** Verificando contenido de pruebas ***"
ls pruebas
```

```
*** Verificando contenido de home ***
bin
Descargas
Documentos
Escritorio
Imágenes
mambaforge
Música
Plantillas
projects
proyectos
pruebas
pruebas2
Público
R
snap
*** Verificando contenido de pruebas ***
archivo1.txt
dir1
dir2
mi_archivo.txt
```

> **Nota**
>
> En el ejemplo anterior se usó por primera vez el comando `echo`. Este comando sirve para imprimir mensajes por pantalla. Es recomendable que siempre el texto del mensaje que se pasa a echo vaya entre comillas.

3.2.8. cat

Uso

```
cat [OPCIONES] [ARCHIVO]
```

Concatena el(los) archivo(s) especificados en `ARCHIVO` a la pantalla (salida estándar).

El comando `cat` es útil para leer archivos y mostrarlos en pantalla. Por ejemplo, veamos el contenido del archivo de configuración `.profile` que está en el *home*:

```
cat .profile
```

```
# ~/.profile: executed by the command interpreter for login shells.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umask
# for ssh logins, install and configure the libpam-umask package.
#umask 022

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
    . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
```

```
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi
```

Los archivos ocultos en Linux siempre tienen un punto (`.`) como primer carácter en el nombre. Para ocultar un archivo basta con anteponer un punto en el nombre. Para ver los archivos oculto con `ls`, se debe usar la opción `-a` (`ls -a`).

El archivo `.profile` del ejemplo anterior es un archivo de configuración vital para el funcionamiento de la línea de comandos. Este archivo permanece oculto para evitar la modificación o eliminación accidental. Absténgase de modificarlo a menos que sepa muy bien lo que hace.

Ahora veremos como imprimir dos archivos con `cat`, pero primero descarguemos los archivos. Para descargar los archivos ejecute la siguientes órdenes:

```
wget -O pruebas/ejemplo_01.fasta \
  https://raw.githubusercontent.com/hspitia/linux_example_files/main/e
 ↪  jemplo_01.fasta
wget -O pruebas/ejemplo_02.fasta \
  https://raw.githubusercontent.com/hspitia/linux_example_files/main/e
 ↪  jemplo_02.fasta
```

```
--2023-03-28 16:00:45--  https://raw.githubusercontent.com/hspitia/linu
 ↪  x_example_files/main/ejemplo_01.fasta
Resolviendo raw.githubusercontent.com (raw.githubusercontent.com)...
 ↪  185.199.109.133, 185.199.110.133, 185.199.111.133, ...
Conectando con raw.githubusercontent.com
 ↪  (raw.githubusercontent.com)[185.199.109.133]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 197 [text/plain]
Guardando como: “pruebas/ejemplo_01.fasta”

     0K                                                      100% 9,21M=0s

2023-03-28 16:00:46 (9,21 MB/s) - “pruebas/ejemplo_01.fasta” guardado
 ↪  [197/197]
```

```
--2023-03-28 16:00:46--  https://raw.githubusercontent.com/hspitia/linu
 ↪  x_example_files/main/ejemplo_02.fasta
Resolviendo raw.githubusercontent.com (raw.githubusercontent.com)...
 ↪  185.199.108.133, 185.199.111.133, 185.199.110.133, ...
Conectando con raw.githubusercontent.com
 ↪  (raw.githubusercontent.com)[185.199.108.133]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 196 [text/plain]
Guardando como: “pruebas/ejemplo_02.fasta”

     0K                                                      100% 31,8M=0s

2023-03-28 16:00:46 (31,8 MB/s) - “pruebas/ejemplo_02.fasta” guardado
 ↪  [196/196]
```

Nota

En el ejemplo anterior se descargaron archivos desde la terminal usando el comando `wget`. Veamos las partes de uno de los comandos ejecutados:

- `wget`: el nombre del comando para descargar el archivo
- `-O pruebas/ejemplo_01.fasta`: Opción (`-O`) y nombre (*path*) con el cual queremos guardar el archivo a descargar
- `https://raw.githubusercontent.com/hspitia/linux_example_-files/main/ejemplo_01.fasta`: la dirección (URL) en la Internet del archivo que se descargará

Otro comando muy utilizado para descargar archivos es `curl`. Es recomendable que explore la ayuda de `wget` y `curl` para saber más sobre su uso y funciones.

Nota

En el ejemplo anterior el carácter `\` (*backslash*) se usa al final de la línea para indicar que el comando continúa en la línea siguiente. Se usa cuando el comando es más largo (en caracteres) que el ancho de la terminal.

Verifiquemos ahora que los archivos fueron descargados en el directorio `pruebas`:

```
ls pruebas
```

```
archivo1.txt
dir1
dir2
ejemplo_01.fasta
ejemplo_02.fasta
mi_archivo.txt
```

Ahora, se imprime el contenido de los archivos descargados usando el comando `cat`:

```
cat pruebas/ejemplo_01.fasta pruebas/ejemplo_02.fasta
```

```
>Secuencia_01
CGAGTCAGCTGTCAGCTAGTCGATCGCGATATGCTATATCGTTCTTTCGATGCTACTTTA
TTCGCTGTAGAAAATGCTCGGCTTAGCTGATCGCTCGATCGACACGATGCATCGATGTTC
GCTAGTACGTACGTACGTACGCATCGATCGGCTAGCCTATATATCGGATCGATCGGATAA
>Secuencia_02
TTCGCCTCCCATTATGCTCGGCTTAGCTGATCGCTCGATCGACACGATGCATCGATGTTC
CGGGGCTAATCGATCCTAGCTAGCTCTATATTGCTATATCGTTCTTTCGATGCTACTTTA
GCTAGTACGTACGTACGTACGTAGATAGATTCGGCTATCGGTTATATAGGATTTATTAAA
```

> **Nota**
>
> El formato FASTA[3] (`.fasta`), uno de lo más comunes en bioinformática, es un formato de solo texto, usado para almacenar una o más secuencias de ADN o de proteínas. Usualmente cuando se almacena más de una secuencia en un archivo, se denomina **multifasta**. Cada secuencia tiene dos partes: (1) el identificador, que consiste de una sola línea con el nombre de la secuencia y que debe iniciar con el símbolo `>` (p. ej. `>Mi secuencia`); y (2) la secuencia, que puede tener una o más líneas con los caracteres que representan la secuencia. Si la secuencia es de ADN, los caracteres representan nucleótidos[4] (`A`, `C`, `G`, `T` o `U`), y si es de proteínas, representan aminoácidos[5].

Cada archivo contiene cuatro líneas de texto (una empezando con `>` con el nombre de la secuencia, y tres con la secuencia de ADN). Como se pasaron los nombres de los dos archivos, `cat` imprime los contenidos uno seguido del otro sin ninguna separación visible, es decir los concatena.

[5]Wikipedia - Formato FASTA: https://es.wikipedia.org/wiki/Formato_FASTA
[5]Wikipedia - Nucleótido: https://es.wikipedia.org/wiki/Nucle%C3%B3tido
[5]Wikipedia - Aminoácido: https://es.wikipedia.org/wiki/Amino%C3%A1cido

3.3. Redirección de entrada y salida

La redirección de entrada y salida es un concepto poderoso de comunicación entre procesos, sumamente útil en la línea de comandos para crear archivos y construir *pipelines* (tuberías) de programas. Estos *pipelines* son fundamentales en bioinformática para procesar grandes cantidades de datos de manera automatizada. Para comprender mejor la redirección primero se deben entender los flujos estándar.

3.3.1. Flujos estándar

La entrada y salida de información en Linux está organizada en flujos (*streams*) estándar (*std*). Existen tres flujos estándar: (1) de entrada (*in*) llamado `stdin` que se encarga de llevar información hacia los programas o procesos; (2) de salida (*out*) llamado `stdout` que lleva información (o resultados) desde los programas hasta el usuario u otros programas; y (3) de error (*error*) llamado `stderr` que lleva información solo de errores desde los programas. Por defecto, la entrada estándar viene desde el teclado, y la salida y el error estándar son dirigidos a la pantalla (Figura 3.2). Cada flujo tiene asociado un número que lo identifica: `0` para `stdin`, `1` para `stdout`, y `2` para `stderr`.

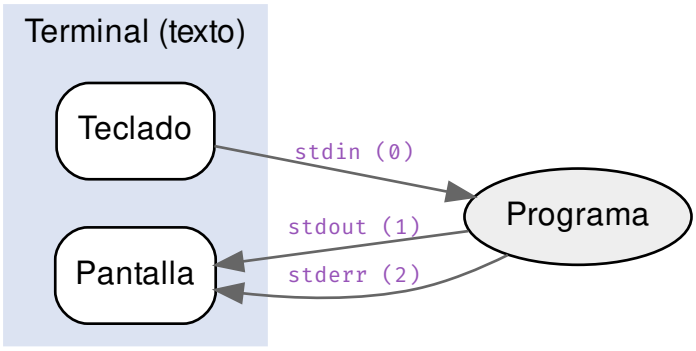


Figura 3.2.: Los flujos estándar de entrada (`stdin`), salida (`stdout`) y error (`stderr`) para un programa (proceso) ejecutado en una terminal de texto. *Diagrama adaptado del artículo de Wikipedia "Redirection (computing)".*

3.3.2. Redirección de flujos

La redirección consiste en redirigir los flujos hacia otros destinos diferentes de los dispuestos por defecto. Para redirigir la salida se usa el carácter `>`, y para redirigir la

entrada se usa `<`.

3.3.2.1. Salida

Cuando usa un comando, como `ls`, vemos el resultado por pantalla dado que este es el dispositivo de salida por defecto. Usemos entonces la redirección para enviar la salida de `ls` a un archivo en lugar de la pantalla:

```
ls > pruebas/salida.txt
```

Como se esperaba, el comando anterior no generó ninguna salida por pantalla y en cambio la escribió en el archivo `pruebas/salida.txt`. Si verificamos el contenido, observaremos que corresponde con la lista de los archivo y carpetas del directorio personal:

```
cat pruebas/salida.txt
```

```
bin
Descargas
Documentos
Escritorio
Imágenes
mambaforge
Música
Plantillas
projects
proyectos
pruebas
pruebas2
Público
R
snap
Vídeos
```

Ahora vamos a redirigir la salida del comando `echo` al archivo `pruebas/salida.txt`, y posteriormente mostrar su contenido:

```
# Se redirige la salida de echo
echo "Prueba de redirección 1" > pruebas/salida.txt
```

```
# Se muestra por pantalla el contenido
cat pruebas/salida.txt
```

```
Prueba de redirección 1
```

¿Qué pasó con el contenido del archivo? Ahora `pruebas/salida.txt` contiene solo la salida de `echo`; el contenido anterior (salida de `ls`) fue sobreescrito.

Para usar la redirección y evitar que el contenido del archivo sea sobreescrito, se debe usar el doble símbolo de redirección `>>`. Así se añade (*append* en inglés) al contenido del archivo:

```
# Se redirige la salida de ls añadiendo al archivo
ls >> pruebas/salida.txt

# Se verifica el contenido
cat pruebas/salida.txt
```

```
Prueba de redirección 1
bin
Descargas
Documentos
Escritorio
Imágenes
mambaforge
Música
Plantillas
projects
proyectos
pruebas
pruebas2
Público
R
snap
Vídeos
```

Ahora vemos que la salida de `ls` fue añadida al contenido previo.

3.3.2.2. Entrada

Podemos redirigir la entrada de un comando desde una fuente alternativa con `<`. Por ejemplo, en lugar de pasar a `cat` el nombre de archivo como un argumento, podemos usar la redirección:

```
cat < pruebas/salida.txt
```

```
Prueba de redirección 1
bin
Descargas
Documentos
Escritorio
Imágenes
mambaforge
Música
Plantillas
projects
proyectos
pruebas
pruebas2
Público
R
snap
Vídeos
```

Vemos que el resultado es el mismo con la redirección que pasando el archivo directamente.

En el anterior ejemplo no hay mucha utilidad adicional a parte de mostrar un ejemplo, sin embargo, la redirección de entrada es útil en otros casos como por ejemplo leer archivos línea a línea con un bucle o ciclo (*loop* en inglés) `while`:

```
while read line
do
  echo "-> $line"
done < pruebas/salida.txt# <1>
```

① Se pasa el archivo al bucle para que sea leído. Se usa la redirección de entrada (`<`)

```
-> Prueba de redirección 1
-> bin
-> Descargas
-> Documentos
-> Escritorio
-> Imágenes
-> mambaforge
-> Música
-> Plantillas
-> projects
-> proyectos
-> pruebas
-> pruebas2
-> Público
-> R
-> snap
-> Vídeos
```

Tip

En el ejemplo anterior se usó una estructura repetitiva, también llamada ciclo o bucle, `while`[6]. La finalidad de esta estructura es que se repitan una serie de acciones o comandos mientras (*while* en inglés) se cumpla una condición (sea verdadera). Veamos la estructura:

