Uso de máquinas virtuales y shell de comandos © Ediciones Roble S. L.

Indice

Uso de máquinas virtuales y shell de comandos

I. Introducción

Una de las principales dificultades que surge en el ámbito del desarrollo de software es la incompatibilidad de las herramientas que se van a utilizar con respecto a los diferentes sistemas operativos. Así, en muchas ocasiones, el software no es multiplataforma, es decir, se ha creado para ejecutarse en un sistema operativo concreto. Esto produce problemas si el sistema operativo sobre el que se va a trabajar no coincide con el que es necesario para el software que se quiere utilizar, por lo que en estos casos sería necesario instalar el sistema operativo requerido por el software.

Otro problema más particular, y ligado al ámbito del análisis de datos y del Big Data, es la complejidad de la instalación y configuración de las herramientas informáticas que se utilizan. En muchos casos, el proceso de instalar y configurar la herramienta con los parámetros adecuados requiere de unos conocimientos de un perfil administrador muy diferente al del usuario de la herramienta concreta. A estas dificultades habría que añadir el tiempo requerido para realizar estas operaciones.

En esta unidad del módulo se van a estudiar las máquinas virtuales como solución a estos problemas. Más concretamente, se analizará la herramienta **Virtual Box** para crear y gestionar máquinas virtuales. Asimismo, en esta unidad se realizará una introducción sobre el uso de la Shell de comandos de Linux, ya que muchas de las herramientas que se utilizan en el ámbito del Big Data se ejecutan sobre entornos de tipo Linux y la manera de interactuar con dichas herramientas es a través del uso de comandos de la Shell.

La unidad se estructura de la siguiente forma. En primer lugar, se realizará una introducción al concepto de máquina virtual y su necesidad de uso. A continuación, se describirá cómo instalar Virtual Box y cómo se puede crear y configurar de manera básica una máquina virtual. En la siguiente sección se mostrará cómo cargar y utilizar una máquina virtual previamente creada. La última sección de la unidad se centra en revisar el uso de los comandos básicos de la Shell de Linux y la creación de guiones que permitirán automatizar tareas y desarrollar habilidades para gestionar el sistema operativo.

Para obtener el máximo aprendizaje de la unidad, es recomendable descargar Oracle VM VirtualBox y realizar la instalación y configuración de una máquina virtual (Linux). Sirviéndose de esta instalación, es importante desarrollar los guiones del último capítulo de la unidad.

II. Objetivos

Los objetivos que los alumnos alcanzarán tras el estudio de esta unidad son:

Entender el concepto y la necesidad del uso de máquinas virtuales.

Saber instalar la herramienta Virtual Box.

Saber crear y configurar de manera básica una máquina virtual en Virtual Box.

Saber cómo se carga una máquina virtual dada en Virtual Box.

Conocer y saber utilizar de manera básica la Shell de comandos de Linux.

III. Concepto de máquina virtual. Virtual Box

La existencia de diferentes sistemas operativos acarrea, como una de las principales desventajas en el ámbito del desarrollo de software, la necesidad de tener que crear para un mismo software diferentes versiones adecuadas para cada uno de los sistemas operativos. Sin embargo, esto no ocurre siempre, ya que la creación de una versión del mismo software para diferentes plataformas supone un esfuerzo económico para las empresas que no siempre es rentable. Esta situación produce problemas en el momento que se quiere utilizar un determinado software que solo se ejecuta para un determinado sistema operativo que no coincide con el que dispone el usuario.

Soluciones anteriores

Antes de la aparición de las máquinas virtuales, la única solución posible consistía en instalar en la misma máquina los sistemas operativos que eran necesarios para el software que se iba a utilizar, compartiendo de esta manera el disco duro entre ellos. Se trata de una solución costosa tanto en tiempo —el necesario para instalar y poner a punto el sistema operativo correspondiente—, como en la cantidad de memoria del disco que deben compartir los diferentes sistemas operativos instalados.

Aparición de máquinas virtuales

En este sentido, las máquinas virtuales aparecen como una solución a la necesidad de tener que usar diferentes sistemas operativos en una misma máquina. La idea que sostiene el funcionamiento de las máquinas virtuales consiste en usar el sistema operativo instalado por defecto en la máquina, denominado sistema anfitrión, para ejecutar una simulación de otro sistema operativo denominado sistema huésped. El sistema huésped se ejecuta como si fuera un programa más y se puede utilizar como si realmente estuviera instalado en la máquina —aunque realmente no se encuentra instalado y lo único que se está ejecutando es una simulación—.

Para poder crear y ejecutar una máquina virtual, es necesario un software especial encargado de llevar a cabo la simulación. Actualmente, las herramientas más utilizadas para la creación y gestión de máquinas virtuales son Virtual Box y VMware.

VMware

La herramienta VMware dispone de una versión gratuita con las características y funciones necesarias para un usuario particular, tales como la creación de una máquina virtual, y una versión de pago orientada a empresas que añade funciones y características más avanzadas tales como la posibilidad de virtualizar sistemas operativos y gestionarlos a través de la red como si se tratase de una nube.

Virtual Box

Por otro lado, Virtual Box es una herramienta de código abierto, totalmente gratuita y disponible para cualquier sistema operativo, propiedad de Oracle. No ofrece las mismas características avanzadas orientadas hacia las empresas de VMware, pero sí dispone de todas las funciones necesarias para crear y ejecutar máquinas virtuales. E incluso dispone de algunas funcionalidades propias como la capacidad de ejecutar varias máquinas virtuales a la vez.

Con respecto a la eficiencia y rendimiento, VMware presenta un rendimiento mejor que Virtual Box en el uso de la memoria y de la CPU. Sin embargo, con respecto al rendimiento del disco duro ambas herramientas son similares.

Indicaciones de uso

Cuando se utilizan máquinas virtuales, hay que tener en cuenta lo siguiente:

1

Todas las operaciones que se realicen sobre y desde elsistema operativo virtualizado no afectarán al sistema operativo anfitrión ni a los demás sistemas operativos huéspedes De hecho, esta es una de las ventajas de usar máquinas virtuales: cada sistema operativo virtualizado es independiente de los demás; si ocurre algún problema en el sistema operativo virtualizado, el sistema operativo anfitrión quedará resguardado y bastará con interrumpir la ejecución de la máquina virtual.

2

Cada vez que se ejecuta una máquina virtual, en el momento en que se apaga, es posible guardar el estado de la máquina virtual. Es decir, es posible mantener el sistema tal como estaba antes de apagarse, de manera que cuando se vuelva a ejecutar, el usuario podrá continuar en el mismo estado en que lo dejó.

3

En general, es posible el intercambio de documentos entre los sistemas operativos anfitrión y huésped, lo que facilita la reutilización de todo lo generado en un sistema operativo virtualizado y el uso de recursos que se encuentran en el sistema operativo anfitrión.

4

Desde un sistema operativo virtualizado se pueden utilizar recursos auxiliares como el acceso a la red de comunicación, impresoras u otros dispositivos que se encuentran conectados a la máquina sobre la que se ejecuta.

5

A cada máquina virtual se le asignan recursos dependiendo de los disponibles del sistema operativo anfitrión, tales como memoria RAM, capacidad de almacenamiento y núcleos de CPU.

CASOS DE USO

Hay dos casos de uso principales cuando se utilizan este tipo de herramientas:

Creación de máquina virtual

Crear una máquina virtual, o bien cargar una máquina virtual. El primer caso surge cuando existe la necesidad de preparar un entorno virtual para ejecutar determinado software y no es posible hacerlo en el sistema operativo instalado por defecto en la máquina.

Uso de máquina virtual existente

El otro caso de uso consiste en utilizar una máquina virtual que alguien previamente ha creado y configurado y que se desea utilizar para poder acceder a determinado software instalado en la misma. Obsérvese, con respecto a este último caso, que existen sitios web que actúan como repositorios de máquinas virtuales que han sido creadas por otros usuarios y que ponen a libre disposición de otros usuarios para que puedan descargarlas y usarlas.

Servicios en la nube

También se llama la atención sobre el hecho de que, además de las herramientas comentadas anteriormente, existe otra forma alternativa de crear y utilizar máquinas virtuales en el contexto de los denominados servicios en la nube.

La computación en la nube —también denominado *cloud computing*— es un modelo de prestación de servicios en el que los usuarios pueden acceder a servicios o recursos que se encuentran en Internet mediante el pago por el consumo efectuado y, en algunos casos, de forma gratuita. Entre los servicios ofrecidos en la nube por parte de las empresas que se dedican a ello, se encuentra la creación de máquinas virtuales, también denominadas instancias.

La creación de una instancia consiste en definir las características de la máquina virtual: sistema operativo, capacidad de almacenamiento, CPU, memoria, capacidad de Red… A la instancia se le asocia una IP pública que servirá para realizar la conexión a la misma. Existe una relación entre las herramientas de virtualización y las instancias de máquinas virtuales de los servicios de la nube que consiste en la posibilidad de importar y exportar máquinas virtuales de un servicio en la nube a una máquina virtual local y viceversa.

Por ejemplo, en la página web de Amazon se describe la importación y exportación de máquinas virtuales entre los servicios en la nube de Amazon (AWS) y la herramienta de virtualización VMware.[1]

[1] Amazon. VM Import/Export.

Virtual box

Esta unidad se va a centrar en el uso de Virtual Box, ya que cubre las necesidades esenciales requeridas en este ámbito, que son esencialmente:

- Evitar las incompatibilidades del software que se va a utilizar con el sistema operativo instalado en una máquina —asegurando así que se podrá hacer uso del software con independencia de la máquina utilizada—.
- Evitar al usuario tener que realizar la instalación y configuración del software, usando para ello máquinas virtuales que ya lo tienen instalado y configurado, lo que solo requerirá cargar la máquina virtual correspondiente.

IV. Creación y configuración de una máquina virtual

Virtual Box es un programa de software libre que permite la instalación de otro sistema operativo, dentro del sistema existente, y la interacción con el mismo. De esta forma, es posible instalar programas diseñados para los sistemas operativos clientes. En esta sección se va a describir cómo se instala Virtual Box, además de cómo se puede crear y configurar de una manera básica una máquina virtual.

4.1. Requerimientos para la instalación de virtual box

Para crear máquinas virtuales con Virtual Box, el ordenador debe cumplir con los siguientes requisitos:

Memoria RAM: mínimo 8 GB, ya que garantizará que el sistema operativo anfitrión y huésped cuenten con suficiente memoria.

Almacenamiento: al menos 20 GB libres para asignar alsistema operativo huésped.

Procesador: el ordenador debe contar con un procesador relativamente nuevo Intel o AMD, ya que el sistema operativo anfitrión comparte recursos con el sistema operativo huésped y, además, el procesador debe soportar la virtualización; esta función debe habilitarse desde el BIOS. En la página web de BlueStacks se describe el proceso para verificar si su procesador soporta la virtualización y cómo habilitarla.[2]

[2] BlueStacks. "¿Cómo puedo habilitar la virtualización (VT) en mi PC?"

4.2. Instalación de Virtual Box

Para instalar Virtual Box, en primer lugar, hay que descargar el software desde la siguiente dirección (figura 2.1.).

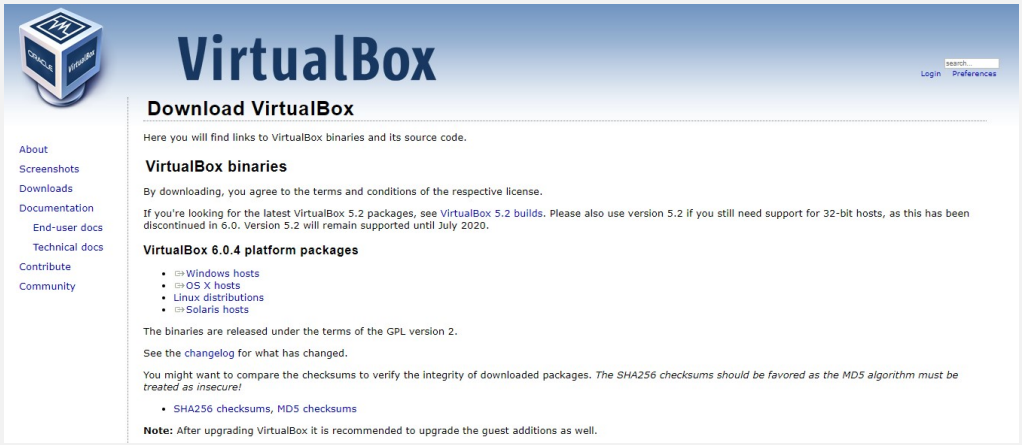

Figura 2.1. Sitio de descargas de Virtual Box.

Descarga de Virtual Box

Durante la unidad, se os enseña cómo descargar e instalar la versión 6.0.4. Sin embargo, es aconsejable que os descarguéis la última versión disponible. En las explicaciones siguientes, se utilizará la versión para Windows y se usarán funciones compatibles entre distintas versiones de Virtual Box (crear, importar y exportar) —para el resto de sistemas operativos el proceso es similar—. En la página de descargas, se pulsa sobre el enlace correspondiente al sistema operativo sobre el que se quiere instalar —en este caso sobre Windows hosts— y comienza la descarga (figura 2.2.).

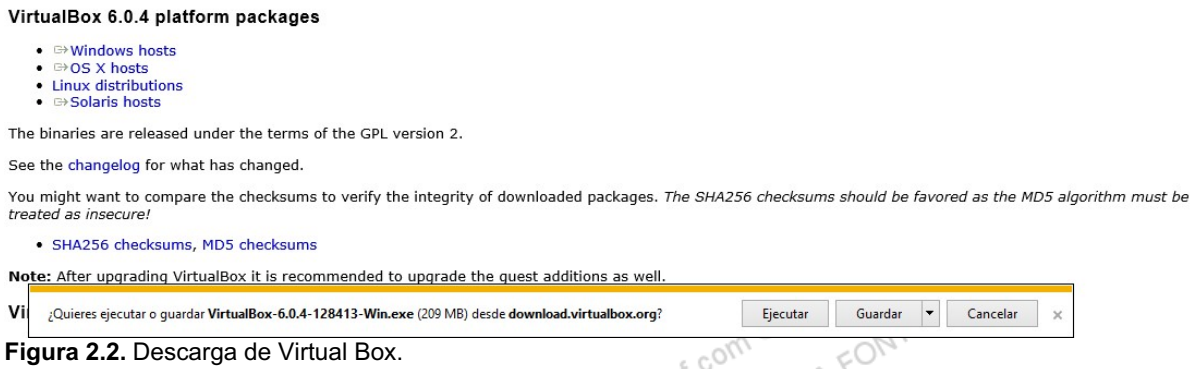

Figura 2.2. Descarga de Virtual Box.

Instalador de Virtual Box

Una vez que se ha bajado el archivo ejecutable, se pulsa sobre el instalador para que comience la ejecución del asistente (figura 2.3.).

Figura 2.3. Instalador de Virtual Box.

Custom Setup

Se pulsa sobre “Next” y aparece la ventana “Custom Setup”. Se deja la selección “VirtualBox Application” y se comprueba que la carpeta de instalación elegida por el instalador es C: \Program Files\Oracle\VirtualBox (figura 2.4.).

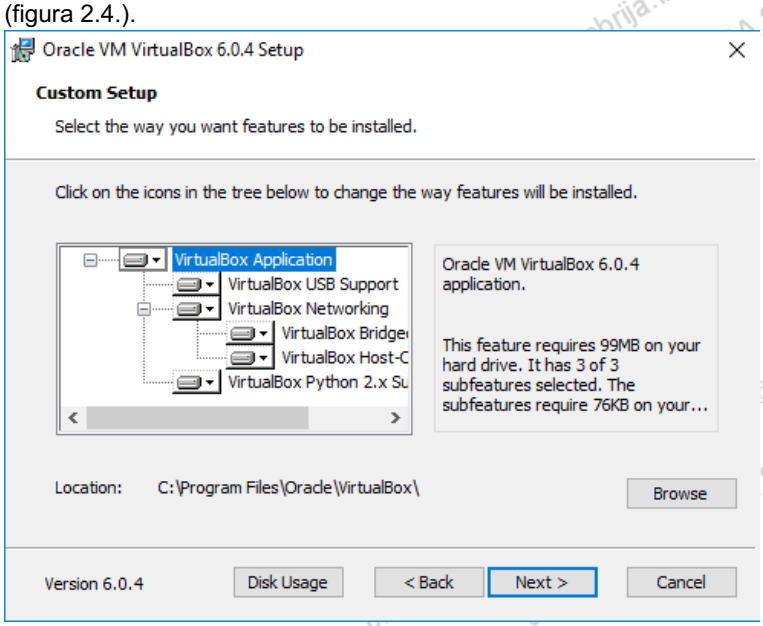

Figura 2.4. Ventana “Custom Setup”.

Se pulsa sobre “Next” en la siguiente ventana, sin cambiar las opciones que aparecen por defecto (figura 2.5.).

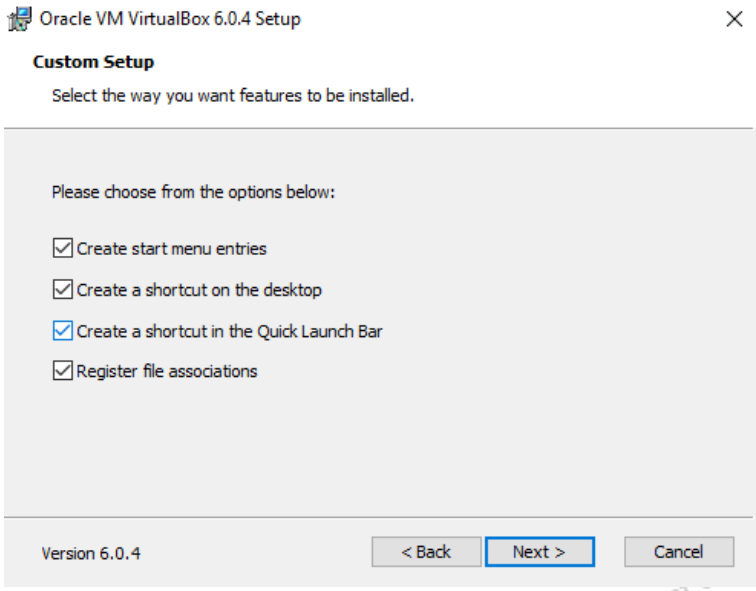

Figura 2.5. Opciones de configuración.

Desconexión de red

En la siguiente ventana que aparece, se indica que el ordenador se va a desconectar momentáneamente de la red. Se pulsa sobre “Yes” (figura 2.6.).

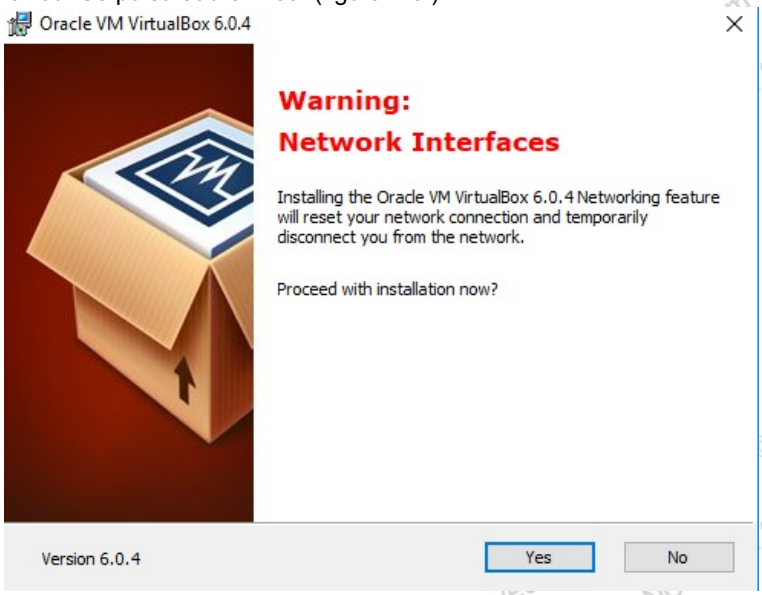

Figura 2.6. Ventana de desconexión de la red.

Instalación

Por último, en la siguiente ventana se pulsa sobre “Install” para que comience la instalación (figura 2.7.).

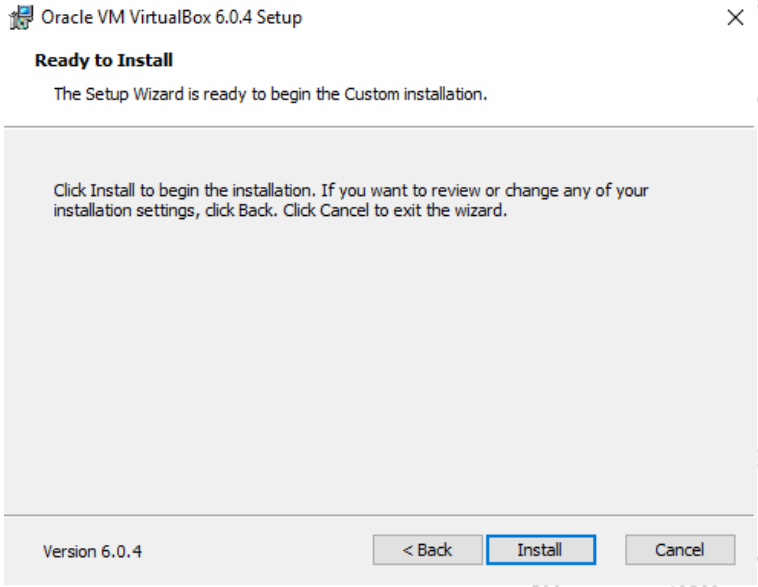

Figura 2.7. Ventana de instalación.

A continuación, aparecerá un aviso de Windows para preguntar si el programa puede hacer cambios en el sistema. Se pulsa sobre “Yes” y se deja que el instalador se ejecute hasta que finalice la instalación. En ese momento, aparece la pantalla de fin de instalación. Se pulsa sobre “Finish”, que ejecutará Virtual Box (figura 2.8.).

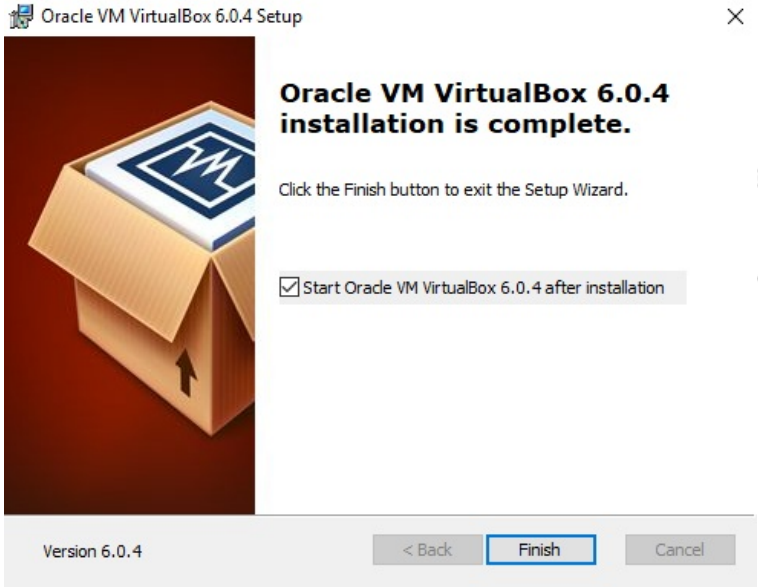

Figura 2.8. Ventana de finalización de instalación.

Interface de Virtual Box

En la figura 2.9., se puede observar la interface principal de Virtual Box.

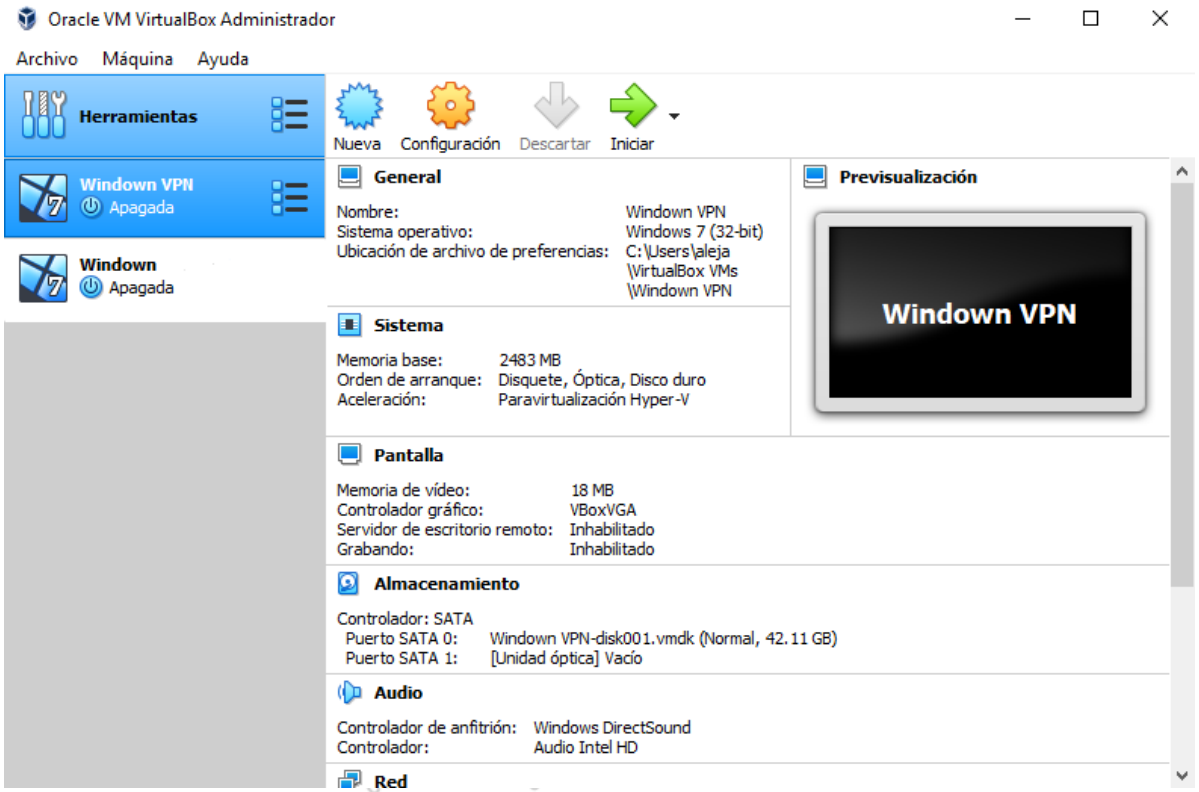

Figura 2.9. Interface principal de Virtual Box.

4.3. Creación y configuración de una máquina virtual

Para crear una máquina virtual, lo primero que se necesita es una imagen del sistema operativo que se desea virtualizar. Para ilustrar la creación de la máquina virtual, se va a utilizar Lubuntu.

Se trata de una distribución de Linux pensada para máquinas que disponen de pocos recursos software. La descarga de Lubuntu se realizar desde la siguiente dirección: página web de Lubuntu, (figura 2.10.).

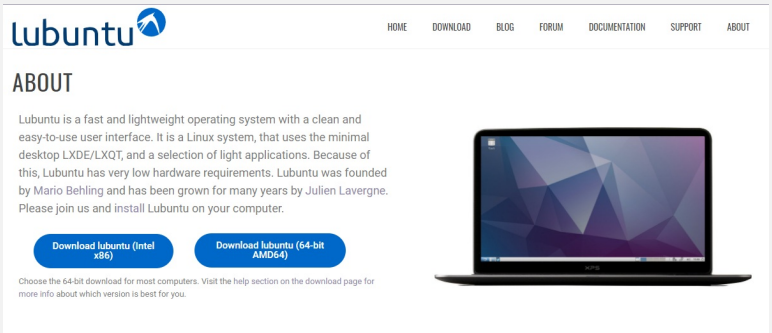

Figura 2.10. Página de descargas de Lubuntu.

Algunas imágenes o figuras se pueden ampliar para mejorar la visualización.

1

En la página de descargas, se pulsa sobre el enlace más adecuado a la máquina que se posea (figura 2.11.).

Figura 2.11. Descarga de Lubuntu.

Una vez descargado Lubuntu, se va a realizar la creación de la máquina virtual. Para ello, se abre Virtual Box pulsando sobre el icono que habrá en el escritorio, o bien en la lista de programas, o bien sobre el archivo VirtualBox.exe que se encuentra en la carpeta de instalación C:\Program Files\Oracle\VirtualBox. En la interface principal de Virtual Box, se pulsa sobre el icono “Nueva” para crear una nueva máquina virtual. Como resultado, se abre el asistente (figura 2.12.).

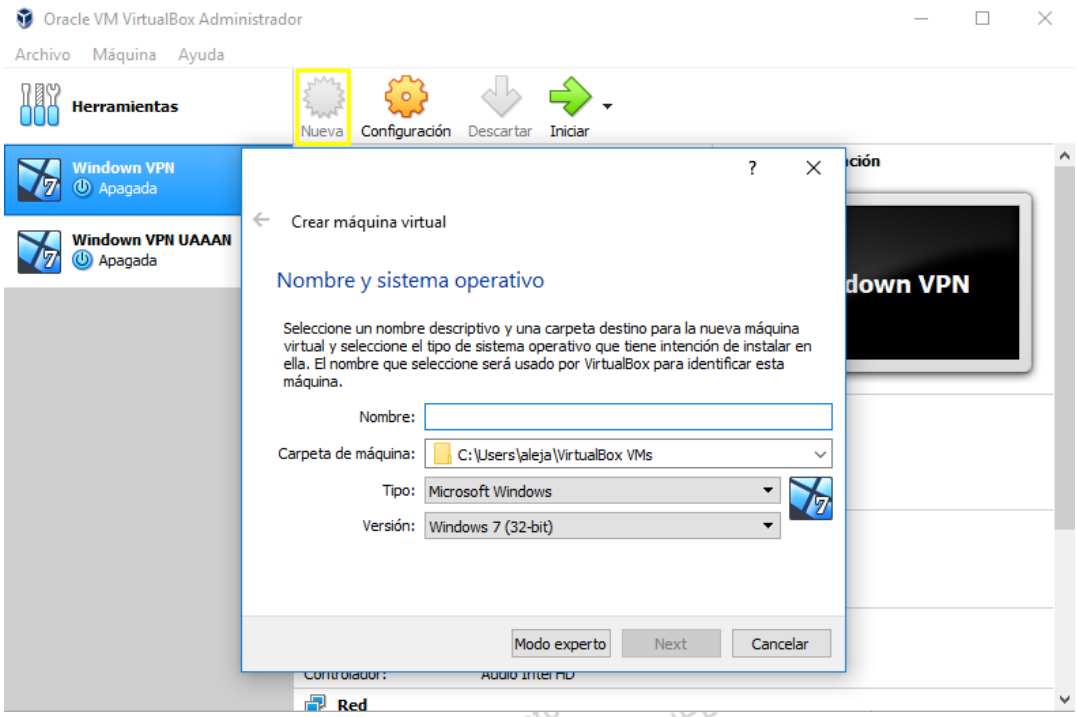

Figura 2.12. Asistente para crear una máquina virtual

2

En el asistente, se inserta un nombre para la máquina virtual “Lubuntu-Virtualizado”, se elige como tipo “Linux” y como versión “Ubuntu (64-bit)”. Por último, se pulsa sobre el enlace “Next” (figura 2.13.).