```
while CONDICION# <1>
do# <2>
  ACCION_1# <3>
  ACCION_2# <3>
    ...# <3>
  ACCION_N# <3>
done# <4>
```

① Inicio del ciclo y condición. Se inicia siempre con `while` (mientras) y luego la condición. `CONDICION` es una expresión lógica, es decir que siempre representa un valor de verdadero (`true`) o falso (`false`). Siempre que la condición sea verdadera, las acciones se ejecutarán, es decir, el ciclo termina cuando la condición se vuelve falsa.

② Inicio del bloque de acciones. Siempre se marca con `do` (hacer), el inicio de las acciones que se repetirán.

③ Las acciones a repetir. Las acciones (comandos) pueden ser una o muchas, e incluso otros ciclos (ciclos anidados).
④ Final del bucle. Se marca con `done` (hecho) el final de las acciones y del bucle en general.

En el ejemplo anterior la condición se hará verdadera siempre que hayan líneas en el archivo por leer, así que la primera línea se lee como "mientras se pueda leer una línea". Aquí el comando `read` (leer) se encarga de leer una línea de texto y guardarla temporalmente en la variable `line`. La única acción que se ejecuta en el bucle anterior, es imprimir con `echo` el contenido de la línea de texto guardada en la variable `line`. Observe que para poder consultar (dereferenciar) el contenido de la variable se debe anteponer el símbolo `$` (`$line`). También observe que se imprimieron los caracteres `->` antes del contenido de cada línea; esto fue solo para cambiar un poco la apariencia de la salida de todo el comando. Finalmente, en el cierre del bucle se usa la redirección de entrada (`<`) para pasar al bucle el archivo que será leído. Este es un caso especial del bucle `while` muy útil para ejecutar acciones dependientes del contenido de un archivo.

3.3.3. *Pipelines* o tuberías

Uno de los mecanismos más útiles y poderosos de la redirección es el de los *pipelines*, *pipes* o tuberías, pues permiten la ejecución de múltiples programas o comandos de manera secuencial como si estuvieran conectados por una tubería (*pipe* en inglés). En los *pipelines* la salida de un programa es pasada directamente como entrada a otro programa usando el carácter `|` (barra vertical o *pipe*). La Figura 3.3 muestra un diagrama de un *pipeline* de tres programas y sus correspondientes flujos de entrada y salida.

Veamos los *pipes* en acción. Supongamos que se desea saber cuántos archivos de texto (con extensión `.txt`) hay en el directorio `pruebas`. Esto se puede lograr con una combinación de los programas `ls` y `wc` conectados con un *pipe*. Primero intentemos listar solo los archivos de texto en `pruebas`:

```
ls pruebas/*.txt
```

```
pruebas/archivo1.txt
pruebas/mi_archivo.txt
pruebas/salida.txt
```

[6]Wikipedia - Bucle while: https://es.wikipedia.org/wiki/Bucle_while

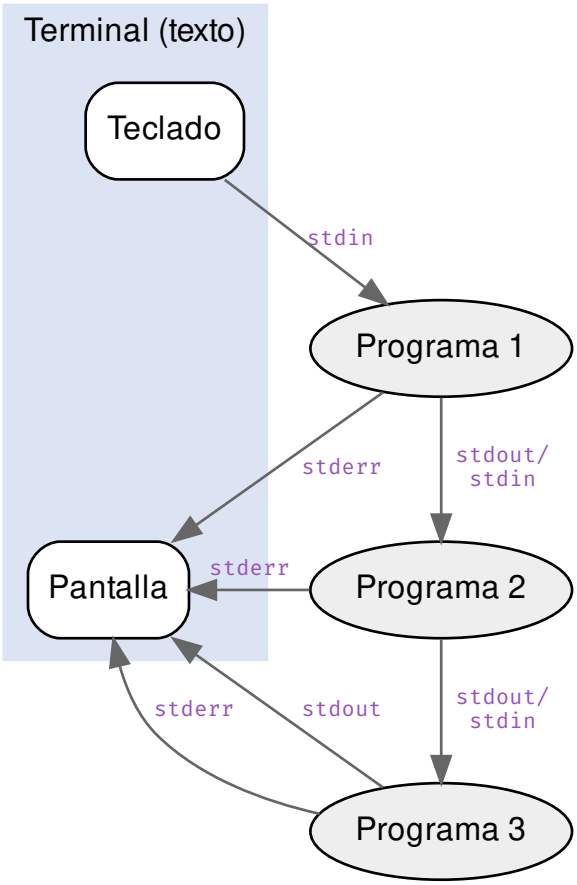


Figura 3.3.: Representación de los flujos de entrada/salida de un *pipeline* (tubería) de tres programas ejecutados en una terminal de texto. En un *pipeline*, la salida de un programa (`stdout`) es la entrada (`stdin`) para el siguiente programa. *Diagrama adaptado del artículo de Wikipedia "Redirection (computing)".*

En el comando anterior, se usa `ls` para listar cualquier archivo (`*`) con extensión `.txt` que se encuentre en el directorio `pruebas`. En el resultado vemos que `ls` nos muestra tres archivos, imprimiendo uno en cada línea independiente.

Ahora se redirige la salida del anterior comando hacia el programa `wc`:

```
ls pruebas/*.txt | wc -l
```

```
3
```

> **Nota**
>
> En el ejemplo anterior se usó el comando `wc`[7] (por ***w**ord **c**ount*) con el cual se cuenta el número de líneas, palabras, bytes, y caracteres en uno o más archivos. Específicamente se usó la opción `-l` que cuenta el número de líneas de un archivo, o como en este caso, de la entrada estándar que es la salida de `ls`.

En el ejemplo anterior se usa el símbolo `|` para redirigir la salida de `ls` como entrada del comando a la derecha, es decir `wc`. `ls` lista los tres archivos de texto y `wc` cuenta entonces el número de líneas que hay en la entrada, por esto `wc` retorna `3`.

> **Tip**
>
> Para comprender mejor los *pipelines* se recomienda ejecutar cada parte por separado. Así verá los resultados intermedios de cada programa que quedan ocultos cuando se ejecuta el *pipeline* completo.

Veamos otro ejemplo. Una tarea muy común es contar el número de secuencias que hay en un archivo multifasta. Esto se puede lograr con los comandos `grep` y `wc` en un *pipeline*. Además, podemos agregar un nivel más en el *pipeline* si agregamos el comando `cat`:

```
cat pruebas/*.fasta | grep '>' | wc -l
```

```
2
```

[7] *Wikipedia - wc (Unix)*: https://en.wikipedia.org/wiki/Wc_(Unix)

> **Nota**
>
> En el ejemplo anterior se usó el comando `grep`[8] que sirve para buscar contenido específico en uno o más archivos. `grep` es una herramienta poderosa que se basa en el uso de **expresiones regulares**[9], las cuales son instrucciones especiales que representan patrones de búsqueda. Específicamente, se usó `grep` para buscar el carácter `>` en la entrada estándar (salida de `cat`).

Analicemos el pipeline. En la primera parte se usa `cat` para concatenar todos los archivos FASTA (`*.fasta`) que están en `pruebas`, tal como en el último ejemplo de la sección Sección 3.2.8. La salida de `cat` es entonces un conjunto de dos secuencias en formato FASTA. Después, la salida de `cat` se convierte en la entrada de `grep` por medio de `|` (*pipe*). En la segunda parte del *pipeline*, `grep` busca el carácter `>` e imprime solo las líneas de la entrada en las cuales hay una o más ocurrencias de ese carácter, es decir las dos líneas que corresponden a los nombres de las secuencias, que por regla, deben empezar con `>`. Luego, la salida de `grep` se convierte en la entrada para el programa `wc` usando de nuevo `|`. En la tercera y última parte, se cuentan las líneas de la entrada con `wc -l`, por lo que finalmente el resultado es `2`.

[9]Wikipedia - Grep: https://es.wikipedia.org/wiki/Grep
[9]Wikipedia - Expresión regular: https://es.wikipedia.org/wiki/Expresión_regular

Bibliografía

Espitia-Navarro, Hector F, Aroon T Chande, Shashwat D Nagar, Heather Smith, I King Jordan, y Lavanya Rishishwar. 2020. «STing: accurate and ultrafast genomic profiling with exact sequence matches». *Nucleic Acids Research* 48 (14): 7681-89. https://doi.org/10.1093/nar/gkaa566.

Microsoft. 2021. «What Is Windows Subsystem for Linux». Microsoft docs. 2021. https://docs.microsoft.com/en-us/windows/wsl/about.

SUSE. 2021. «What Is a Linux Distribution? | Answer from SUSE Defines». SUSE Defines. 2021. https://www.suse.com/suse-defines/definition/linux-distribution/.

Acerca del autor

Universidad de **Nariño**
FUNDADA EN 1904

C E S U N
CENTRO DE ESTUDIOS EN SALUD
Universidad de **Nariño**

Héctor Fabio Espitia-Navarro es Ph.D. en bioinformática del *Georgia Institute of Technology* (Atlanta, Georgia, EE. UU.) y actualmente se desempeña como científico de datos (*Data Scientist*) especializado en bionformática, en el *Corporate Reserach Materials Laboratory* de la compañía 3M (Saint Paul, Minnesota, EE. UU.). Obtuvo su título de Ingeniero en Sistemas y un Máster en Ingeniería en la Universidad del Valle (Cali, Valle, Colombia). Es miembro del Grupo de Investigación Salud Pública de la Universidad de Nariño[10] (Pasto, Nariño, Colombia), y del Grupo de Investigación de Bioinformática y Biocomputación de la Universidad del Valle[11] (Cali, Valle, Colombia). Además, es colaborador del Centro de Investigación de la Caña de Azúcar de Colombia - Cenicaña[12] (Cali, Valle, Colombia) y del *Indian River Research Education Center*[13] de *Florida University* (Fort Pierce, Florida, EE. UU.).