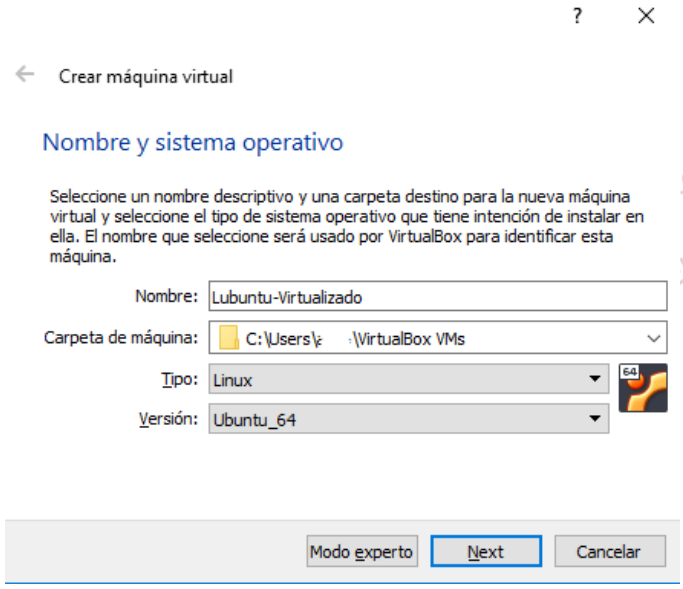

Figura 2.13. Datos de la máquina virtual

En la siguiente ventana, se debe elegir cuánta memoria se va a asignar al sistema virtualizado (memoria que se quita a Windows), y se pulsa sobre “Next”. Se puede seleccionar la recomendación que indica el asistente (figura 2.14.).

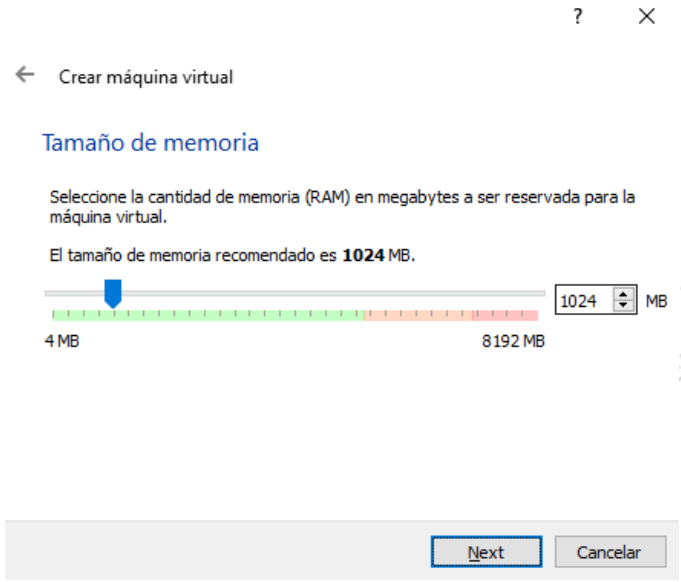

Figura 2.14. Tamaño de la memoria de la máquina virtual.

En la siguiente ventana, se verifica que se ha seleccionado la opción de “Crear disco virtual ahora” y se pulsa sobre “Crear” (figura 2.15.).

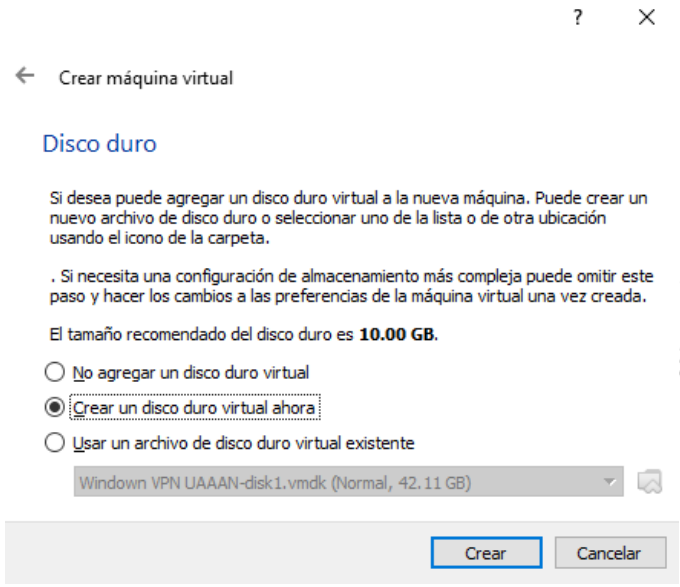

Figura 2.15. Creación de máquina virtual.

En la siguiente ventana, se elige el tipo de archivo que se desea utilizar para el disco duro virtual. Se puede dejar la recomendación que indica el asistente (figura 2.16.) y se pulsa sobre “Next”.

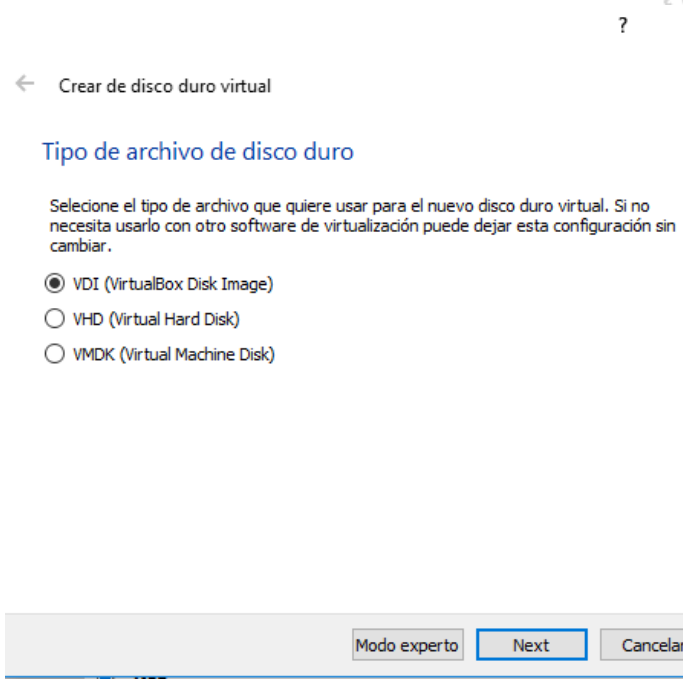

Figura 2.16. Selección del tipo de archivo del disco duro

3

Config. almacenamiento

En la siguiente ventana, se selecciona cómo se realizará el almacenamiento en el disco duro. Se mantiene la recomendación que indica el asistente (figura 2.17.) y se pulsa sobre “Next”.

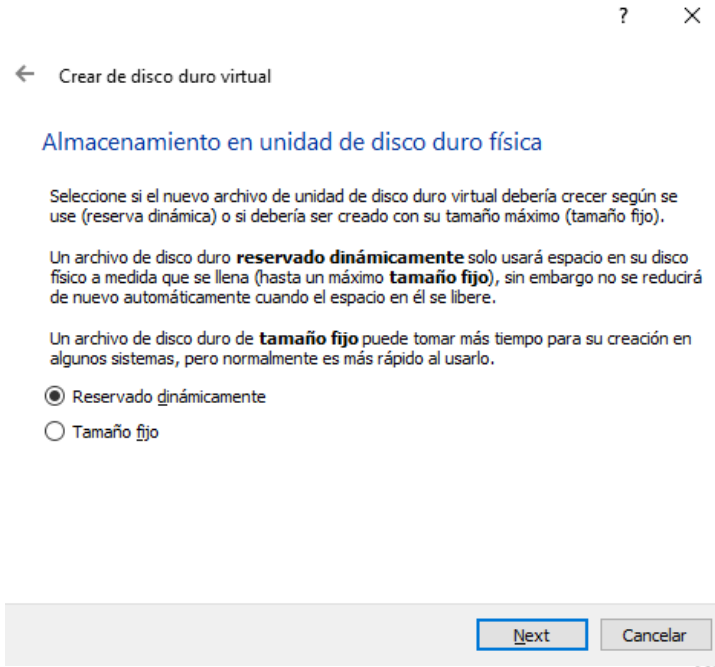

Figura 2.17. Selección del tipo de almacenamiento en el disco duro

Ubicación

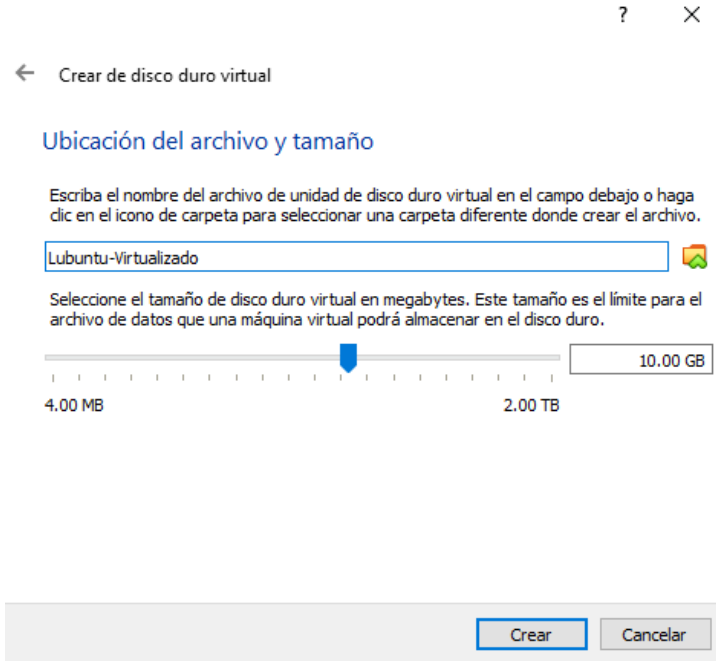

Figura 2.18. Ubicación del archivo del disco duro

"Crear"

Al pulsar sobre “Crear”, aparece la ventana de Virtual Box actualizada con la máquina que se ha configurado (figura 2.19.).

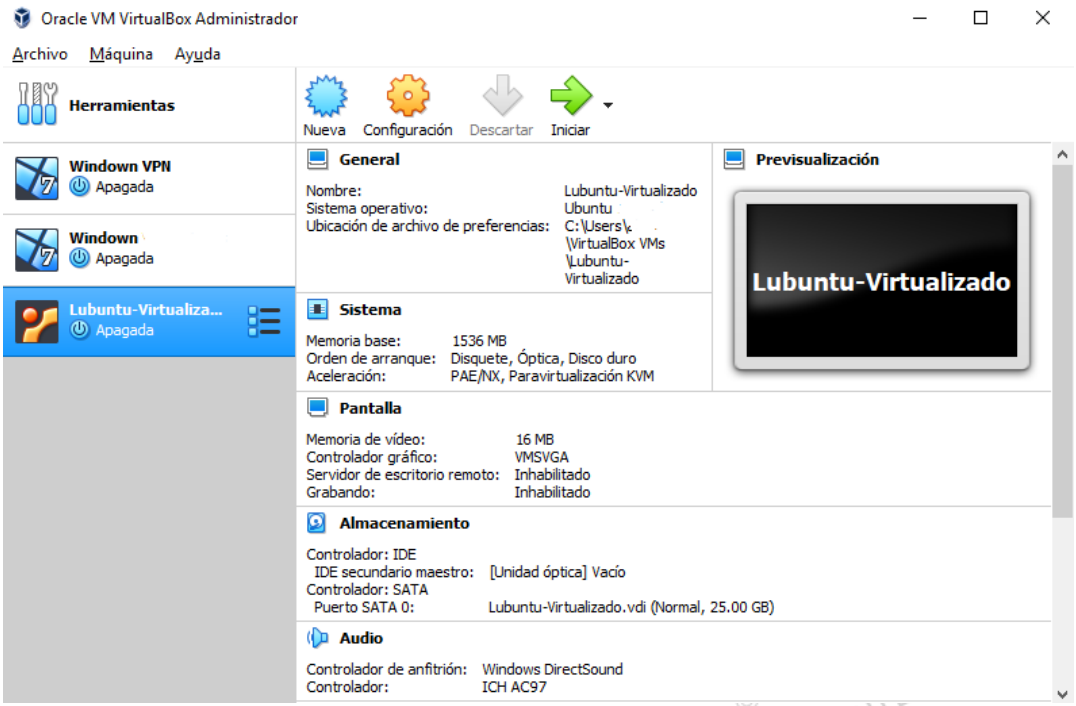

Figura 2.19. Máquina virtual creada.

Config. máquina virtual

La máquina virtual aún no se ha creado, lo que se ha hecho es configurar las características de la máquina. Antes de crear la máquina es necesario configurar algunas más, lo cual se realiza seleccionando la máquina y pulsando sobre el enlace “Configuración” que genera la ventana mostrada en la figura 2.20.

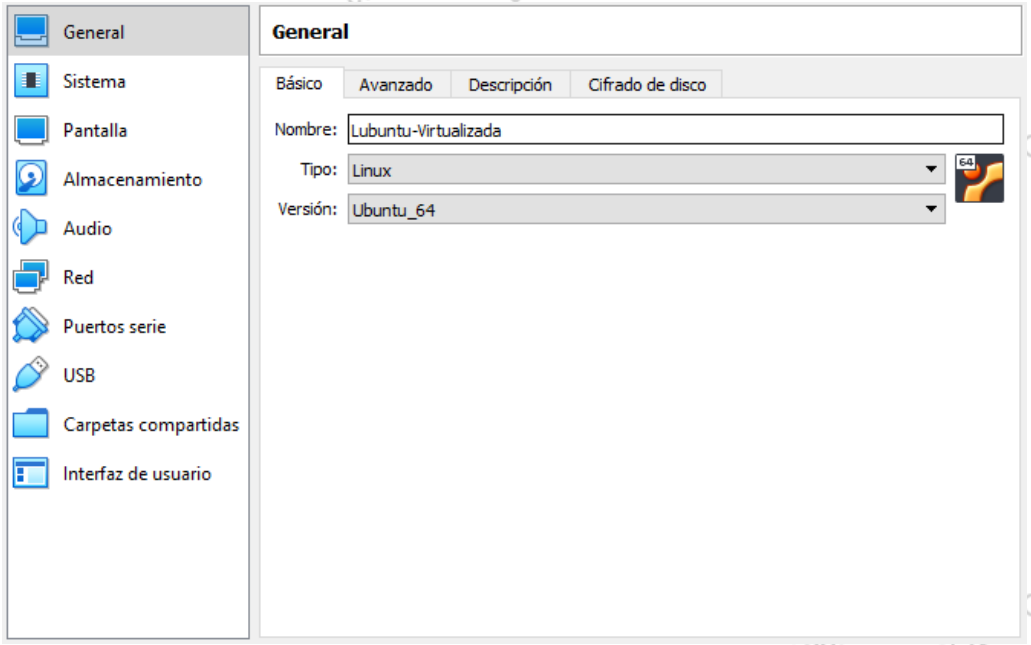

Figura 2.20. Configuración de la máquina virtual.

Almacenamiento del asistente

En esta ventana, se pulsa sobre el icono “Almacenamiento” y aparece la pestaña que se muestra en la figura 2.21.

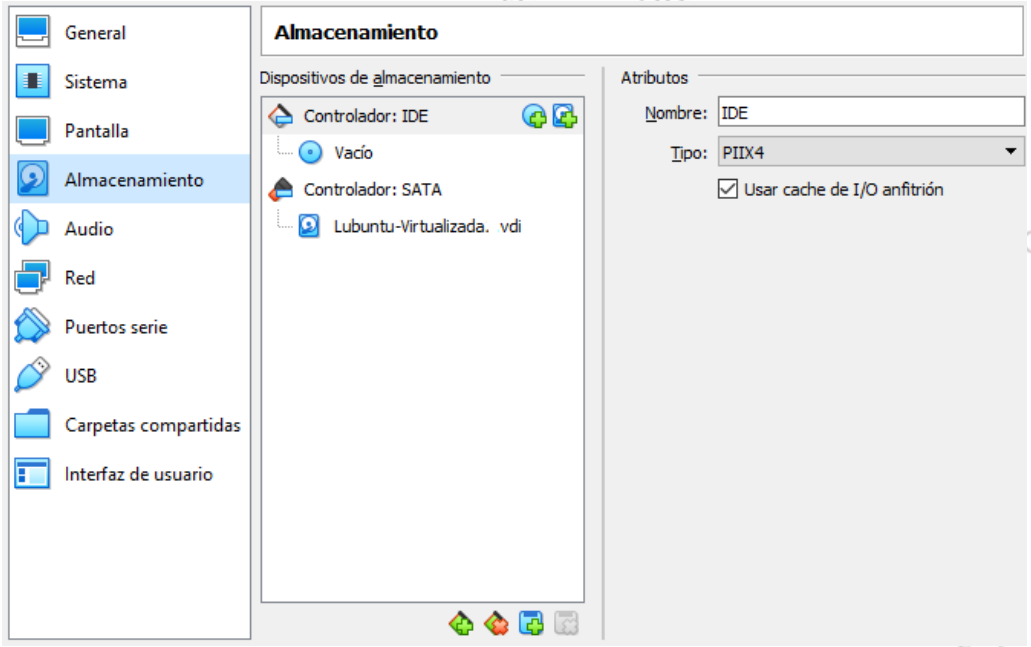

Figura 2.21. Pestaña de almacenamiento del asistente de configuración.

Configurador de controlador

En esta pestaña, se pulsa en el icono “CD” del cuadro “Árbol de almacenamiento” del “Controlador: IDE” que aparece vacío (figura 2.22).

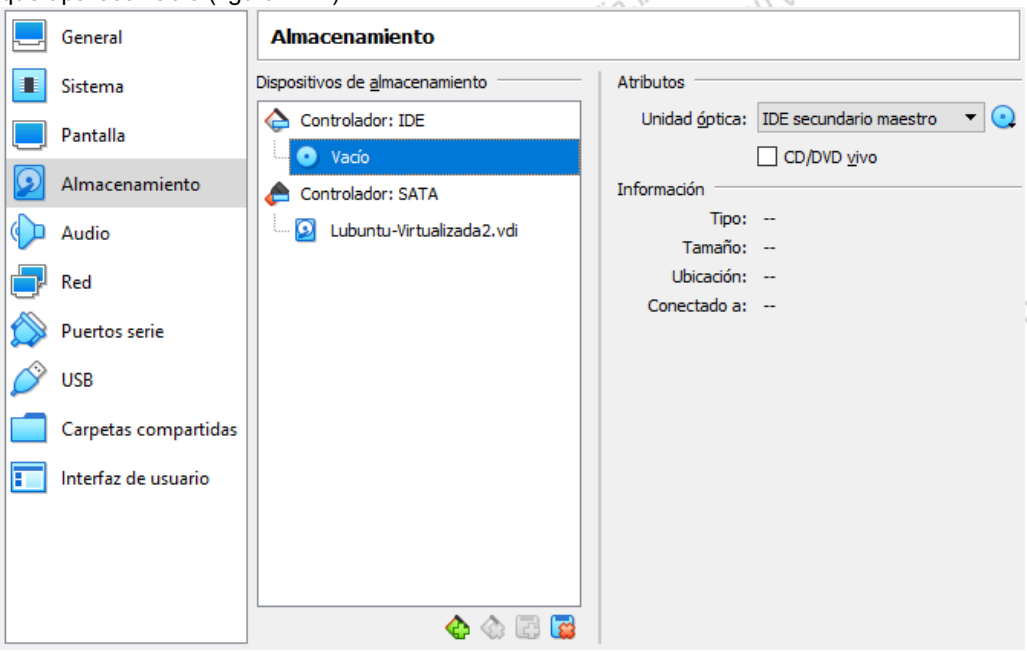

Figura 2.22. Configuración del controlador.

Disco óptico virtual

En el cuadro “Atributos”, se pulsa sobre el icono del “CD” y aparece una lista desplegable. Se selecciona “Seleccione archivo de disco óptico virtual…” (figura 2.23.).

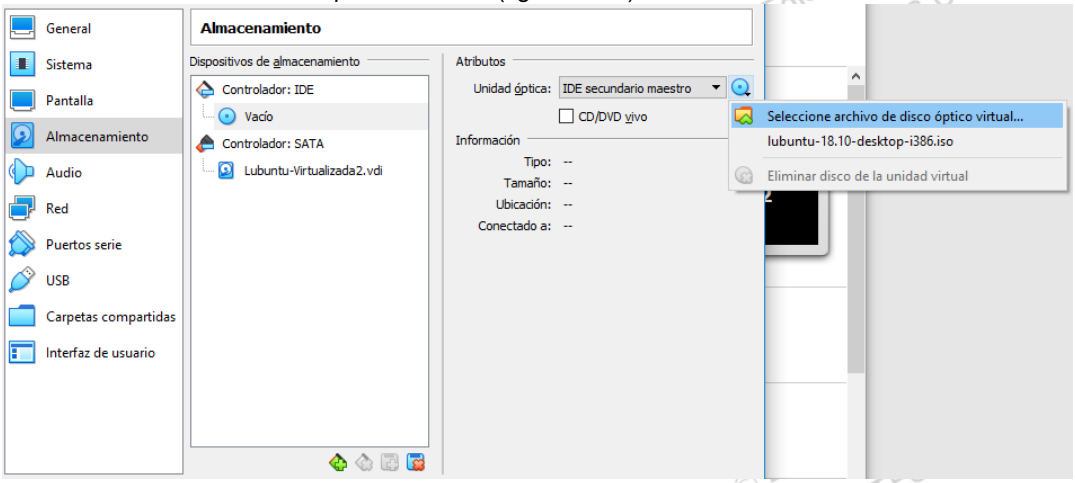

Figura 2.23. Selección de archivo

4

Aparece un navegador de archivos y se selecciona el archivo de “Lubuntu” que se ha descargado previamente (figura 2.24.).

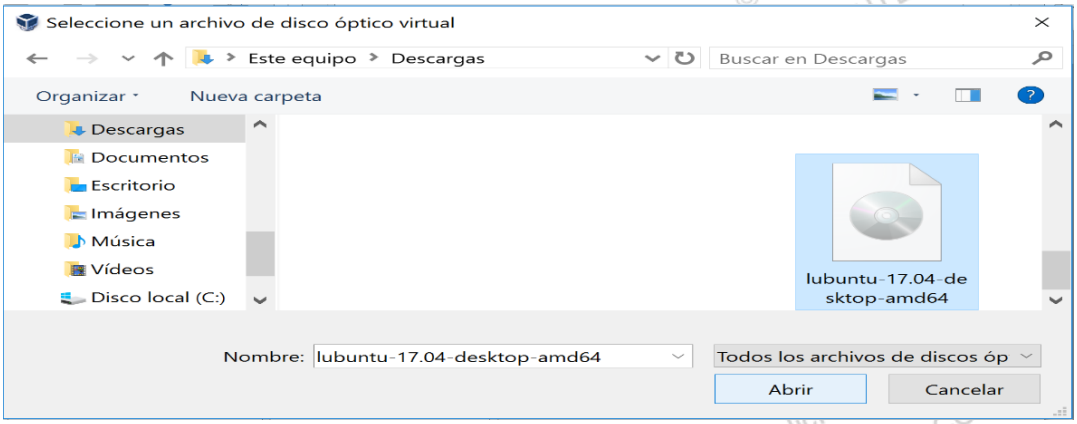

Figura 2.24. Selección del archivo de Lubuntu.

Tras realizar la selección, la pantalla aparece como en la figura 2.25.

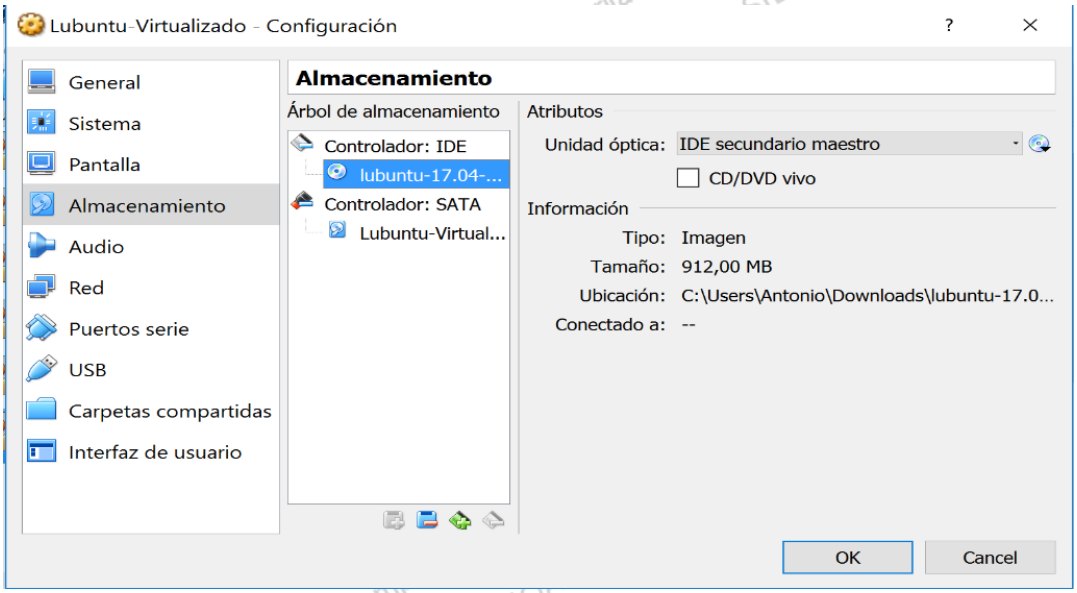

Figura 2.25. Pantalla resultante de la configuración.

A continuación, se pulsa sobre “OK” para volver a la interface principal de Virtual Box en la que se puede observar que se ha actualizado el cuadro almacenamiento (figura 2.26.).

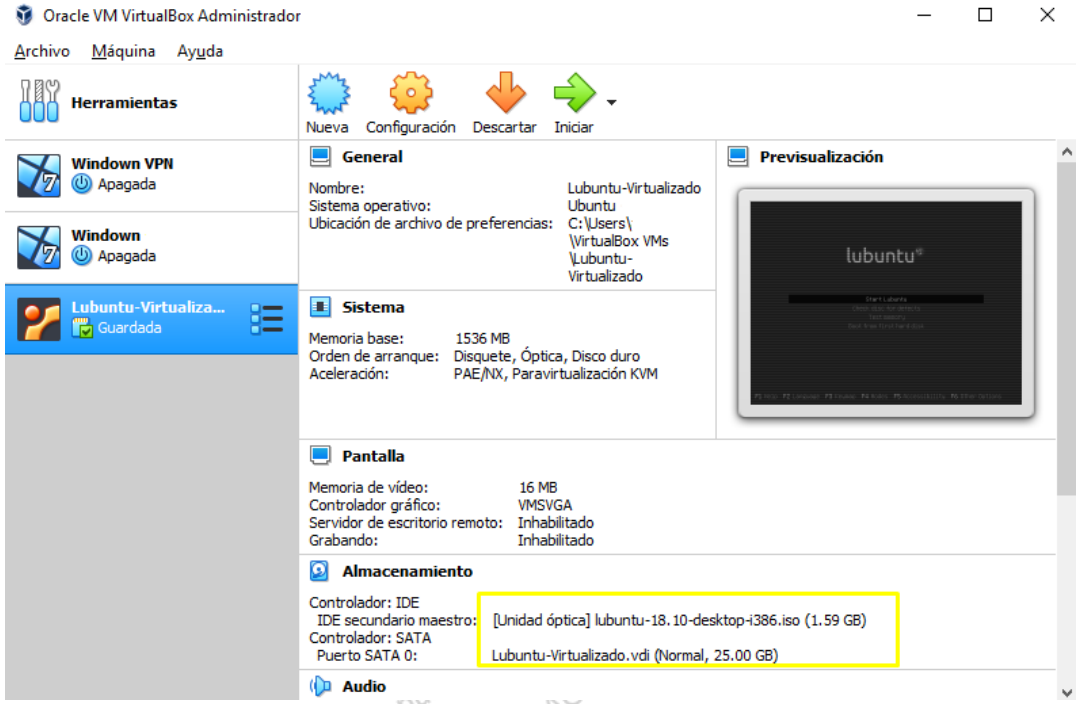

Figura 2.26. Interface principal actualizada.

Ahora, se va a crear el sistema virtualizado. Desde la pantalla principal, se pulsa sobre el icono de "Iniciar" para que se cargue el sistema.

Aparecerán diferentes mensajes y pantallas de instalación (figura 2.27.). Son mensajes similares a los que se mostrarían si se estuviera instalando realmente el sistema operativo. Estos mensajes solo aparecerán la primera vez que se cargue el sistema virtualizado, no lo harán en las restantes ocasiones.

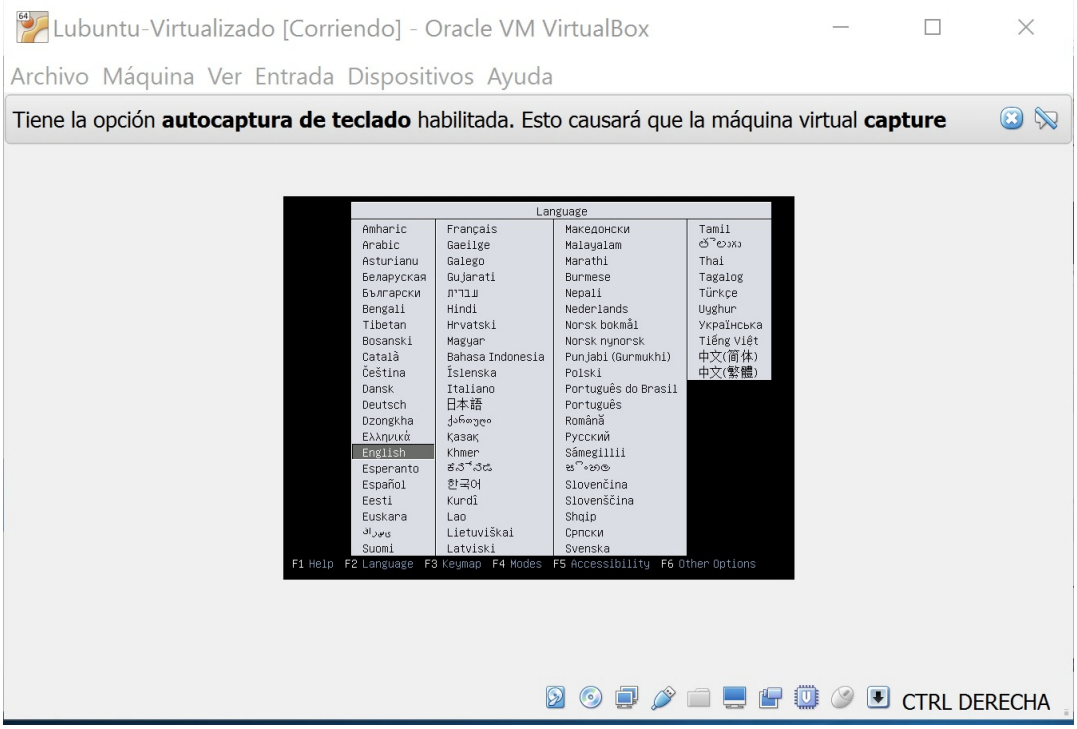

Figura 2.27. Pantalla de instalación del sistema

Se selecciona “Instalar Lubuntu” (figura 2.28.).

Figura 2.28. Instalación de Lubuntu.

5

Se selecciona el idioma (figura 2.29.).