El Dr. Espitia-Navarro se especializa en investigación en bioinformática aplicada, especialmente enfocada en la genómica comparativa y la metagenómica, y se interesa por el desarrollo de software y algoritmos bioinformáticos. Empezó su trabajo en bioinformática desde el pregrado con el desarrollo de una aplicación para análisis multifractal de secuencias de ADN y proteínas. Posteriormente, durante su maestría trabajó en Cenicaña en la integración de la bioinformática para el mejoramiento del cutivo de la caña de azúcar. Específicamente, desarrolló un modelo computacional para la identificación y *ranking* de genes candidatos relacionados con la tolerancia a estrés hídrico en caña de azúcar. Después, durante su doctorado trabajó en el desarrollo de algoritmos para epidemiología molecular basada en secuenciación de segunda generación (NGS del inglés *Next Generation Sequencing*). Su principal proyecto fue STing (Espitia-Navarro et al. 2020), un algoritmo

[10]Grupo de Investigación Salud Pública, GrupLac, Minciencias, Colombia: https://scienti.minciencias.gov.co/gruplac/jsp/visualiza/visualizagr.jsp?nro=00000000008168
[11]Grupo de Investigación de Bioinformática y Biocomputación, GrupLac, Minciencias, Colombia: https://scienti.minciencias.gov.co/gruplac/jsp/visualiza/visualizagr.jsp?nro=00000000000759
[12]Cenicaña, sitio Web: https://www.cenicana.org/
[13]IRREC, sitio Web: https://irrec.ifas.ufl.edu/

libre de alineamiento de secuencias basado en frecuencias de *k-mers*, para la eficiente caracterización y clasificación taxonómica de patógenos bacterianos. Actualmente su trabajo en la compañía 3M está orientado al uso la bioinformática y el aprendizaje automático (*machine learning*) en la investigación en los campos de la purificación de biofarmacéuticos y la cicatrización de heridas.

A. Instalación de Ubuntu Linux como máquina virtual

A.1. Descarga de archivos

1. Descargar Ubuntu. Usando su explorador Web, ingrese a https://ubuntu.com/download/desktop.

   Se recomienda descargar la versión LTS (del inglés *Long Term Support*) que ofrece actualizaciones durante 5 años.

   Busque el botón "Download" y haga clic en él. Esto iniciará la descarga del archivo de imagen de Ubuntu Linux (.iso). Guarde el archivo en la ubicación deseada.

   Para este manual se descargó la versión 20.04 (LTS) de escritorio de Ubuntu Linux.

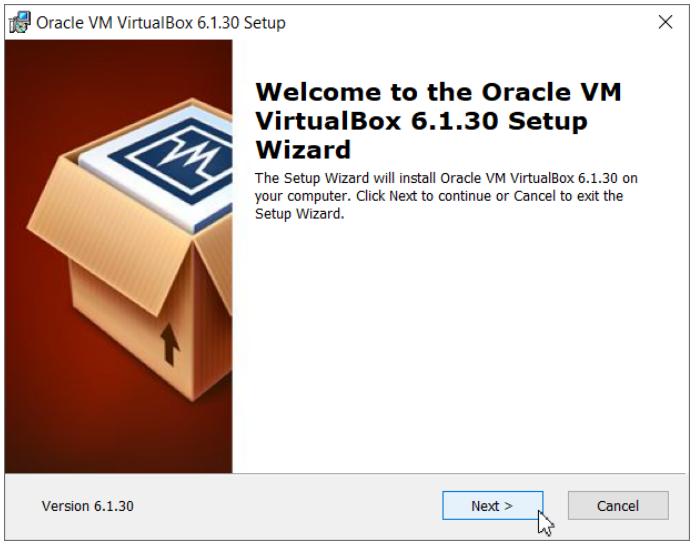
2. Descargar VirtualBox. Vaya a https://www.virtualbox.org/wiki/Downloads. Busque el enlace llamado "Windows hosts" y haga clic en él. Esto iniciará la descarga del instalador. Guarde el instalador en la ubicación deseada.

A.2. Instalación de VirtualBox

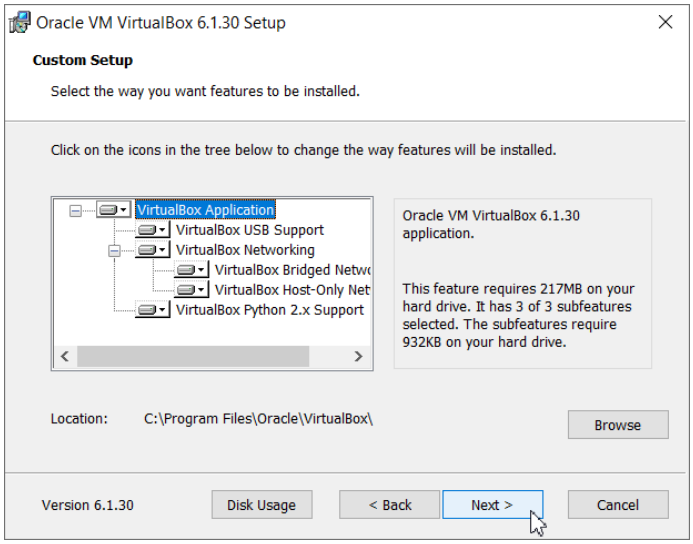
Las instrucciones que encontrará a continuación son para la versión 6.1.30 (build 148432) de VirtualBox.

3. Usando el explorador de archivos de Windows, busque el instalador que fue descargado en el paso anterior y ejecútelo haciendo doble clic en él, y siga estas instrucciones:

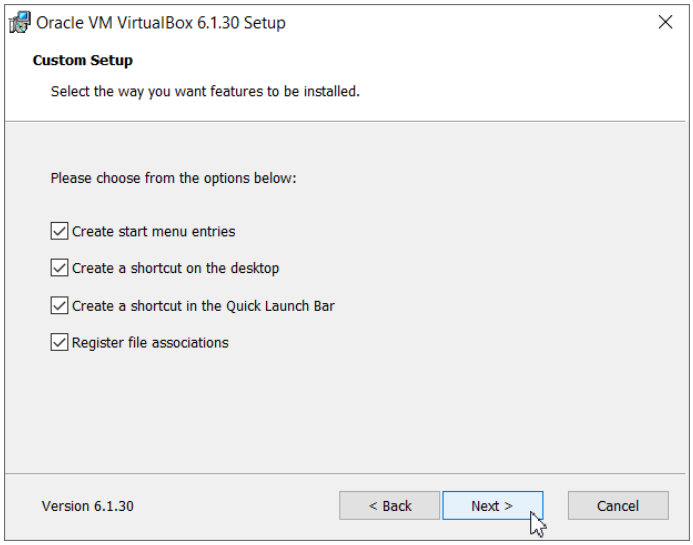