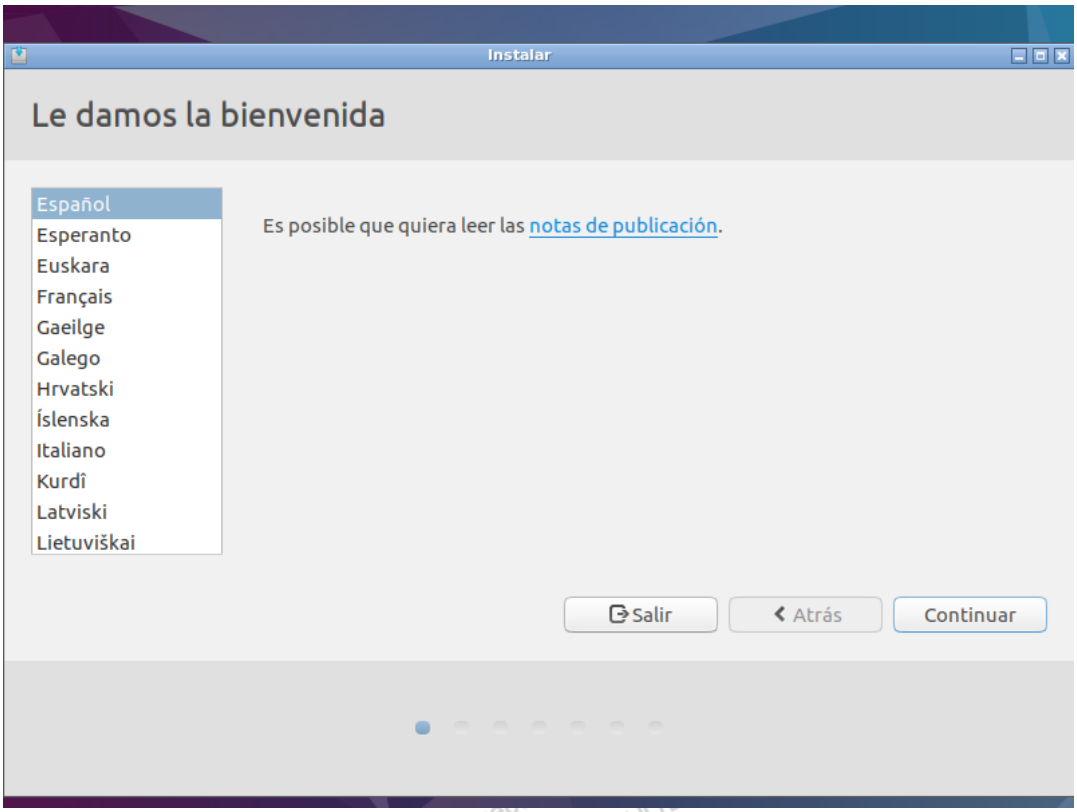

Figura 2.29. Selección del idioma.

Se seleccionan paquetes de software adicional para su instalación (figura 2.30.).

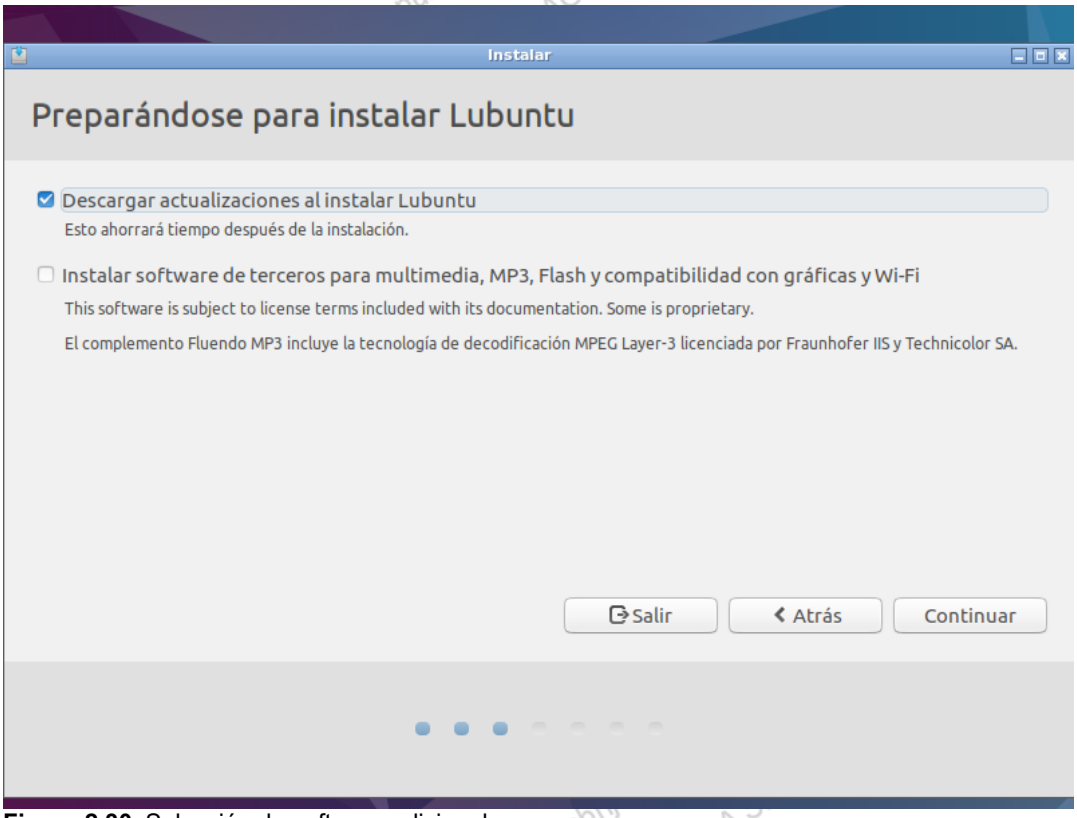

Figura 2.30. Selección de software adicional.

Y el tipo de instalación (figura 2.31.).

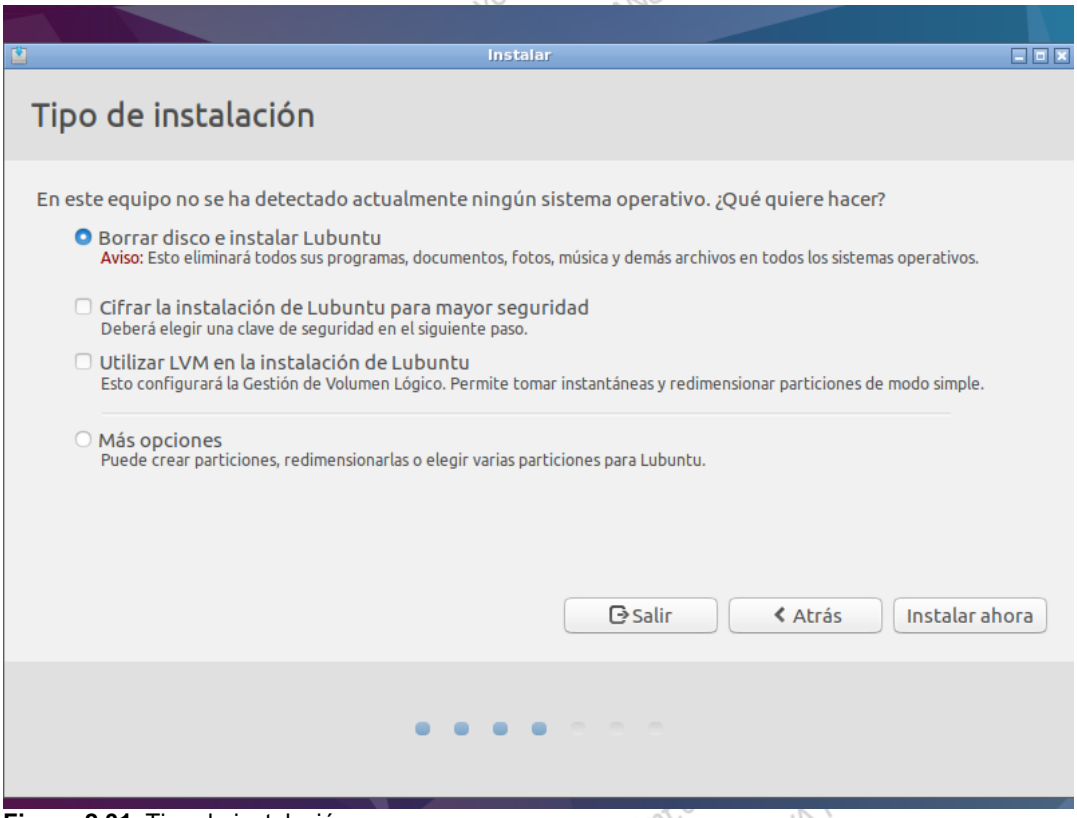

Figura 2.31. Tipo de instalación.

Se pulsa sobre “Continuar” en la pantalla que aparece a continuación (figura 2.32.).

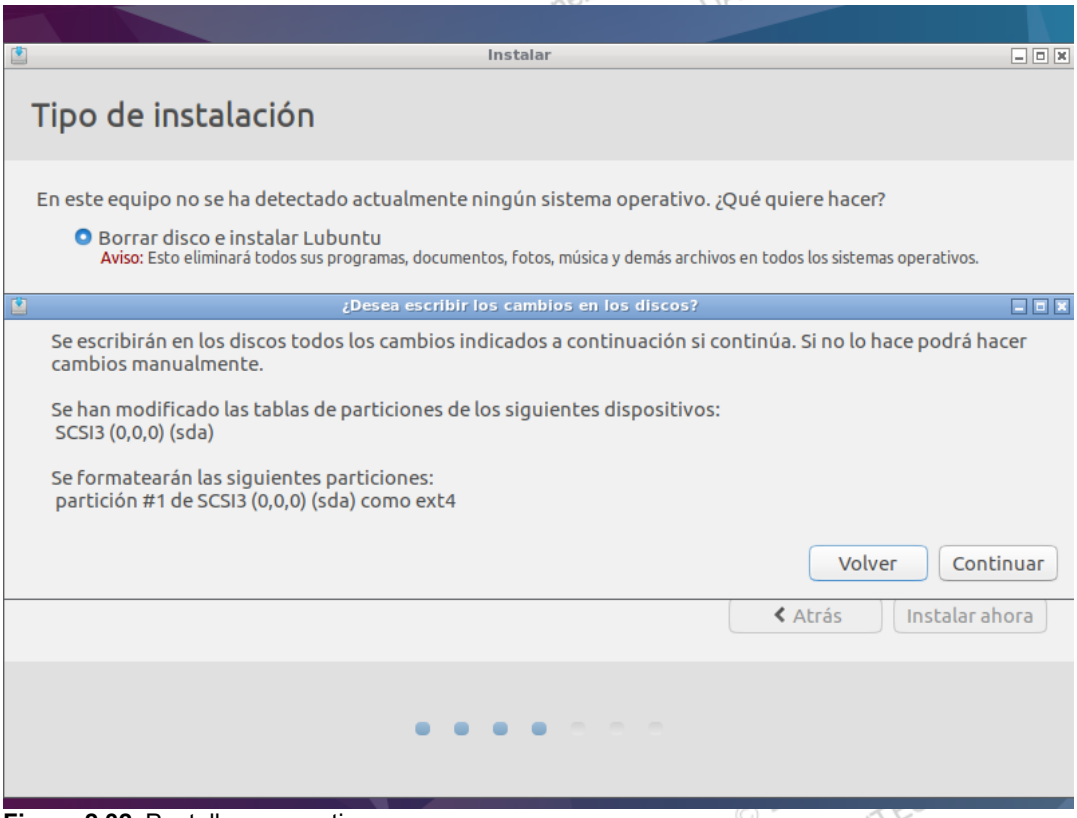

Figura 2.32. Pantalla consecutiva.

Se selecciona nuestra localización (figura 2.33.).

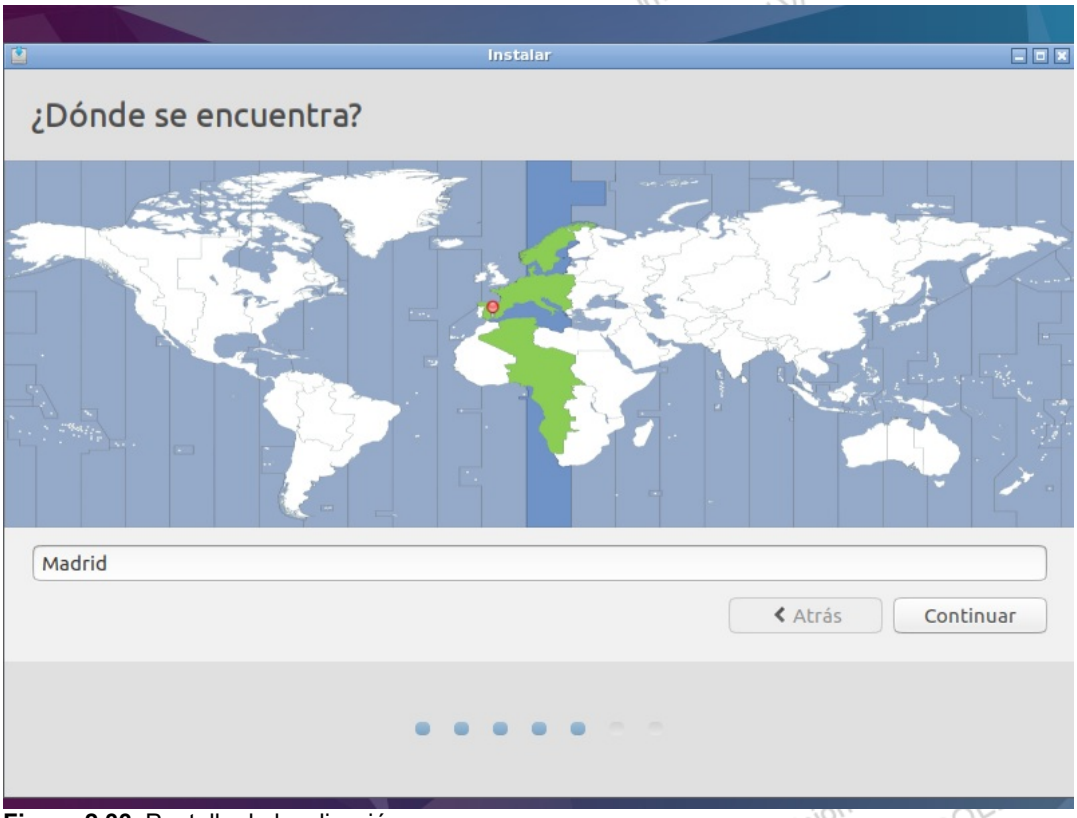

Figura 2.33. Pantalla de localización.

Y la disposición del teclado (figura 2.34.).

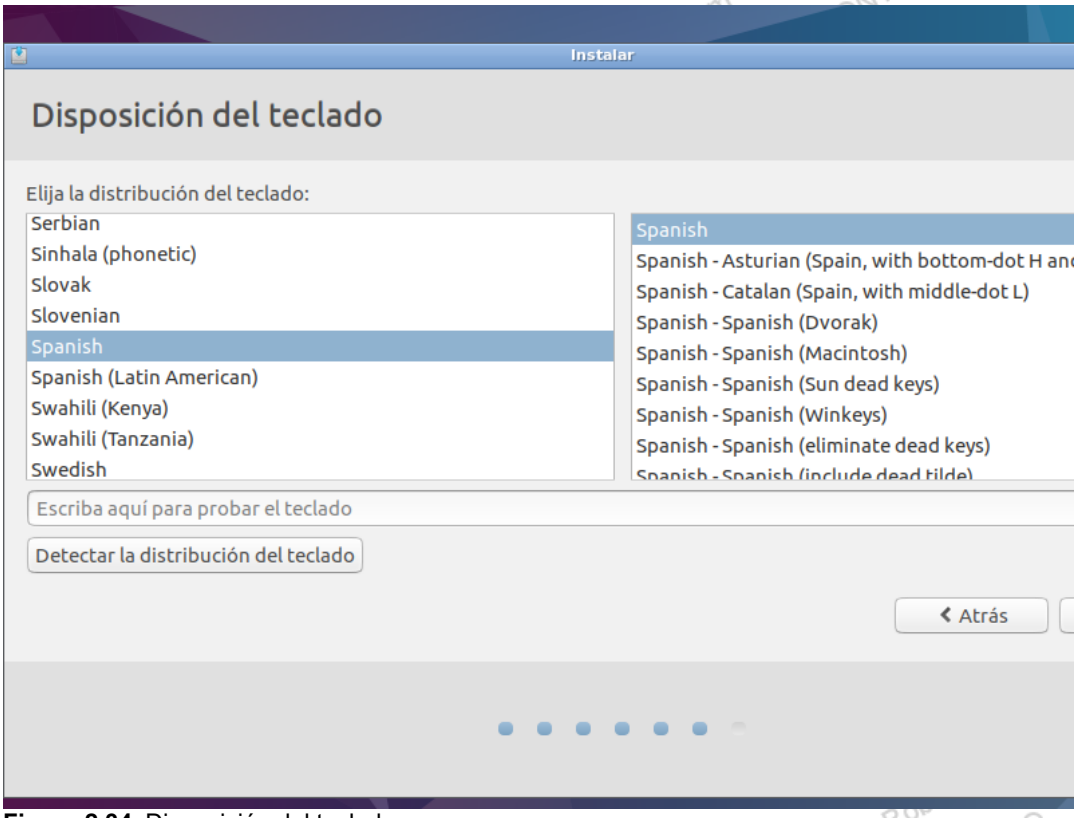

Figura 2.34. Disposición del teclado.