4. Clic en "Next".



5. Elementos de instalación. Instale los elementos por defecto dando clic en “Next”

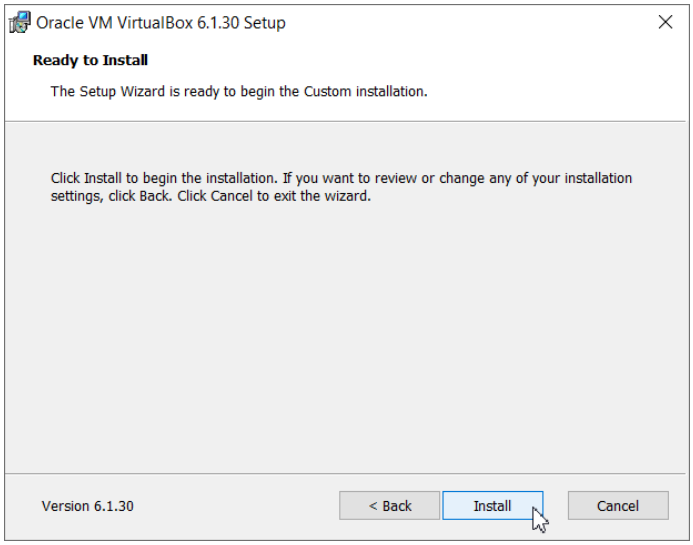


6. Haga clic en “Next”

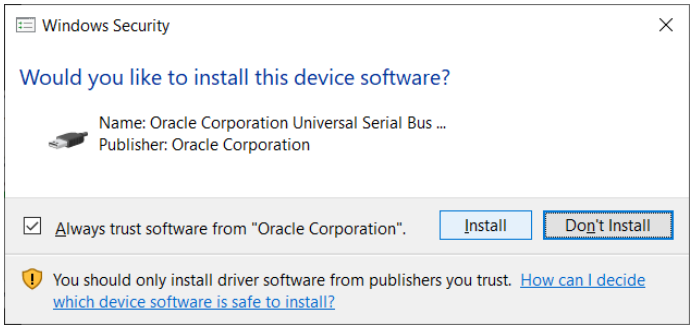


7. Acepte la instalación de interfaces de red (*Network Interfaces*) haciendo clic en "Yes". VirtualBox necesita estas interfaces para proveer acceso a Internet a su máquina virtual.

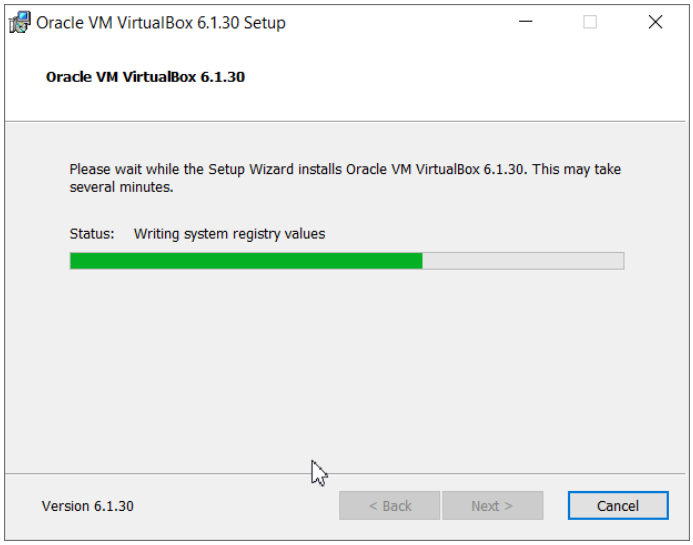
8. Inicie la instalación haciendo clic en "Install".



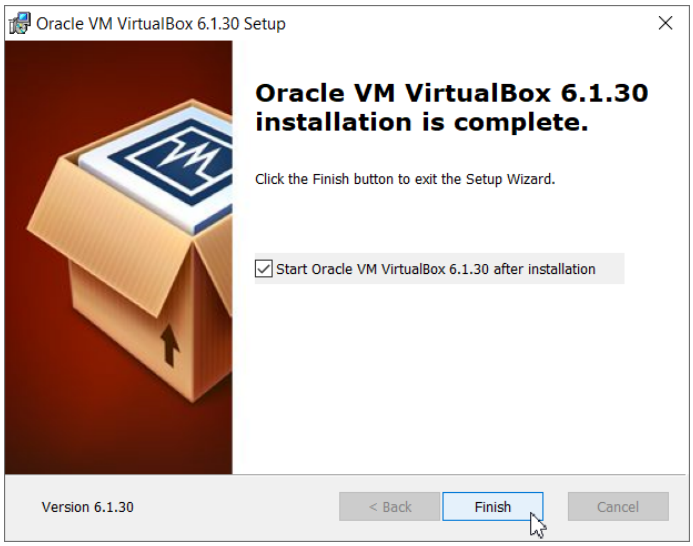
9. Acepte la notificación de Windows acerca de hacer cambios a su computador.
10. Acepte la instalación de dispositivo "Universal Serial Bus …" haciendo clic en "Instalar".



En este punto la instalación debe mostrar la copia de archivos:



11. La instalación ha finalizado. Verifique que la casilla “Start Oracle VirtualBox …” esté marcada y luego termine la instalación haciendo clic en “Finish”.

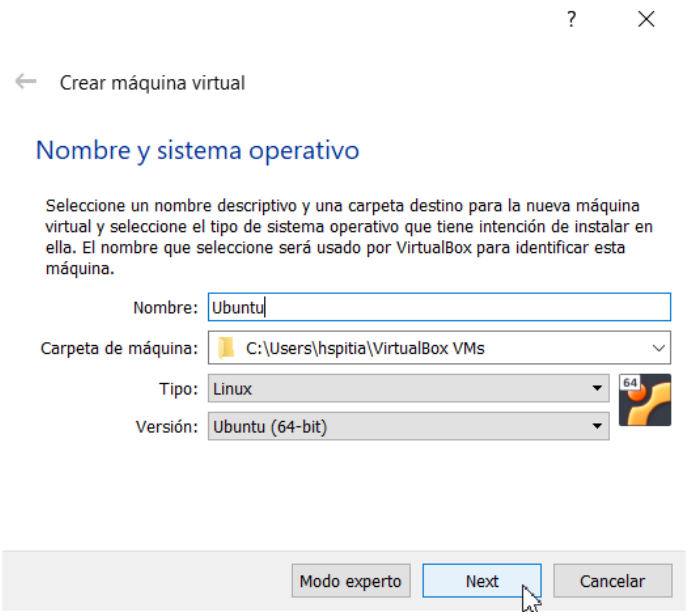


12. Ahora VirtualBox debería iniciar y usted debería ver una ventana como esta:

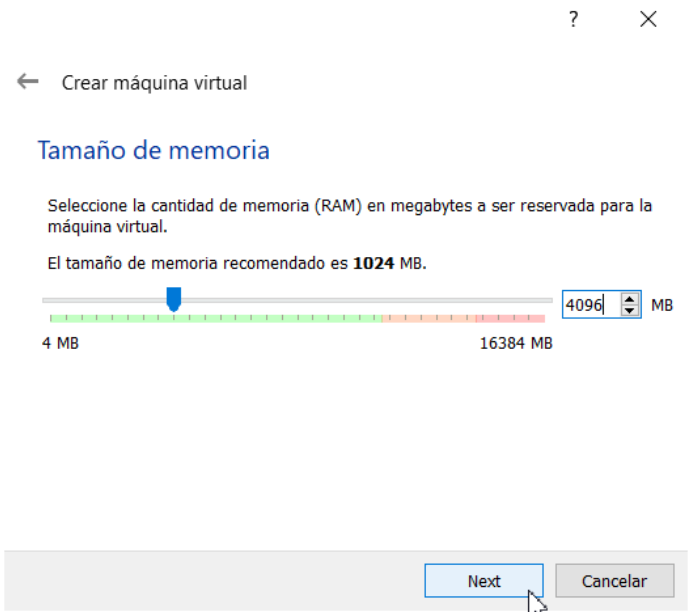
A.3. Creación de la máquina virtual Linux

13. Haga clic en el botón “Nueva” del menú derecho.

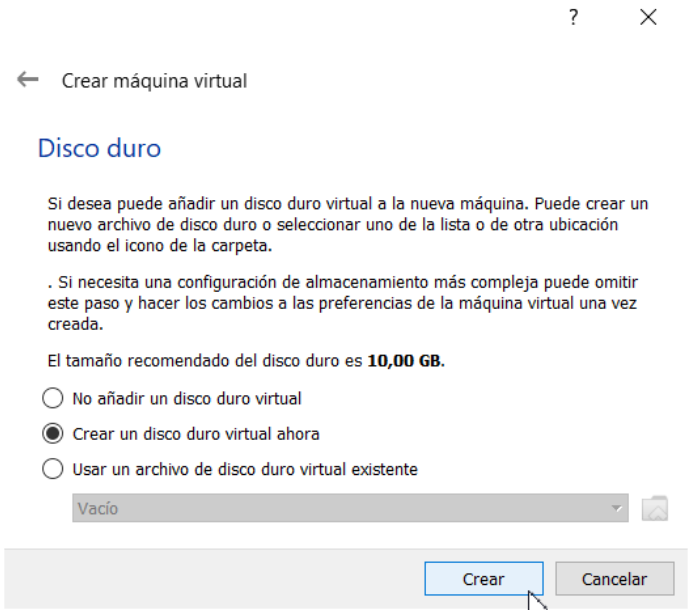
14. Escriba el nombre para máquina virtual, en este caso “Ubuntu”, y haga clic en “Next”.



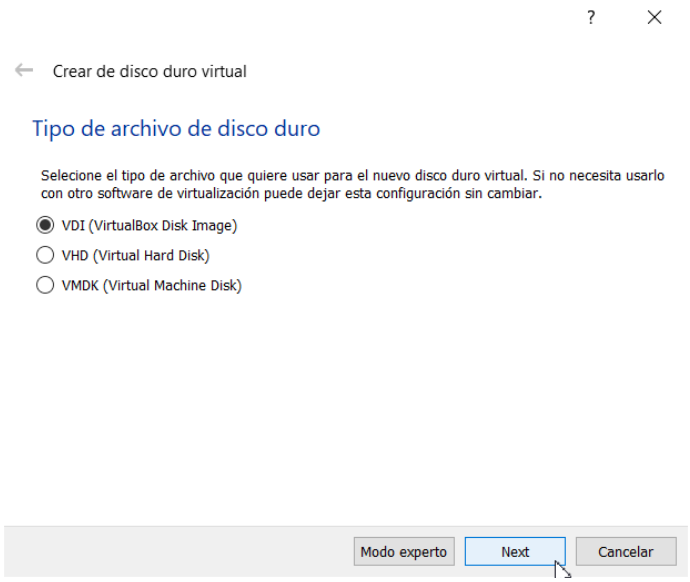
15. Asigne la cantidad de memoria (RAM) para la máquina virtual. La RAM de su computador va a ser compartida entre su equipo físico y la máquina virtual, por eso e importante asignar una cantidad que deje suficiente RAM para su computador físico y su SO Windows. Dependiendo de la RAM de su computador, VirtualBox le recomendará una cantidad mínima (posiblemente 2GB o 2048 MB). Se recomienda asignar como máximo la mitad de la RAM disponible, por ejemplo si su computador tiene 8GB (8192 MB) de RAM, asigne como máximo 4GB (4096 MB). En el este caso se ha elegido 4096 MB (4GB) la cual es una cantidad aceptable para trabajar:



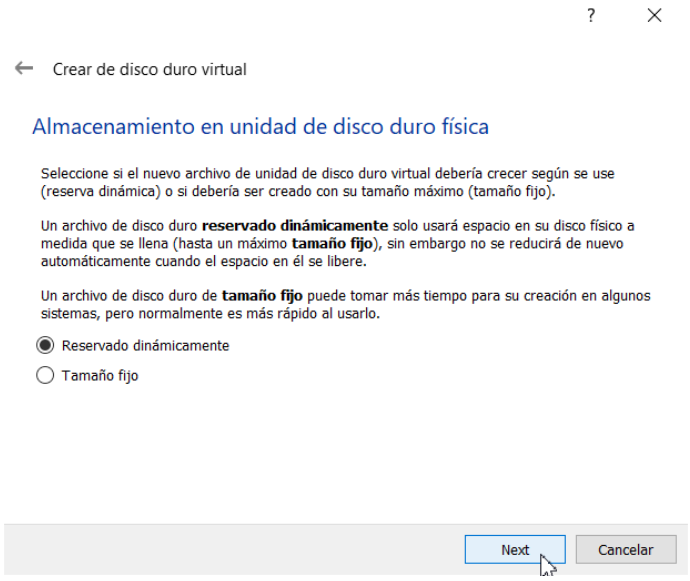
16. Creación del disco duro virtual. Asegúrese de que la opción "Crear un disco duro virtual ahora" esté marcada y después haga clic en "Crear".



17. Seleccione la opción "VDI (VirtualBox Disk Image)" para el tipo de disco a ser creado y haga clic en "Next".

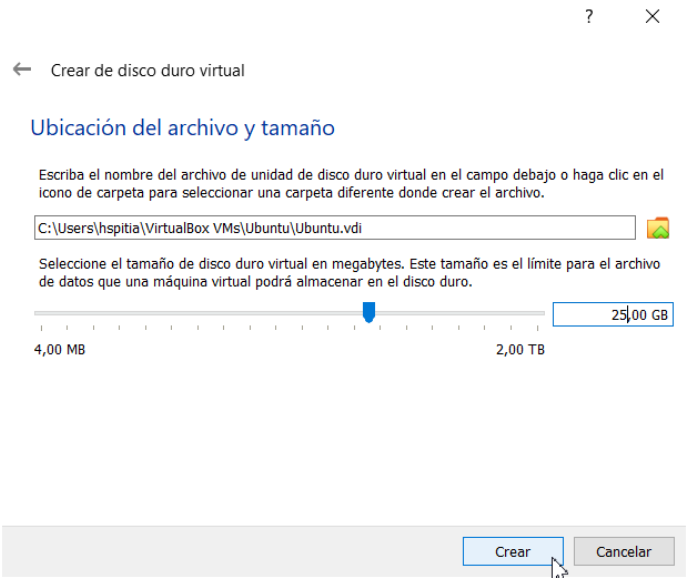


18. Seleccione la opción "Reservado dinámicamente" para el tipo de almacenamiento en su disco duro. Con esta opción el disco duro virtual crecerá en tamaño a medida que se almacenen archivos en él (dinámicamente). El espacio en todo caso crecerá de manera dinámica hasta el máximo que usted especifique después. Haga clic en "Next".



19. El archivo de disco duro virtual estará ubicado en el sitio que se muestra en pantalla, en este caso en `C:\Users\hspitia\VirtualBox VMs\Ubuntu\Ubuntu.vdi`. Conserve

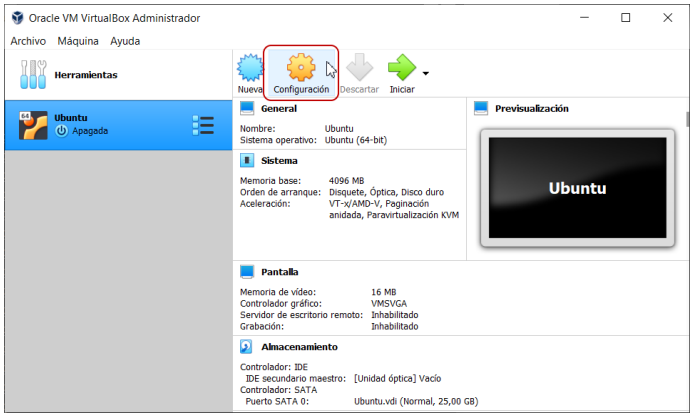
la ubicación por defecto, y asigne el tamaño máximo que tendrá el disco duro virtual. En este caso se asignó 25 GB. Se recomienda un tamaño de 20 GB como mínimo. Una vez asignado el tamaño haga clic en "Crear".