Se configura el usuario (figura 2.35.).

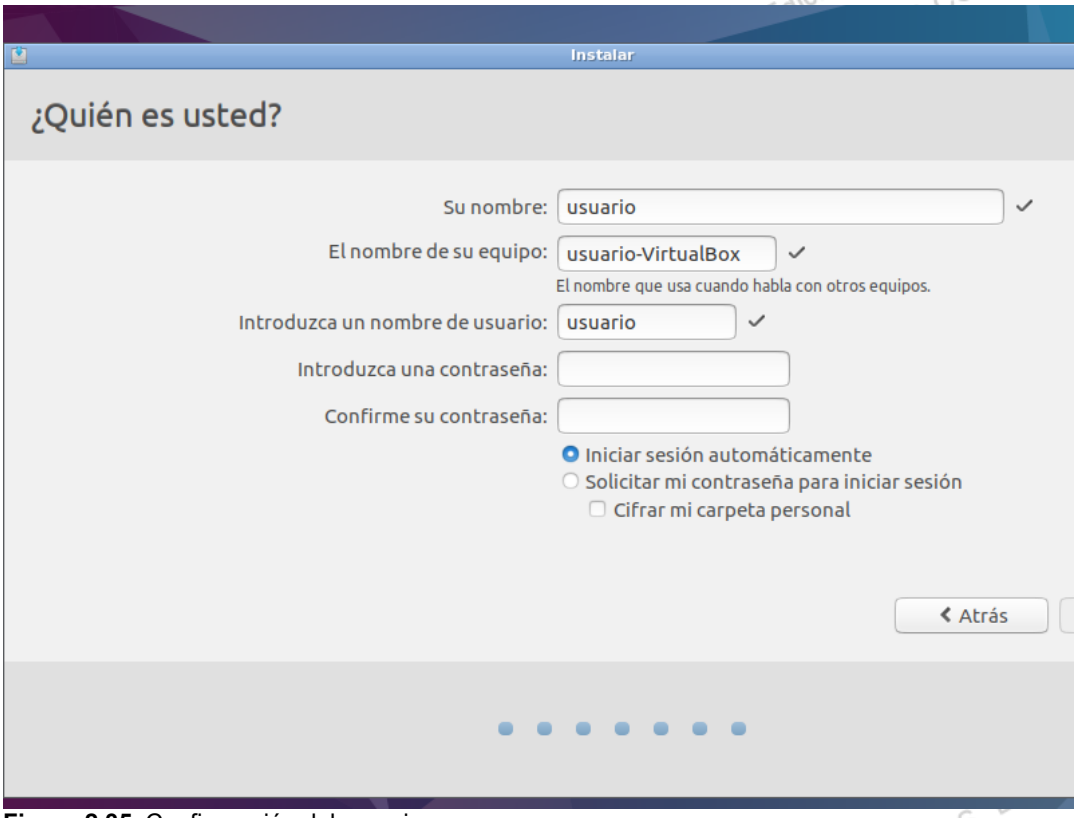

Figura 2.35. Configuración del usuario.

6

Se realiza la instalación de Lubuntu (figura 2.36.).

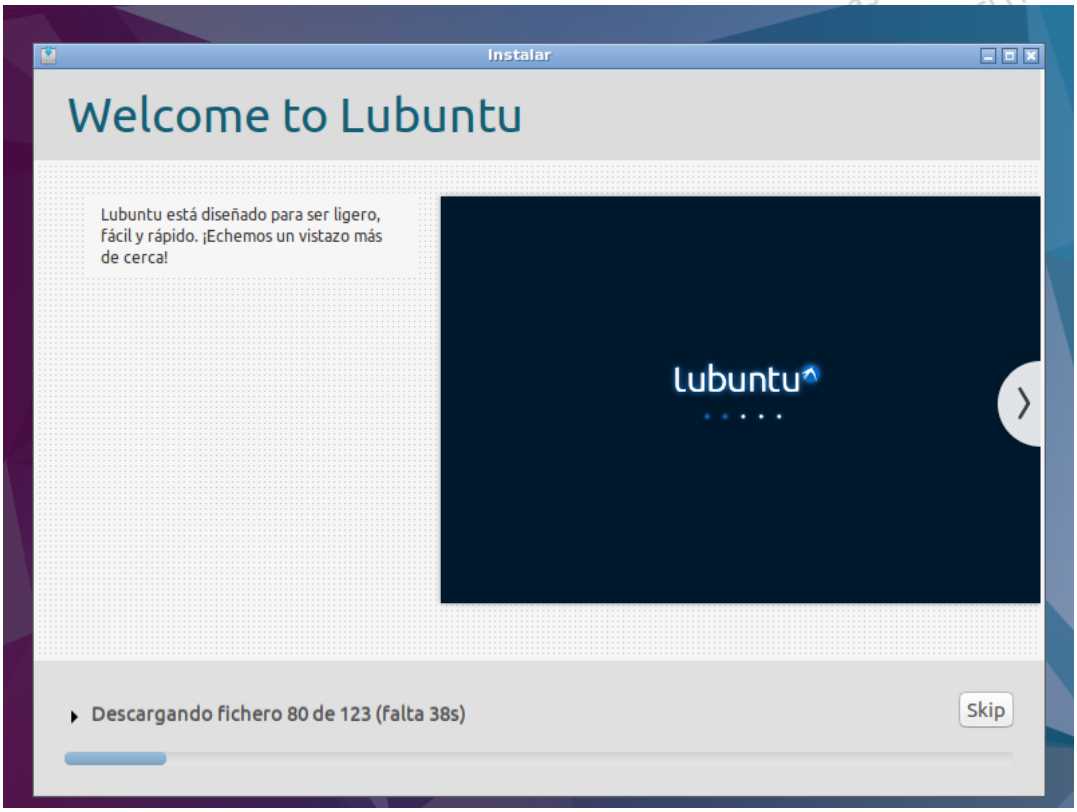

Figura 2.36. Instalación de Lubuntu.

Se completa la instalación y se pulsa sobre “Reiniciar ahora” (figura 2.37.).

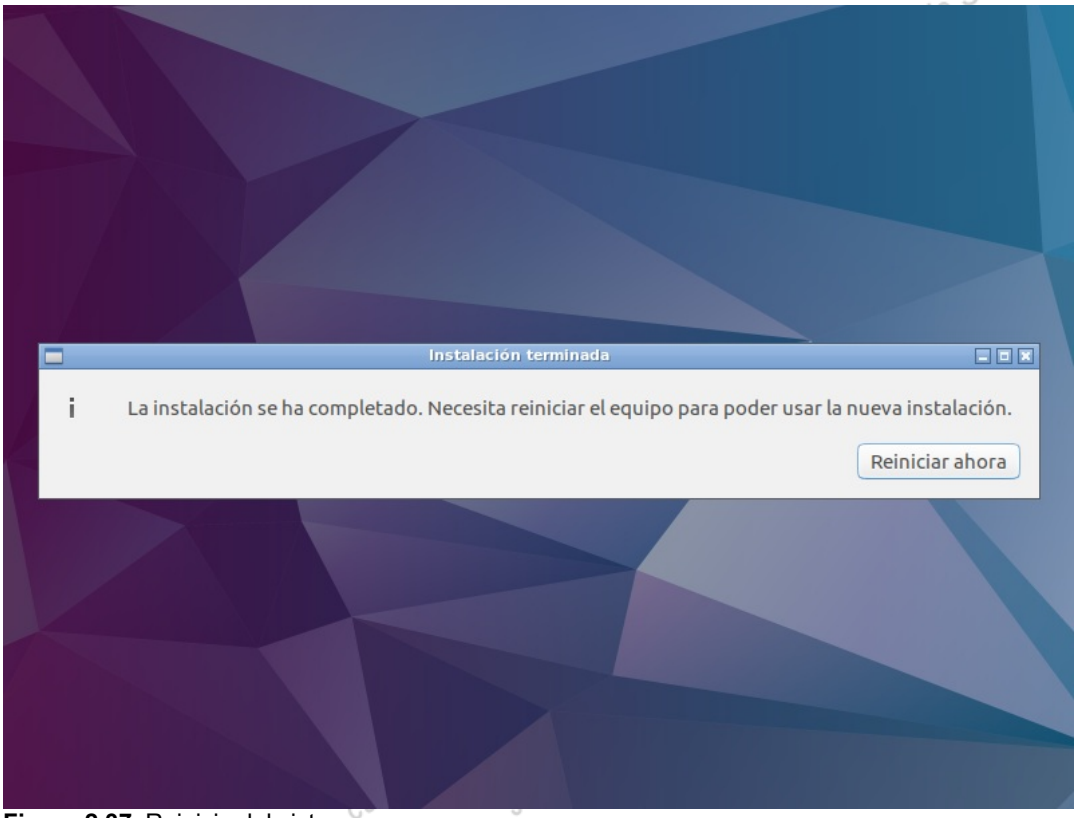

Figura 2.37. Reinicio del sistema.

Se carga el sistema operativo (figura 2.38.).

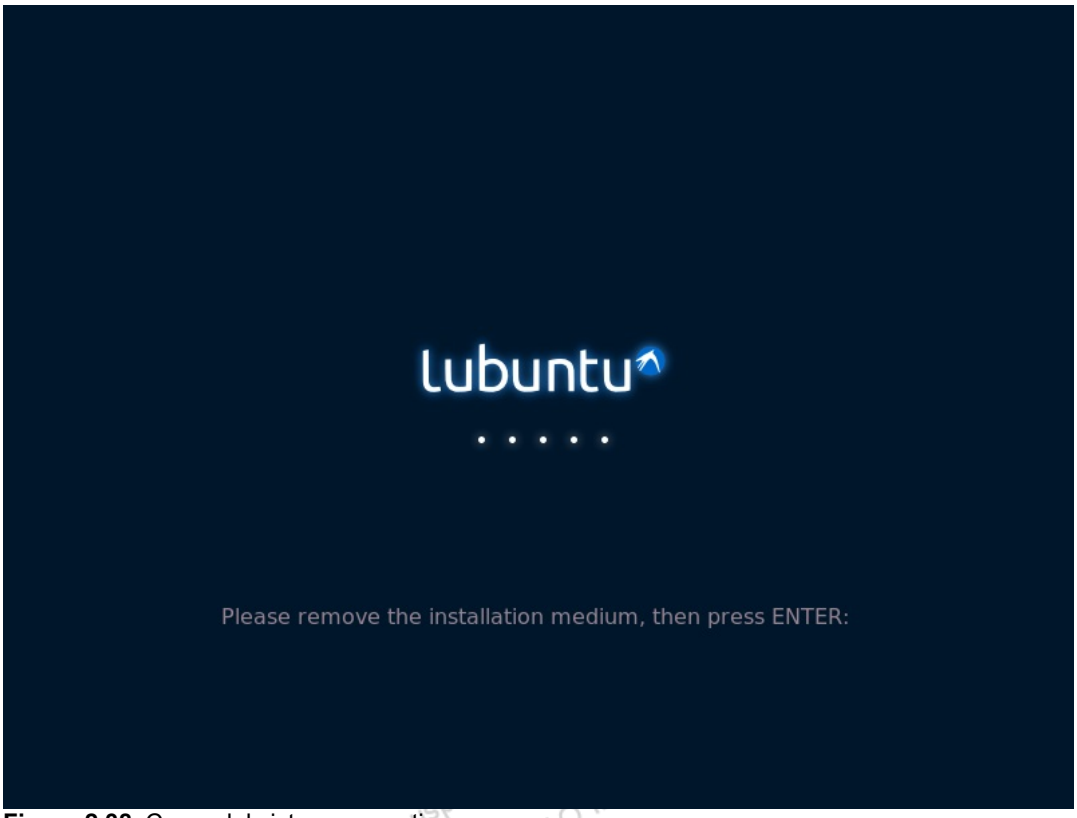

Figura 2.38. Carga del sistema operativo.

En la figura 2.39., se puede observar el sistema operativo virtualizado.

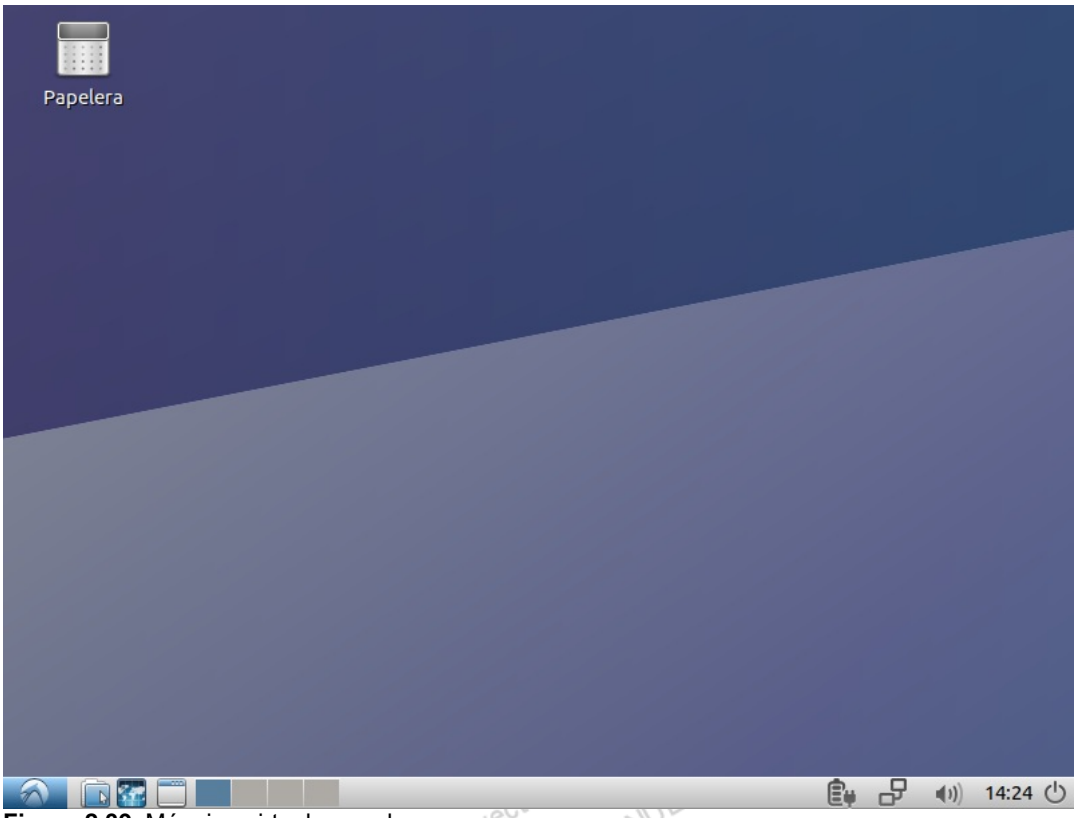

Figura 2.39. Máquina virtual cargada.

Cuando se cierra la máquina virtual, aparecen**varias opciones** (figura 2.40.)

- Guardar el estado de la máquina: cierra la MV guardando el estado en el que se encuentra en ese momento, de modo que, al volver a iniciarla, continuará justo en el punto donde se cerró. No se pierde ningún tipo de información.
- Enviar señal de apagado: es el apagado ACPI.
- Apagar la máquina: equivale a cerrar la máquina sin guardar el estado, por lo que el SO no se cerrará correctamente y al iniciar la MV posteriormente, el SO hará los chequeos previstos cuando este no se cierre correctamente. Es posible que se pierdan datos, por lo que debe usarse como último recurso cuando la MV se ha bloqueado.

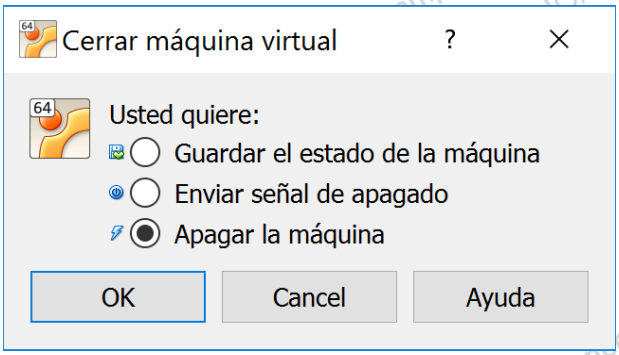

Figura 2.40. Opciones de cierre de la máquina virtual.

La próxima vez que se quiera ejecutar el sistema virtual bastará con seleccionar la máquina virtual y pulsar sobre “Iniciar” en la pantalla principal de Virtual Box.

V. Carga de una máquina virtual

Las máquinas virtuales son archivos que tienen el software de un sistema operativo virtualizado y permiten ejecutarlo en un sistema de virtualización. Virtual Box y VM Ware tienen funciones que permiten importar y exportar máquinas virtuales previamente configuradas, con el fin de facilitar y reducir los tiempos de configuración.

Existen varios formatos para exportar/importar máquinas virtuales. Físicamente son archivos con extensión *.ova, *.vdi, etc. Los más comunes son *.ova y *.vdo, que son compatibles con Virtual Box y VMWare.

OVA

El formato OVA (Open Virtualization Application/Appliance) es un formato estándar para empaquetar servicios virtualizados. Internamente es un fichero compreso TAR que almacena una colección de ficheros necesarios para virtualizar uno o más sistemas operativos. Virtual Box soporta OVA desde su versión 2.2.0.

VDI

Por su parte, VDI (Virtual Desktop Infrastructure) es un formato de imagen de un sistema operativo propio de Virtual Box.

Repositorios gratuitos

Hay repositorios gratuitos con máquinas virtuales ya preparadas para descargar e instalar:

- OSBoxes, máquinas virtuales disponibles para VirtualBox y VMWare de diferentes distribuciones Linux.
- VirtualBoxes, máquinas virtuales disponibles para VirtualBox de diversos sistemas operativos libres o de código abierto.

Descarga distribución de Lubuntu

Para ilustrar cómo cargar una máquina virtual, es posible descargar una de una distribución de Lubuntu desde el siguiente enlace (figura 2.41.): SourceForge,

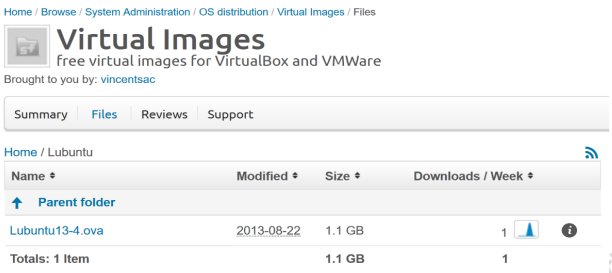

Figura 2.41. Descarga de una máquina virtual.

Importación a Virtual Box

Una vez descargada, se importa en Virtual Box. Para ello, se selecciona en el menú la opción: "Archivos"> "Importar Servicio Virtualizado" y se selecciona la máquina virtual que se ha descargado (figura 2.42.).

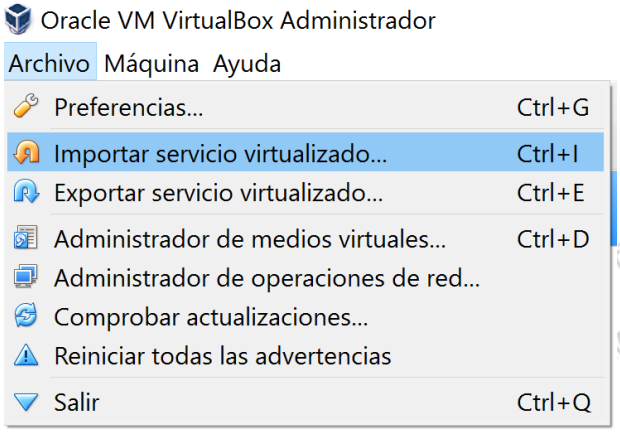

Figura 2.42. Importar máquina virtual.

Selección de máquina virtual

A continuación, se selecciona la máquina virtual que se quiere importar (figura 2.43.).

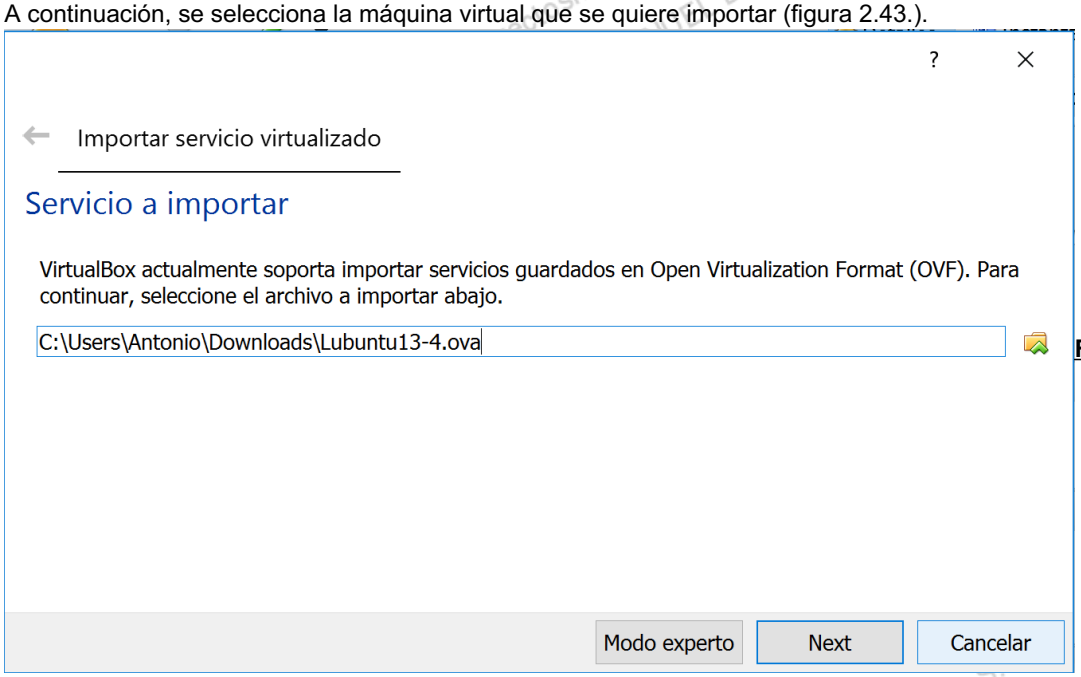

Figura 2.4

3. Selección de la máquina virtual a importar.

Resumen de máquina virtual

A continuación, se pulsa sobre “Next” y aparecerá un resumen de la máquina virtual que se va a importar. Dependiendo de la versión de virtual box, se debe marcar la opción “Reiniciar la dirección MAC de todas las tarjetas de red” y pulsar sobre “Importar” (figura 2.44.)

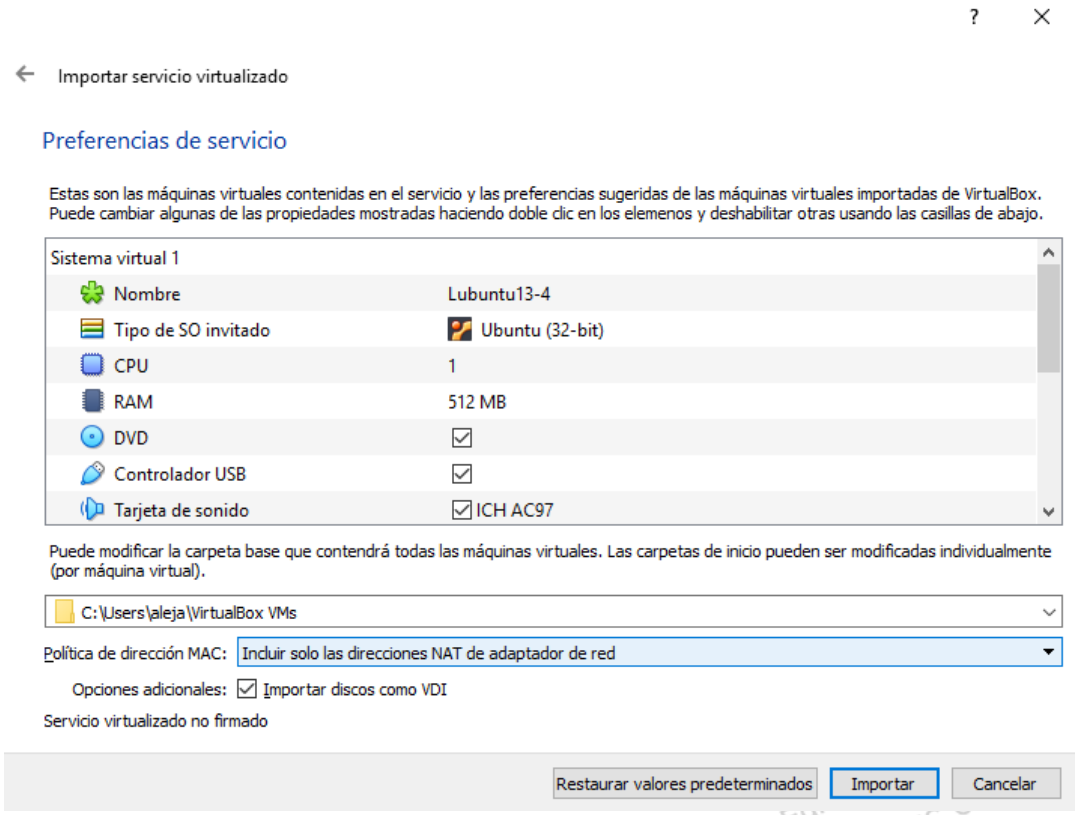

Figura 2.44. Configuración de la importación.

Se muestra una ventana con el estado del proceso de importación. El tiempo varía dependiendo de las características de la máquina (figura 2.45.).

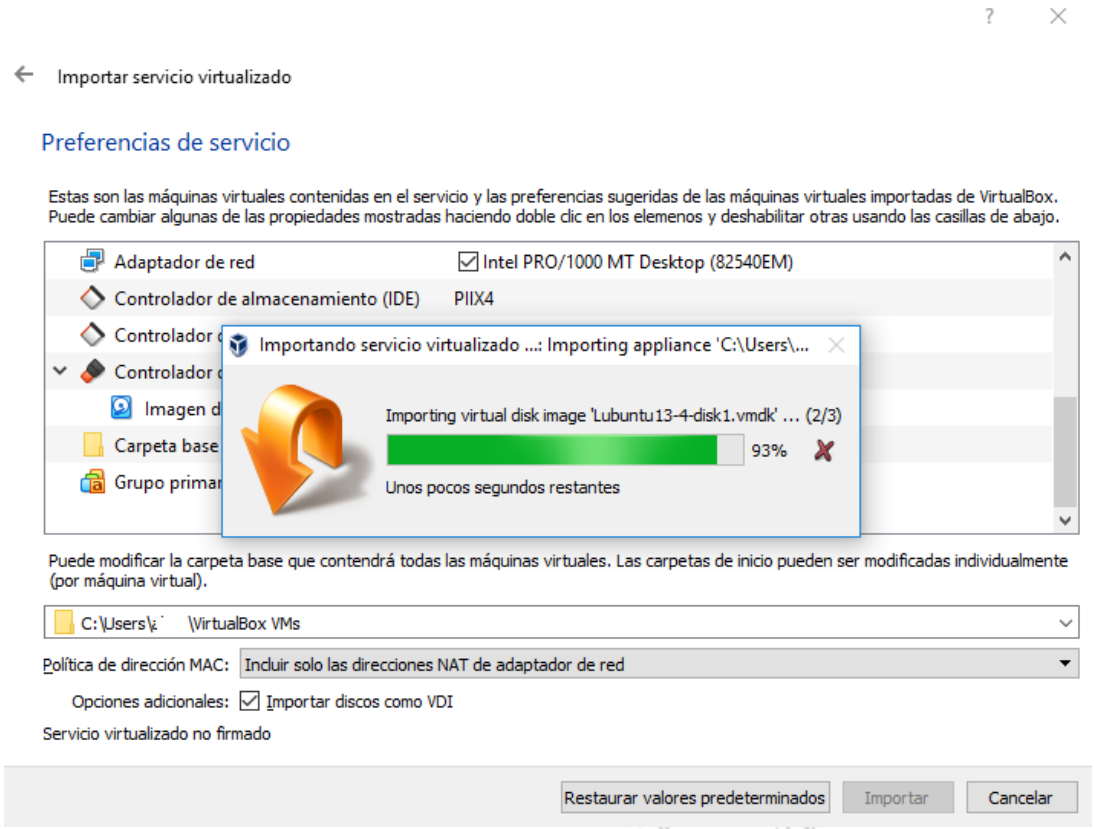

Figura 2.45. Importación de la máquina virtual.

Máquina virtual importada

Una vez importada la máquina virtual, debería aparecer en la parte de la izquierda de la pantalla una nueva máquina virtual de nombre Lubuntu13-4 (figura 2.46.).

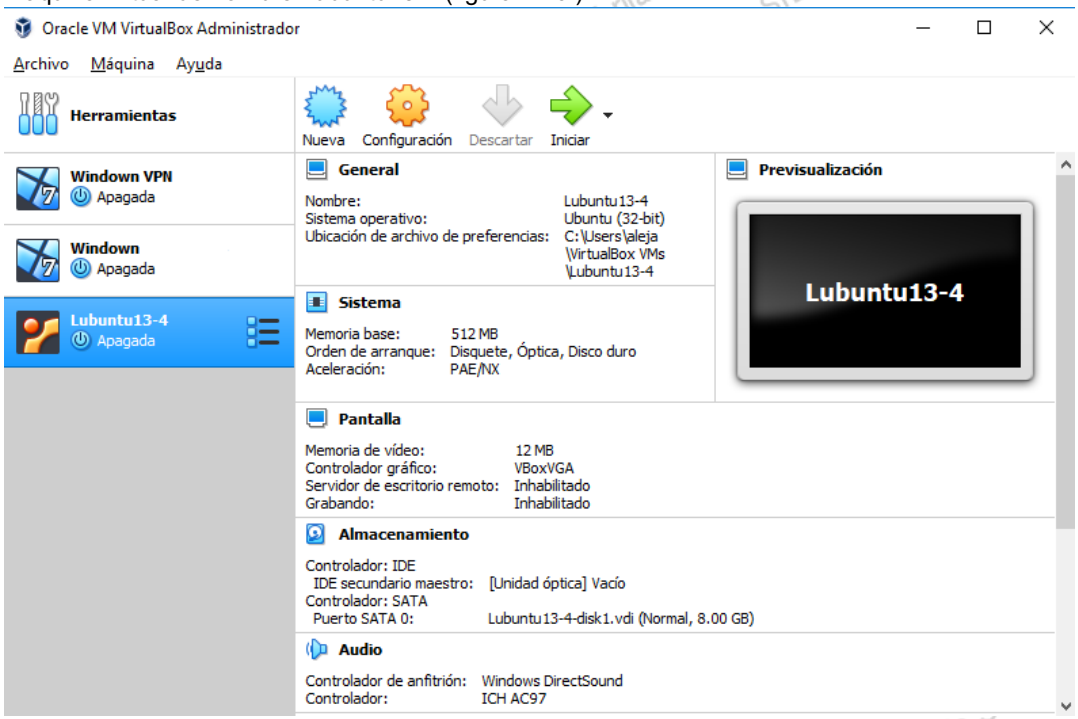

Figura 2.46. Máquina virtual importada.