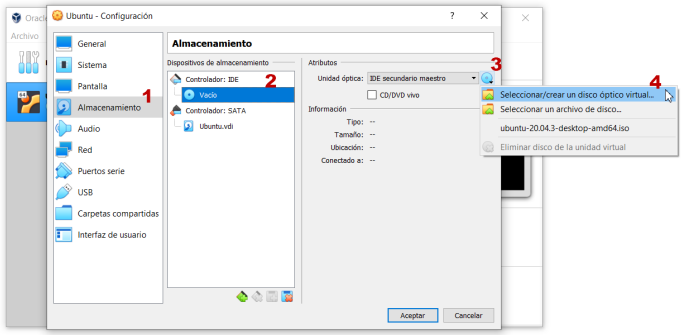
A.4. Instalación de Ubuntu Linux

Las instrucciones que encontrará a continuación son específicas de la versión de Ubuntu Linux 20.04. Si usted va a instalar un versión diferente seguramente encontrará algunas diferencias en el proceso. En todo caso, la instalación es bastante parecida entre versiones.

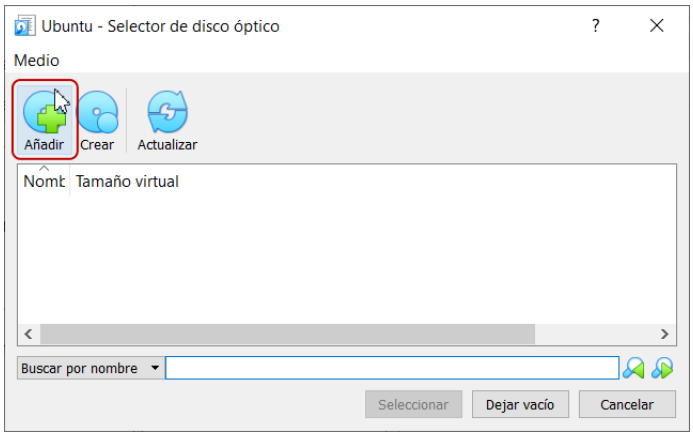
20. Ahora que la máquina virtual está creada y se puede instalar Ubuntu en ella. Haga clic en el botón "Configuración" del menú derecho:



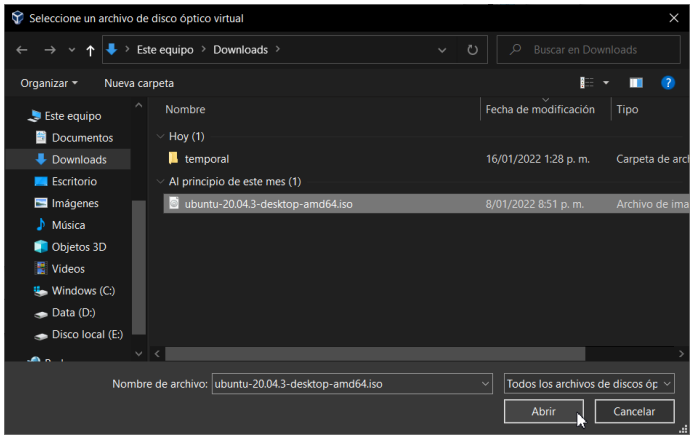
21. En la nueva ventana, seleccione “Almacenamiento”, luego “Vacío” bajo “Controlador: IDE”, y haga clic en el ícono del disco azul al final de “Unidad óptica”. Ahora haga clic en “Seleccionar/crear un disco óptico virtual…”.



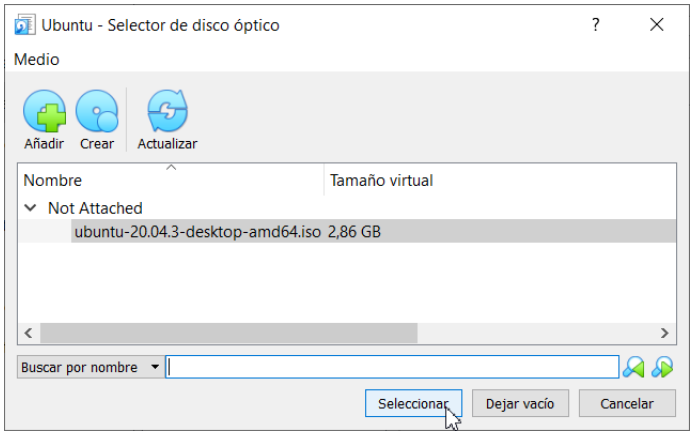
22. Haga clic en “Añadir”



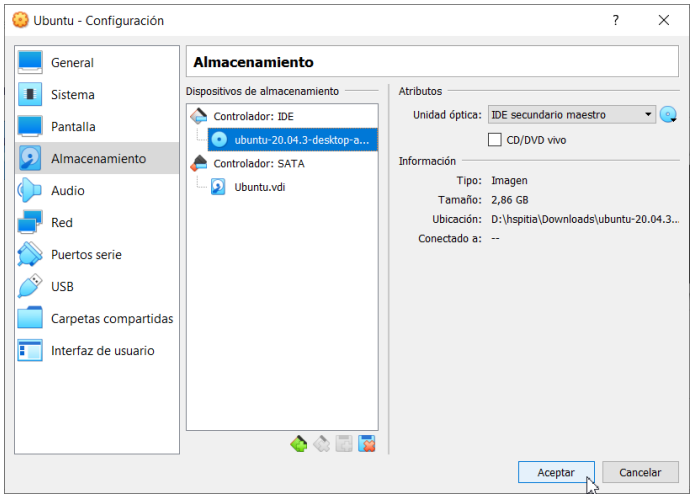
23. Busque y seleccione el archivo de imagen de Ubuntu Linux (.iso) que descargó en el paso 1, y haga clic en Abrir. Esto dos últimos pasos simulan la inserción del disco de instalación de Ubuntu Linux en el lector de discos (CD/DVD) de la máquina virtual.



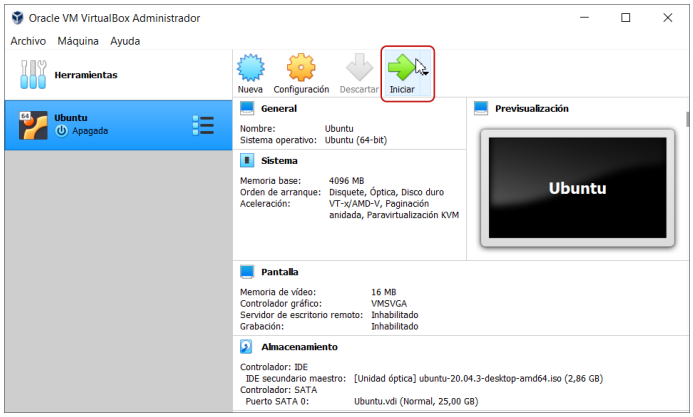
24. Haga clic en “Seleccionar”.



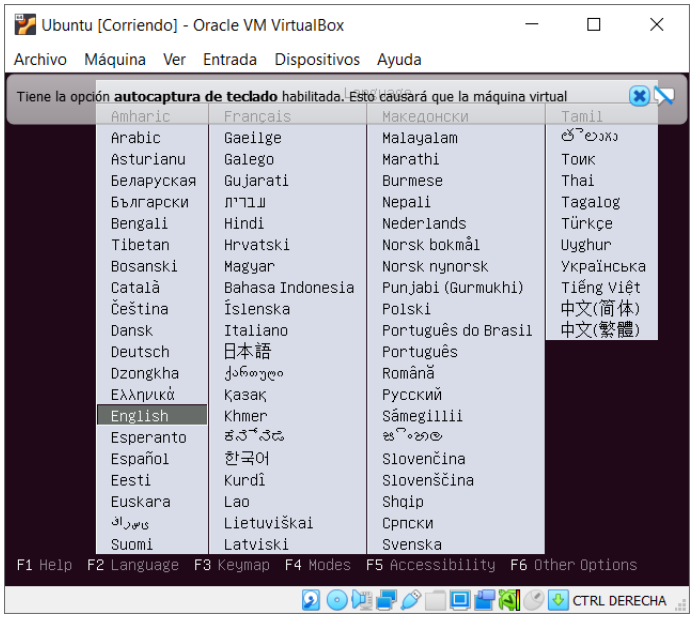
25. Haga clic en “Aceptar”.



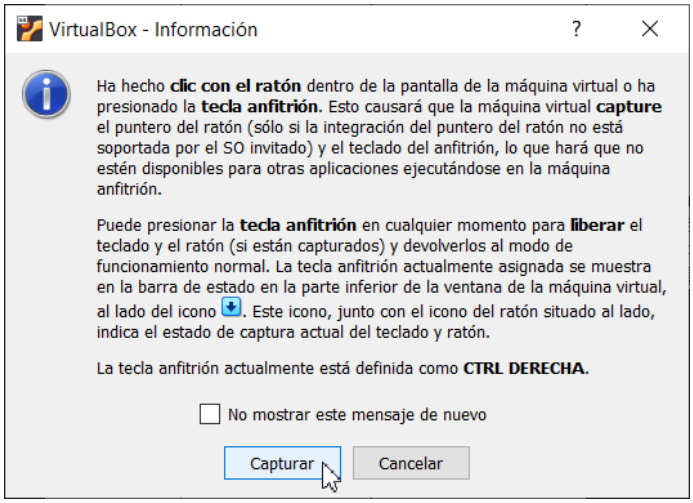
26. Ahora inicie la máquina virtual haciendo clic en el botón “Iniciar”.



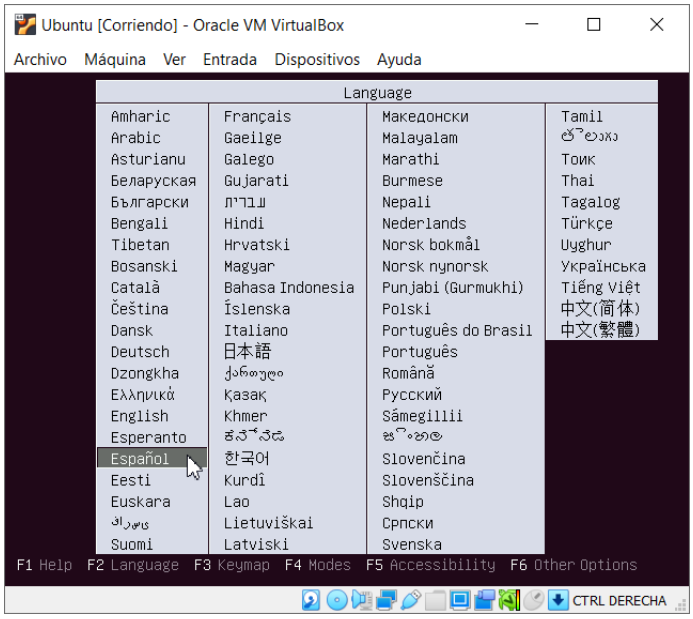
27. Ahora verá una pantalla que muestra diferentes opciones de lenguaje para la instalación de Ubuntu Linux. Haga clic para seleccionar la opción.



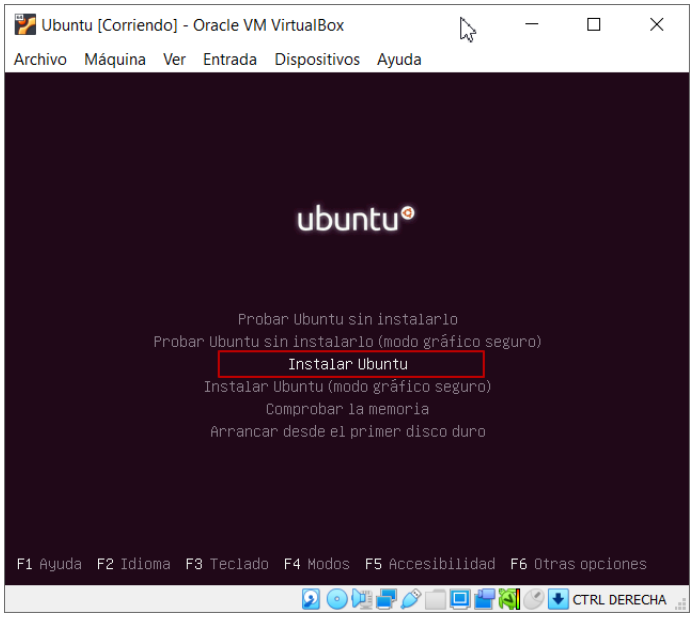
28. Seguramente aparecerá una ventana de notificación luego de hacer clic en el lenguaje. Esta ventana le notifica que su mouse ha sido "capturado" por la máquina virtual, lo que significa que tiene exclusividad sobre él. Lea cuidadosamente la información y haga clic en "Capturar" para aceptar.



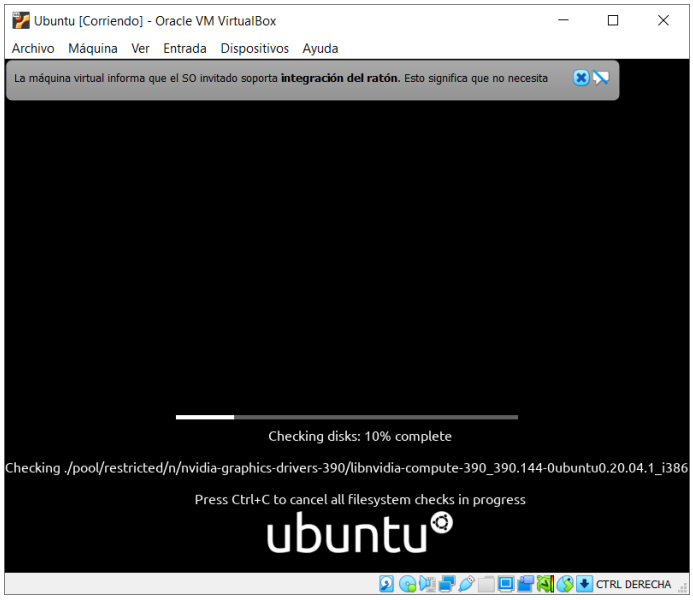
29. Seleccione "Español" y presione la tecla "Enter" en su teclado.



30. Ahora seleccione “Instalar Ubuntu” en la pantalla de opciones.

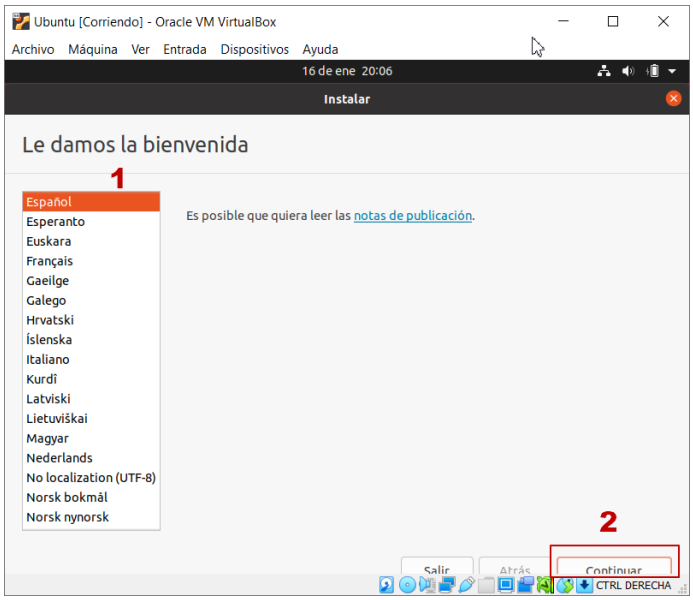


31. El instalador deberá empezar a comprobar la integridad de la imagen de instalación.

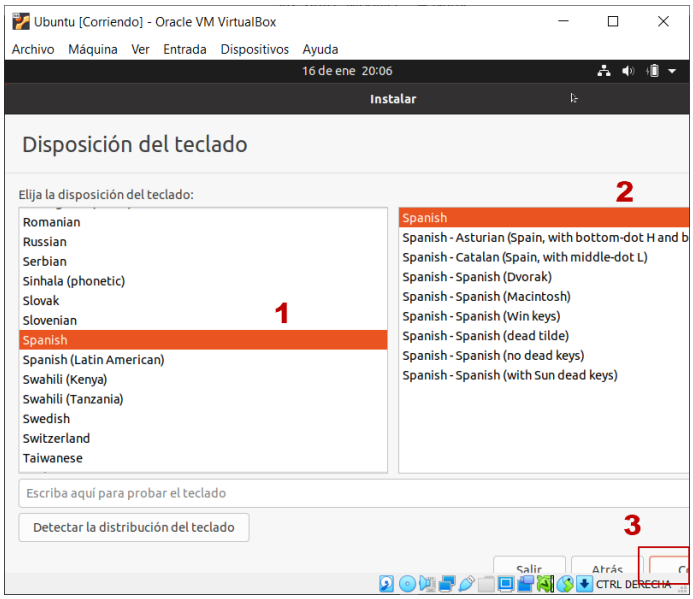


32. En este punto notará que las ventanas de instalación de Ubuntu no se pueden visualizar de manera completa. Esto sucede porque faltan algunos controladores de video en la máquina virtual. Esta situación se corregirá después de terminar la instalación de Ubuntu.

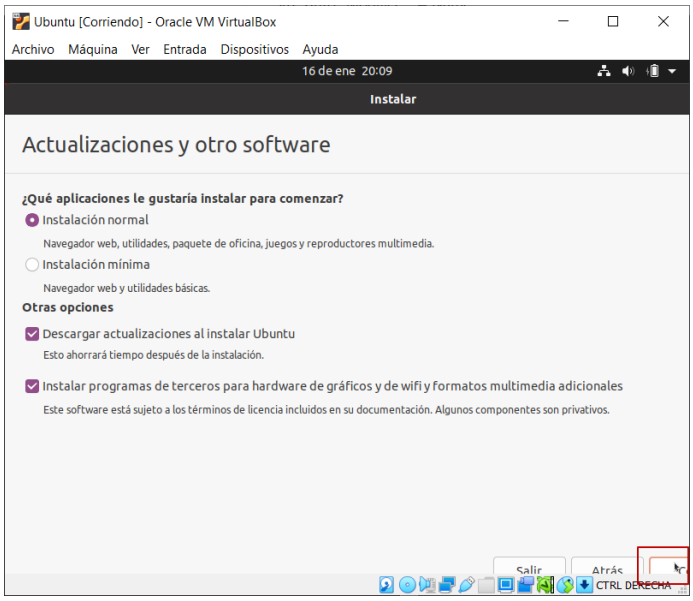
    Seleccione ahora Español y haga clic en “Continuar”.



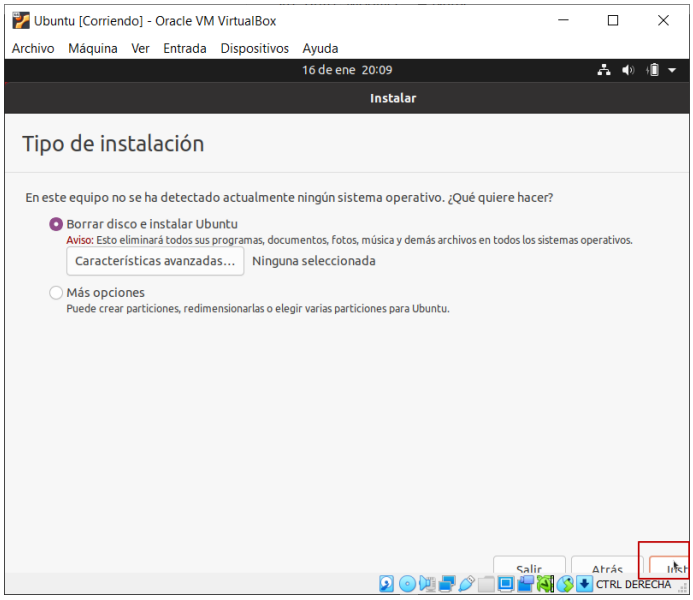
33. Verifique que “Spanish” está seleccionado en ambos cuadros de opciones y haga clic en “Continuar”.



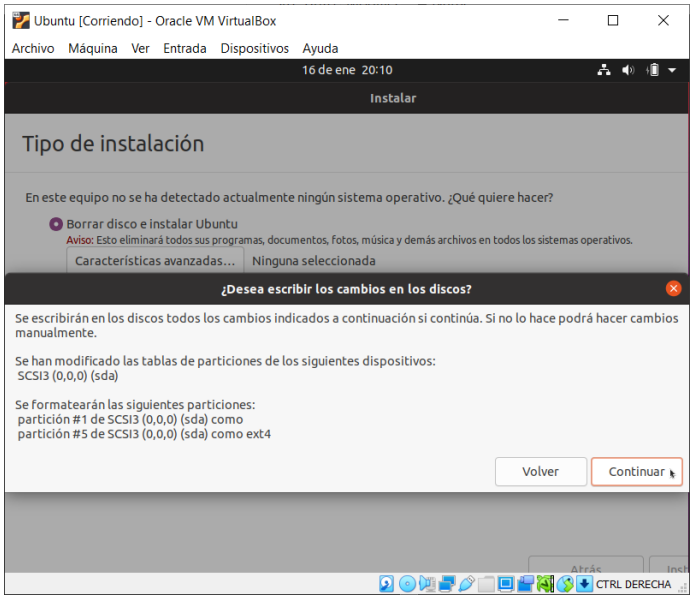
34. Asegúrese de marcar las opciones que se muestran en la imagen y haga clic en “Continuar”.



35. Verifique que la opción “Borrar disco e instalar Ubuntu” esté marcada y haga clic en “Instalar”.



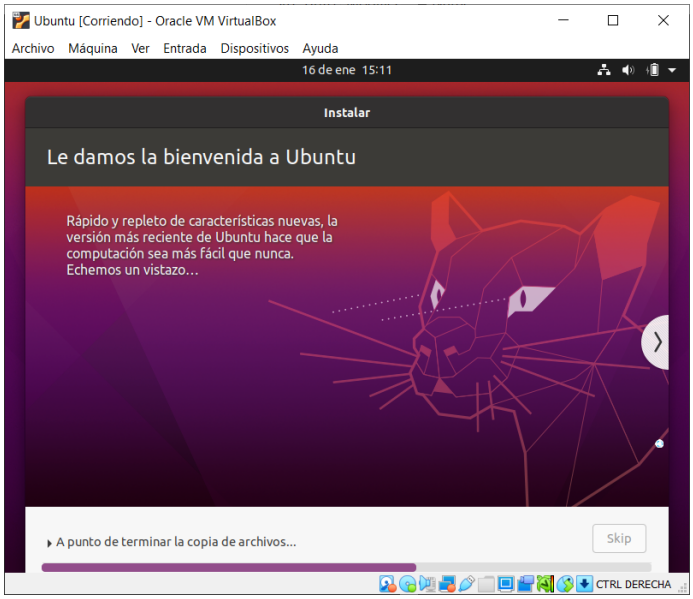
36. En la nueva ventana, haga clic en “Continuar”.

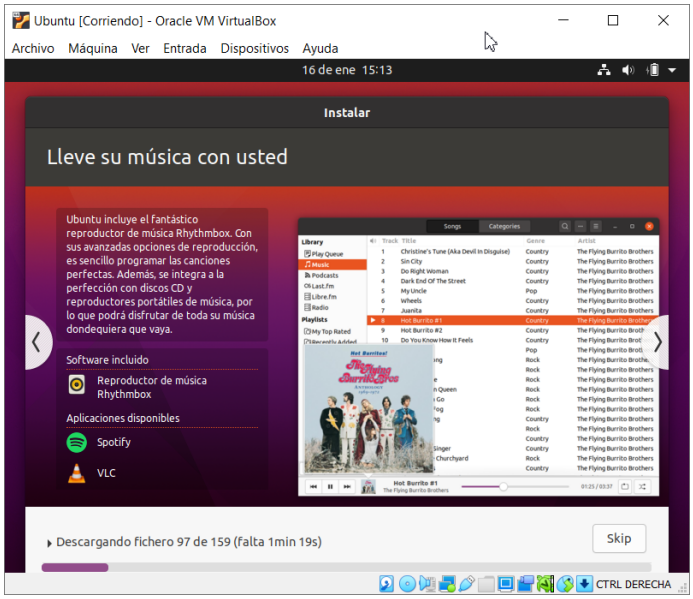


37. Seleccione la zona horaria adecuada y haga clic en “Continuar”.

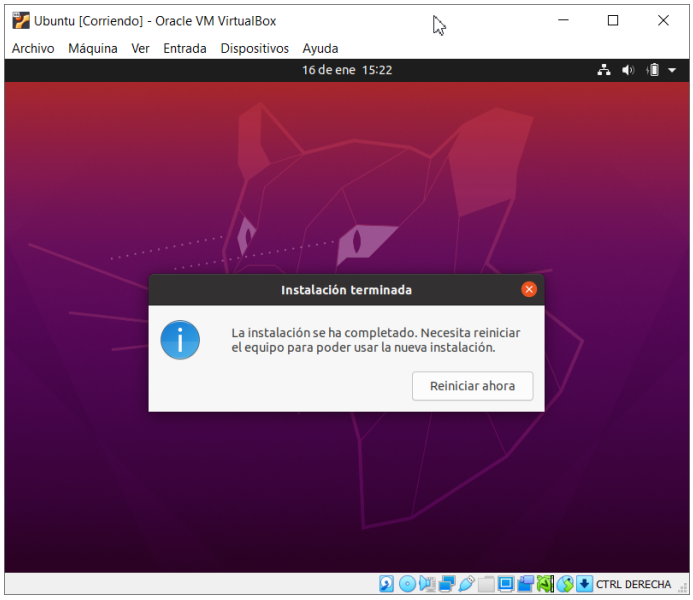
38. Entre el nombre, nombre del equipo, nombre de usuario, y elija una contraseña. Verifique que la opción "Solicitar mi contraseña para iniciar sesión" esté marcada y haga clic en "Continuar".

39. Ahora el proceso de instalación debe continuar con la copia de archivos. Este proceso debería durar unos 20 minutos aproximadamente.

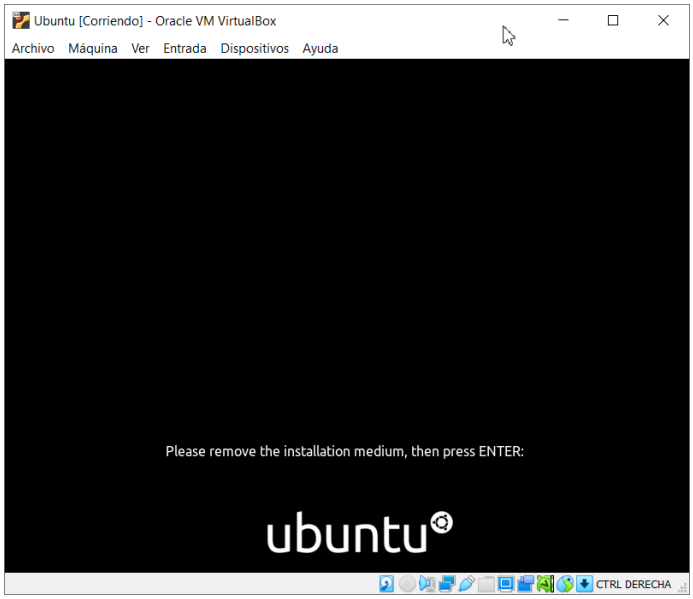




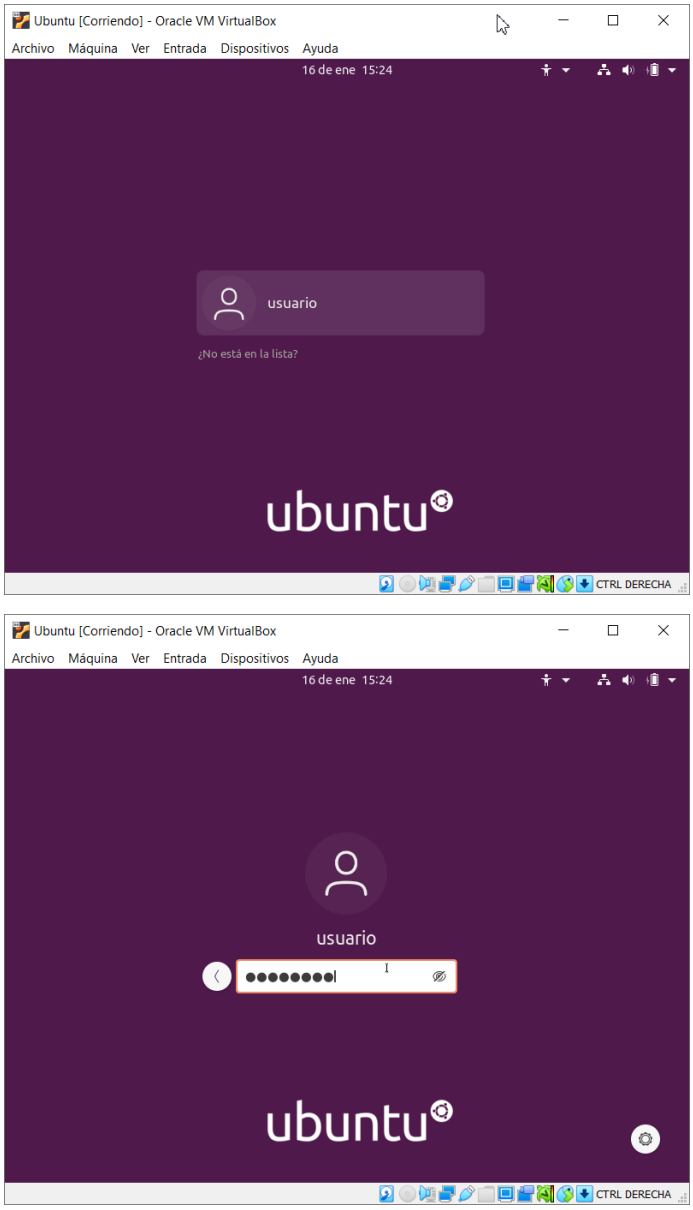
40. Ahora el instalador le notificará que la instalación ha terminado. Finalice haciendo clic en “Reiniciar ahora”.



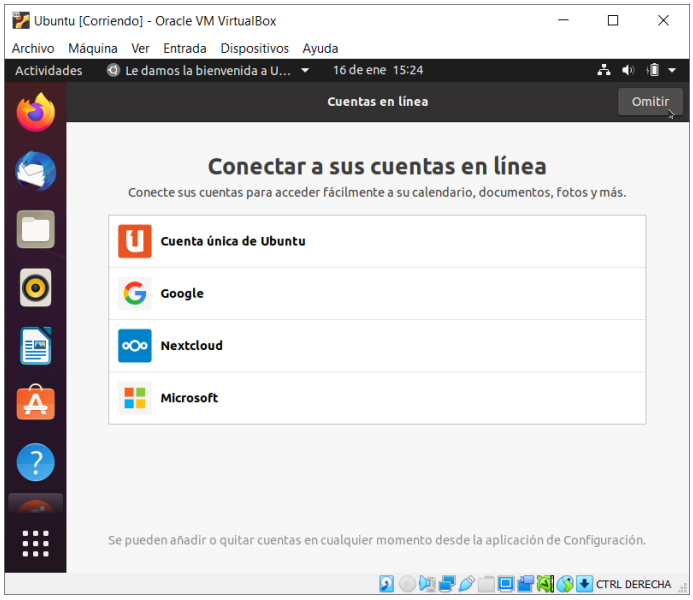
41. Posiblemente vea una pantalla que solicita retirar el medio de instalación. Presione la tecla “Enter” para continuar.



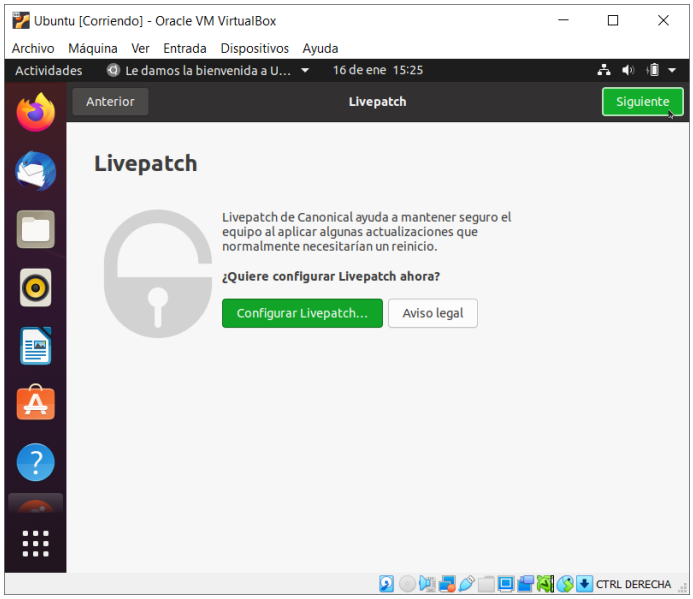
42. Ahora la máquina virtual ha reiniciado y requiere que entre su contraseña para empezar sesión en Ubuntu.



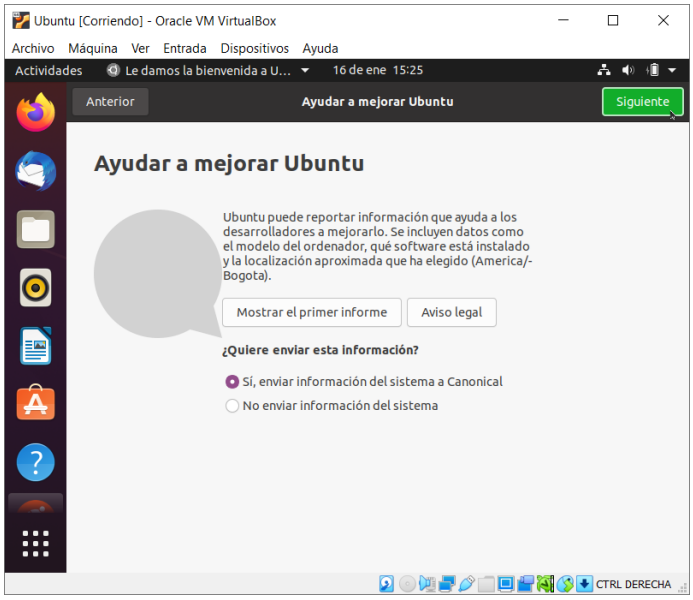
43. Ahora verá una ventana para la configuración de algunas características de Ubuntu. Para empezar, si lo desea puede conectar sus cuentas a Ubuntu. En este caso se ha elegido obviar este paso haciendo clic en “Omitir”.



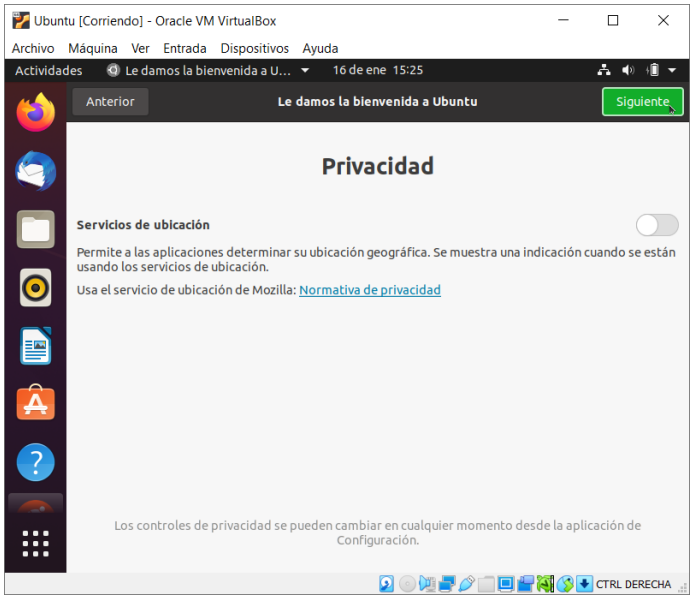
44. Si lo desea puede configurar la actualización del sistema en segundo plana llamado *Livepatch*. Si elige esta opción deberá crear una cuenta en Ubuntu. En este caso este paso se ha omitido haciendo clic en "Siguiente".



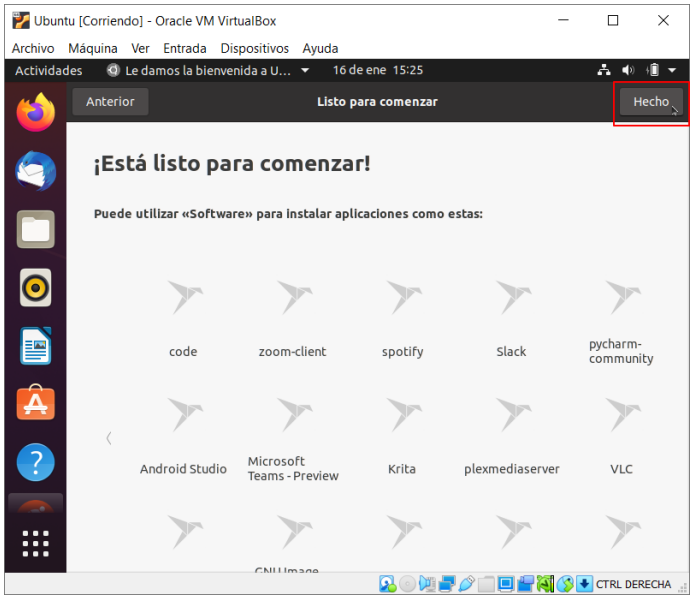
45. Haga clic en "Siguiente".



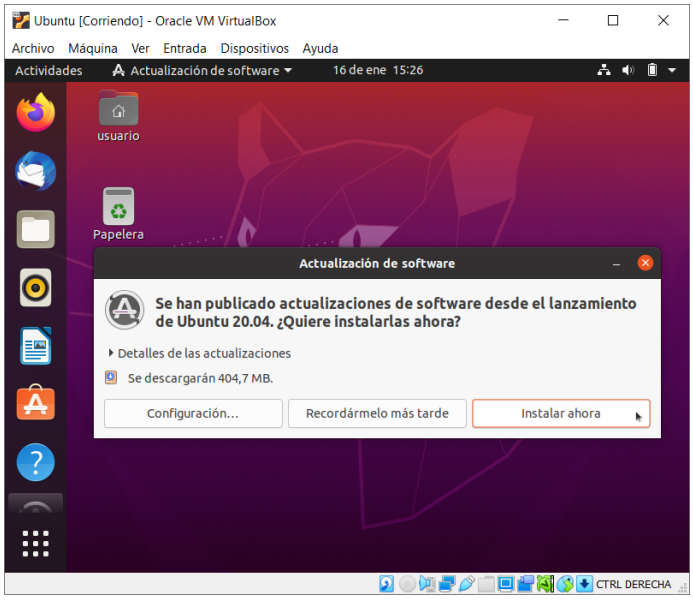
46. Haga clic en “Siguiente”.



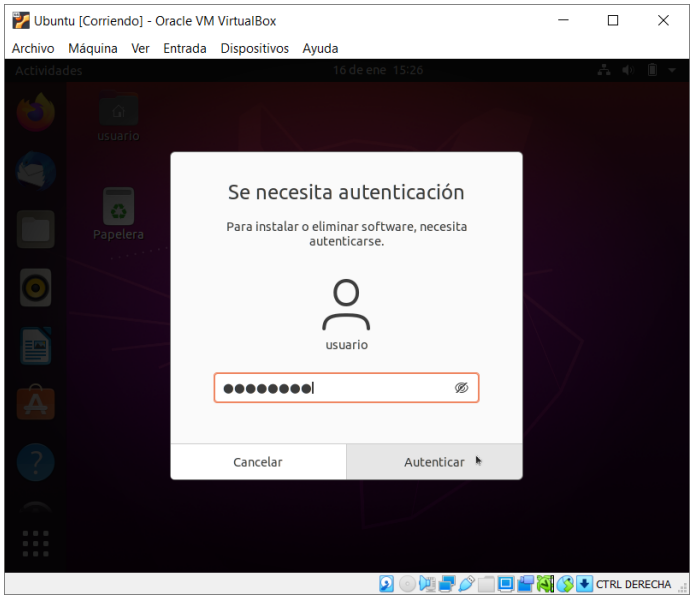
47. Ahora puede finalizar haciendo clic en “Hecho”.



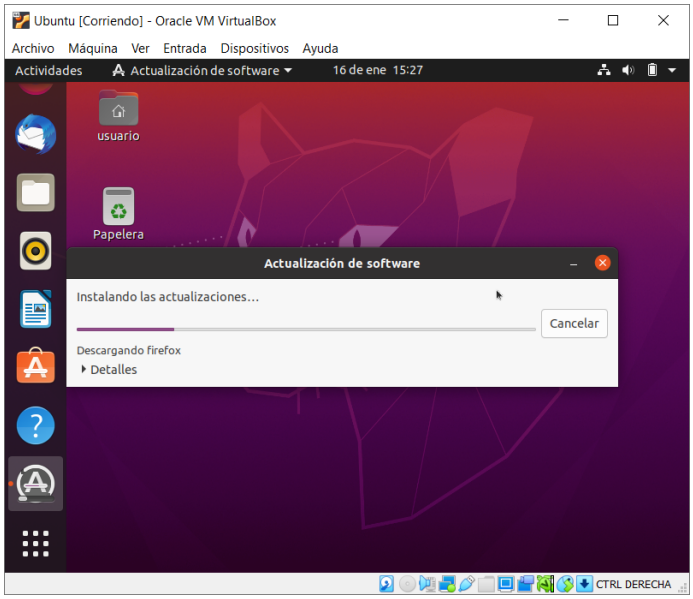
48. Seguramente verá ahora una ventana de instalación de actualizaciones. Aplíquelas haciendo clic en “Instalar ahora”.



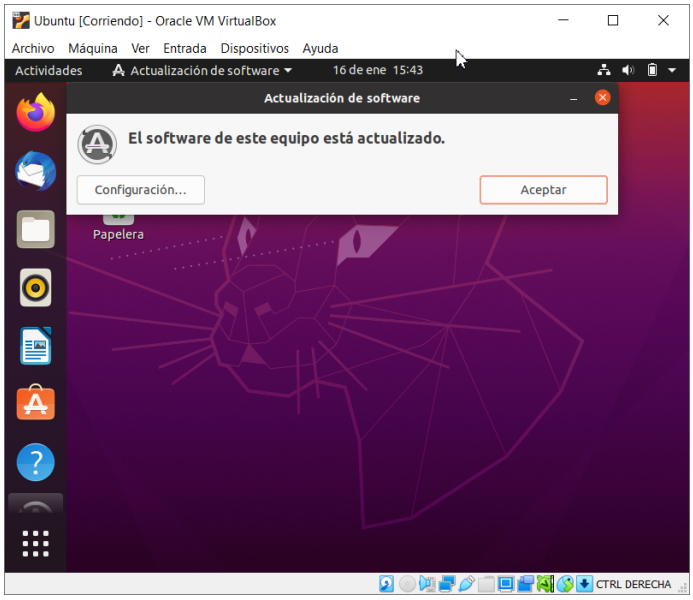
49. Digite su contraseña para empezar a instalar las actualizaciones.



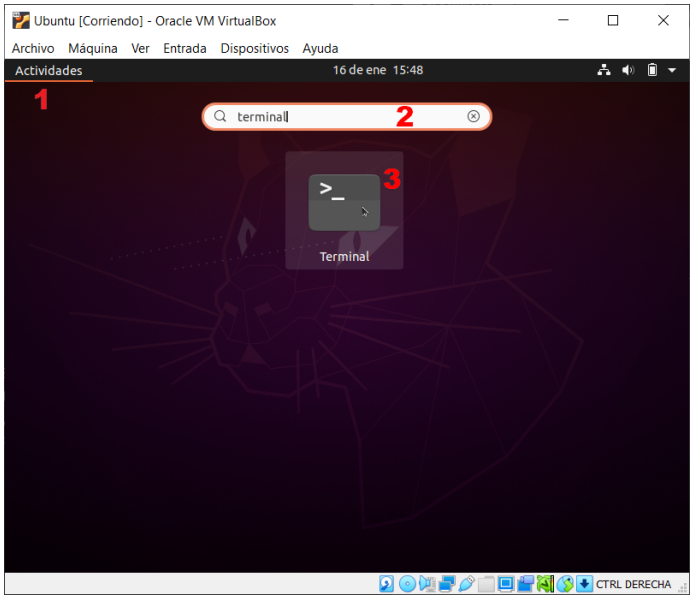
50. Dependiendo de la velocidad de su conexión a Internet, el proceso de actualización puede tomar unos 20 minutos aproximadamente.



51. Ahora que la actualización ha terminado, haga clic en “Aceptar” para finalizar.



52. Ahora se procederá a instalar los controladores de video para corregir la visualización de la pantalla de Ubuntu. Para ello, empiece haciendo clic en "Actividades" en la esquina superior izquierda y a continuación digite la palabra "terminal". Haga clic en el ícono "Terminal" que aparece debajo del cuadro de texto.

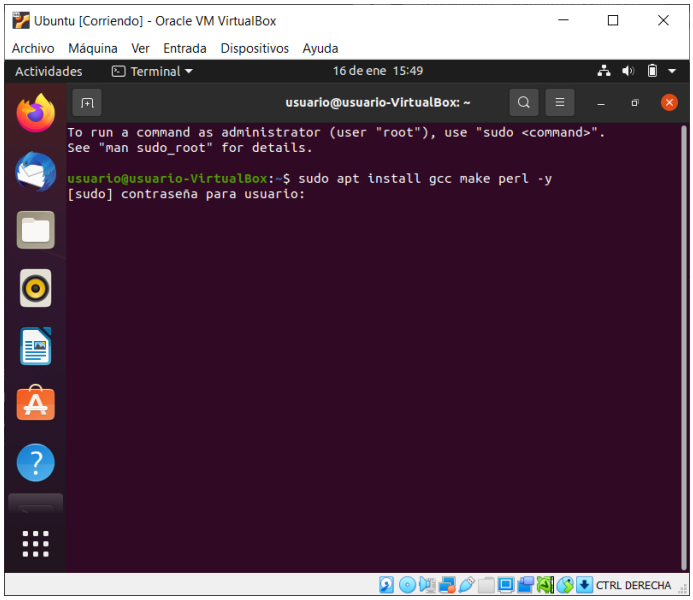


53. Ahora verá la Terminal o consola que permite la introducción de comandos. Antes de instalar los controladores de video, es necesario instalar tres paquetes. Para ello,

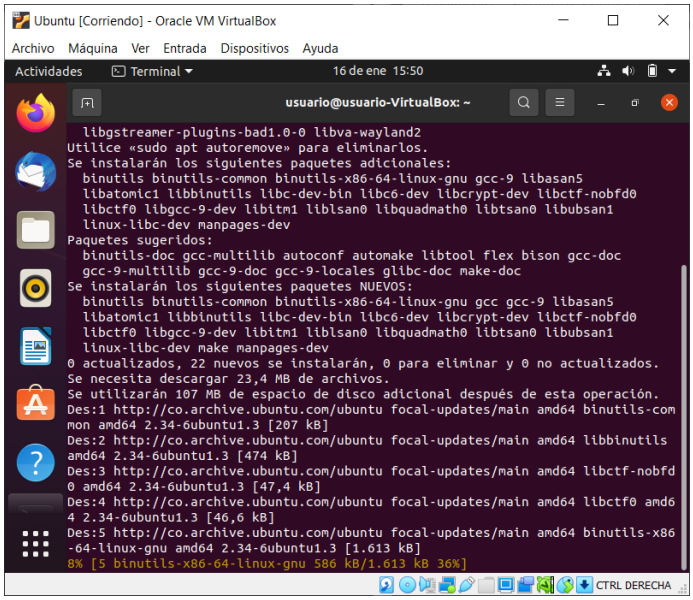
escriba el siguiente comando y presione “Enter”:

```
sudo apt install gcc make perl -y
```

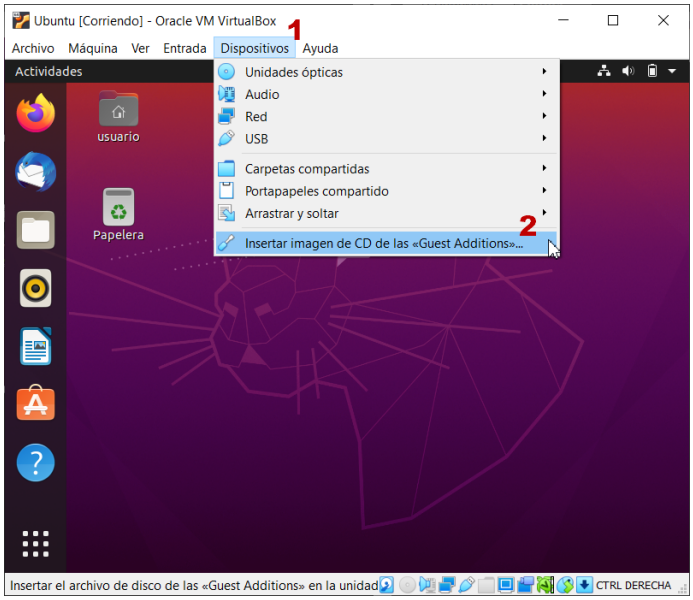
Digite su contraseña para aceptar la instalación de los paquetes:



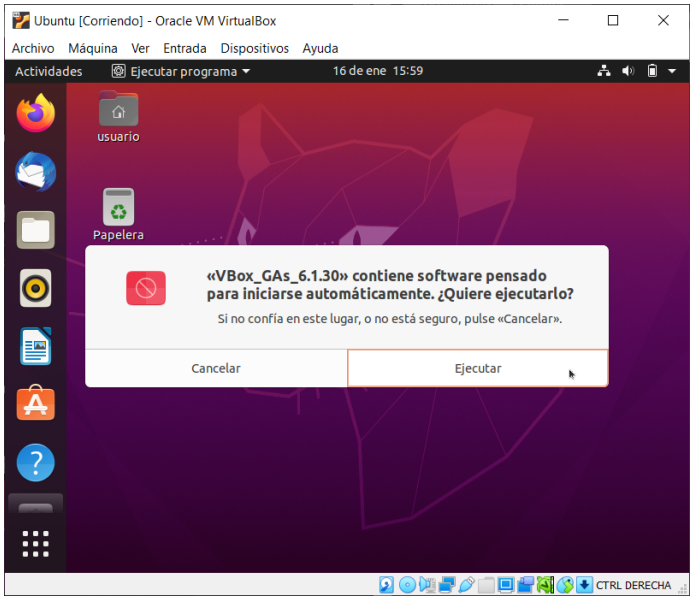
54. El proceso toma al rededor de un minuto. Una vez finalice la instalación, puede cerrar la ventana de la Terminal.



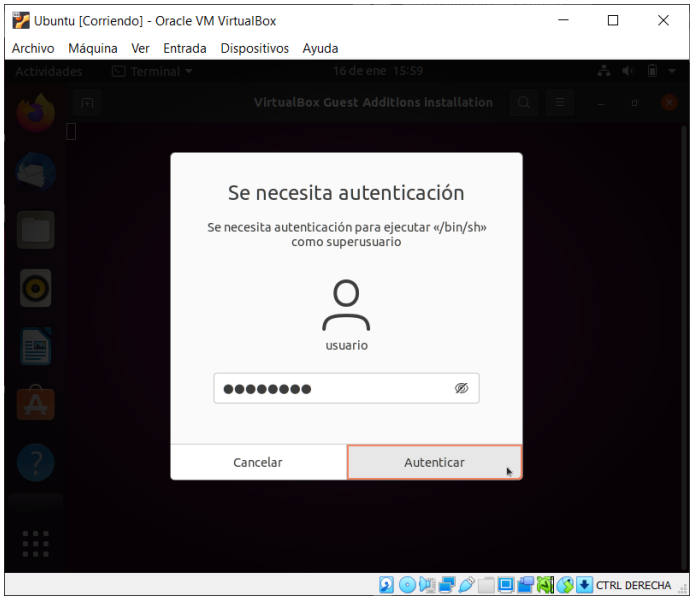
55. Ahora se puede instalar los controlador de video. En el menú de la ventana VirtualBox, haga clic en la opción "Dispositivos" y luego en "Insertar imagen de CD de las «Guest Additions…»". Esto simula la inserción de un CD en la unidad óptica de la máquina virtual.



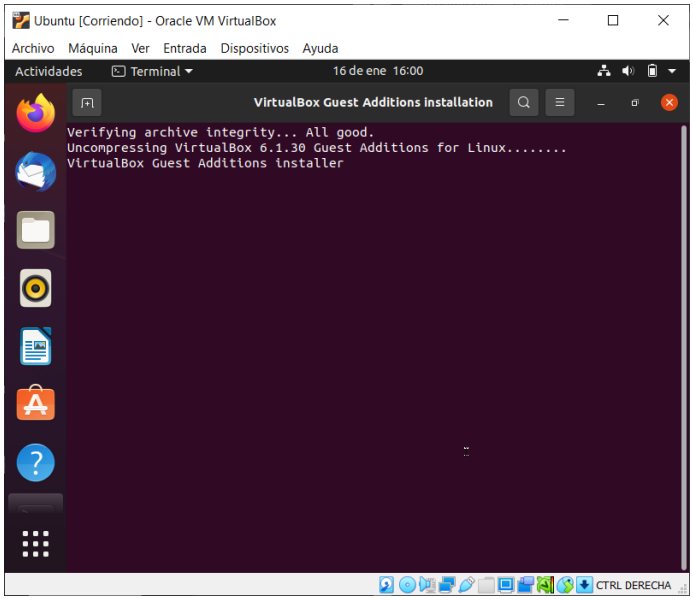
56. Ahora verá un ventana de notificación de ejecución automática del programa en el CD insertado. Ejecute este programa haciendo clic en "Ejecutar".



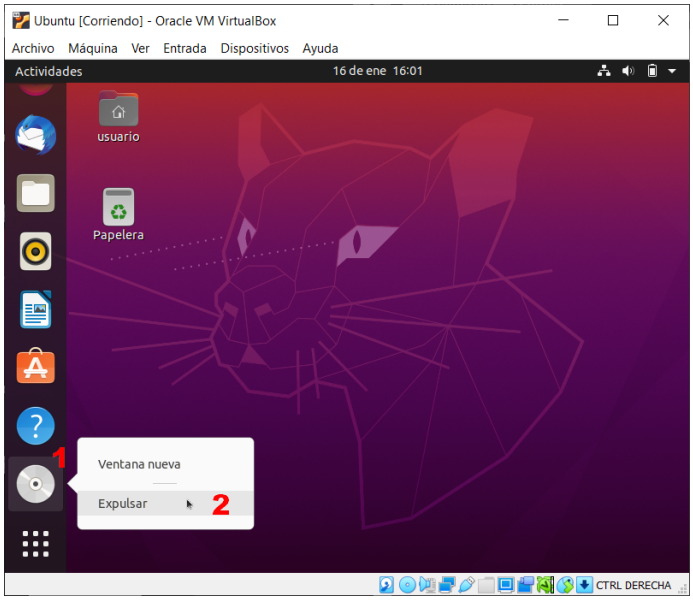
57. Digite su contraseña para iniciar la instalación de las “Guest additions”



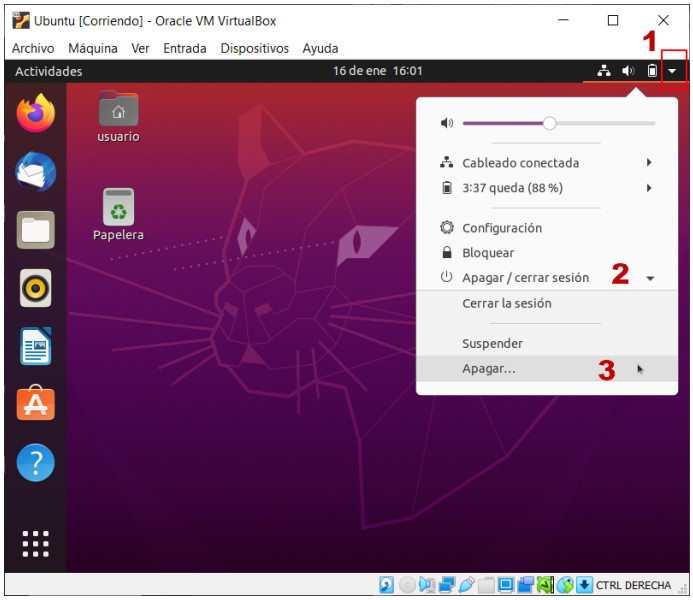
58. El proceso de instalación debería durar solo unos segundos. Cunado finalice, puede cerrar la ventana de la Terminal que se abrió automáticamente.



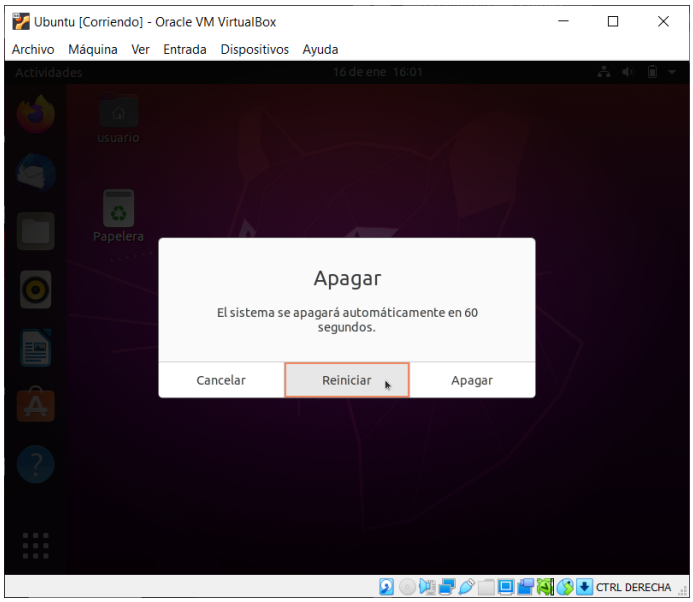
59. Ahora, en la barra izquierda de Ubuntu (llamada "Dock Panel"), ubique el ícono del CD de las "Guest additions" y haga clic derecho en él. Finalmente haga clic en "Expulsar".



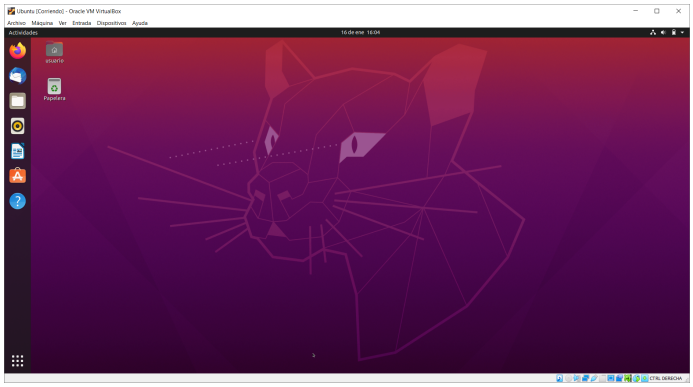
60. Ahora se puede proceder al reinicio de la máquina virtual. Vaya a la barra superior, y despliegue las opciones haciendo clic en el ícono del triángulo invertido. Haga clic en "Apagar / cerrar sesión", y a continuación haga clic en "Apagar…".



61. Ahora reinicie la máquina haciendo clic en “Reiniciar”.



62. Una vez termine el reinicio de la máquina virtual, aumente el tamaño de la pantalla de VirtualBox, notará que el escritorio de Ubuntu se ajusta al nuevo tamaño de pantalla y podrá visualizar toda la interfaz de manera completa.



63. ¡Felicitaciones! ha terminado la instalación de Ubuntu Linux correctamente.

Universidad de **Nariño**
FUNDADA EN 1904

Universidad de **Nariño**
ACREDITADA DE ALTA CALIDAD
RESOLUCIÓN MEN 10567 - MAYO 23 DE 2017

CESUN
CENTRO DE ESTUDIOS EN SALUD
Universidad de **Nariño**