Se pulsa sobre Lubuntu 13-4 y se selecciona la opción de “Configuración” (figura 2.47.).

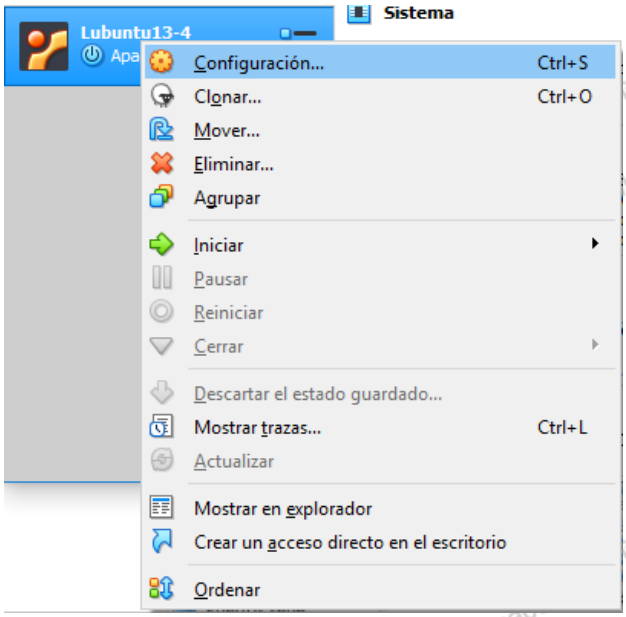

Figura 2.47. Configuración de la máquina virtual importada.

Memoria de la máquina virtual

A continuación, se selecciona en “Sistema” la opción “Placa Base” para fijar la cantidad de memoria de la máquina virtual (figura 2.48.). Dependiendo de la memoria RAM que tenga el ordenador que se está usando, se puede aumentar o disminuir el valor. 512 Mb debería ser suficiente, pero se recomienda 1024 Mb para que funcione con fluidez.

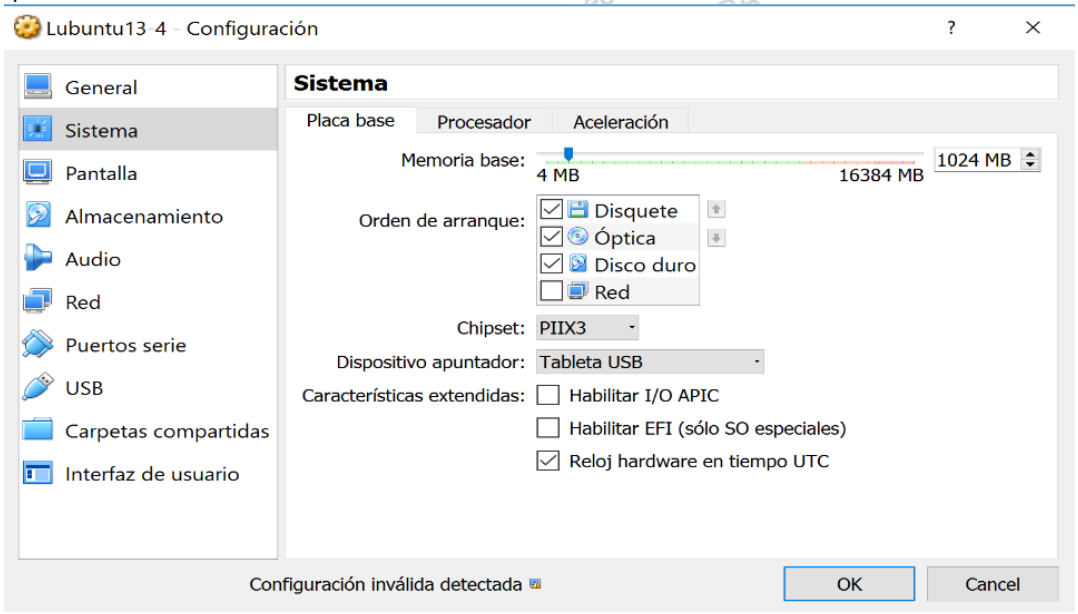

Figura 2.48. Configuración de la Memoria RAM.

Se selecciona la opción “Pantalla” y, en esta pestaña, se ajusta la cantidad de memoria para la tarjeta gráfica: 32 Mb debería ser suficiente (figura 2.49.).

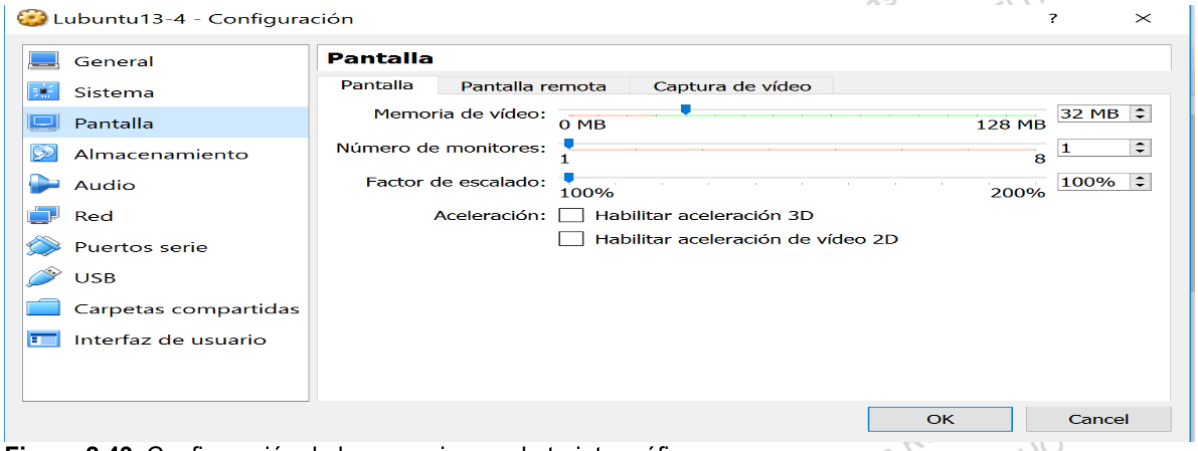

Figura 2.49. Configuración de la memoria para la tarjeta gráfica.

"Iniciar"

A continuación, se pulsa en el botón “OK”. En la pantalla principal de Virtual Box, se selecciona la máquina virtual “Lubuntu 13-4” y se pulsa en el botón “Iniciar” (figura 2.50.).

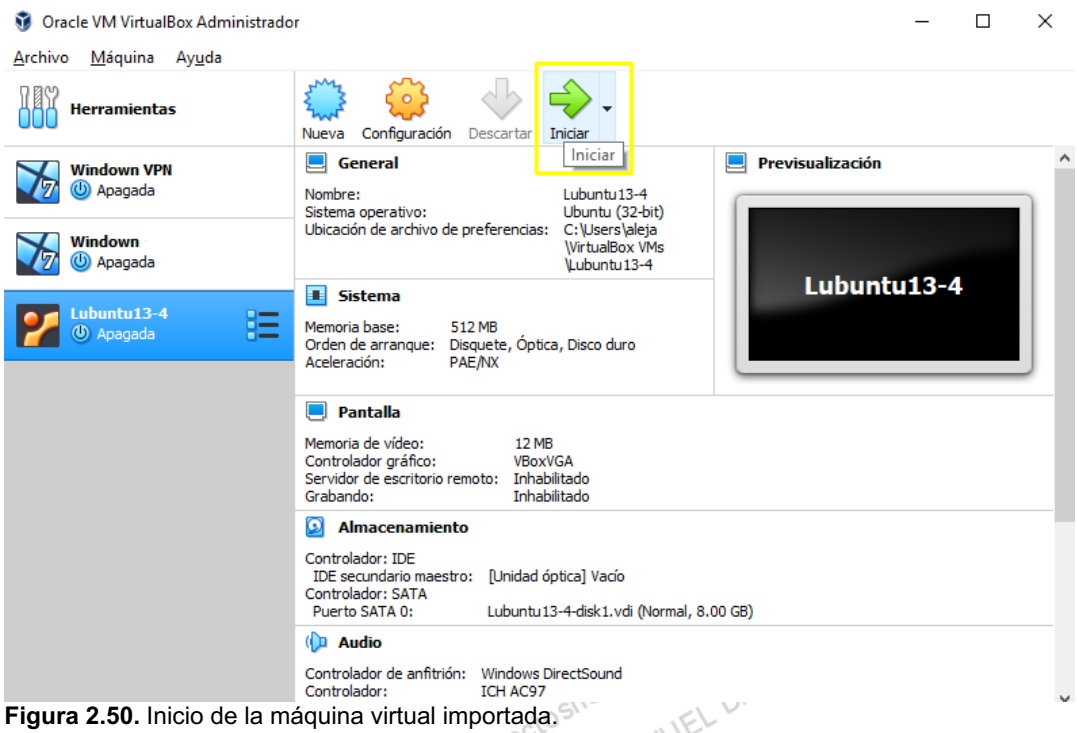

Figura 2.50. Inicio de la máquina virtual importada.

A continuación, se ejecuta la máquina virtual “Lubuntu 13-4” y ya se podrá trabajar con ella. Se selecciona la cuenta “Guest Account” y se pulsa sobre “Login” (fgura 2.51).

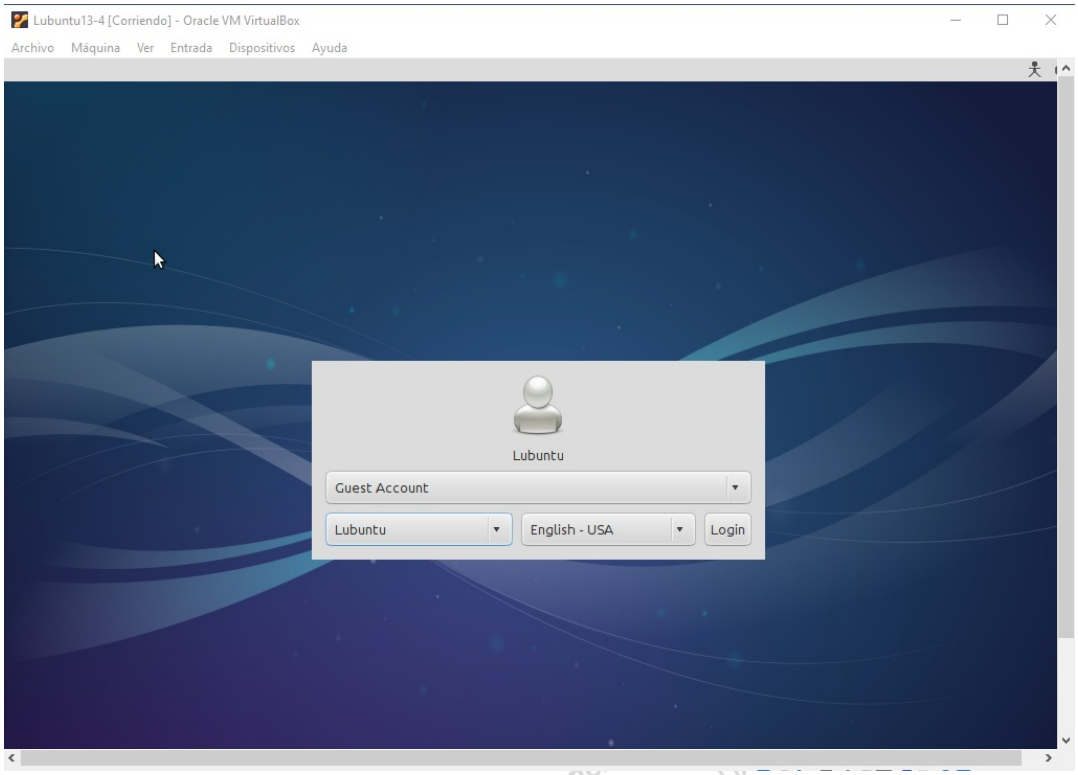

Figura 2.51. Login en la máquina virtual importada.

A continuación, aparece el escritorio del sistema operativo virtualizado (figura 2.52.).

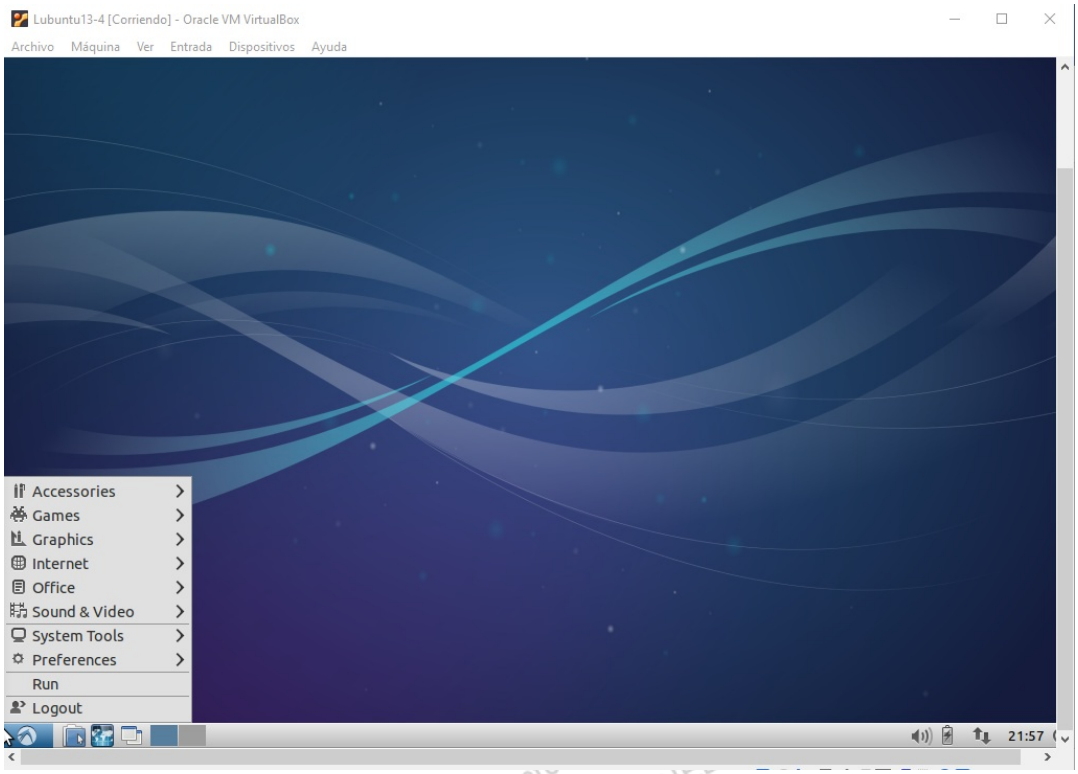

Figura 2.52. Escritorio de la máquina virtual importada.

Algunas observaciones que se deben tener en cuenta:

1
Para que funcionen las máquinas virtuales es necesario que el PC permita la "Virtualización Hardware". Esta es una opción que suele encontrarse en la BIOS del equipo. Puede llamarse "Virtualización Hardware" o similar en función del modelo del PC.
2
Para confirmar que el PC soportará la virtualización, se puede usar la siguientepágina.
3
Si la importación da un error estilo ERROR_ZIP_CODE, muy posiblemente el archivo .ova esté corrupto. Para solucionarlo, se debe descargar de nuevo la máquina virtual.
4
Una vez importada la máquina, si al intentar encenderla no lo hiciera, se debe revisar el punto 1.
5
También es importante que la máquina virtual esté preparada para el sistema operativo anfitrión, es decir, saber si es de 64 bits o 32 bits.
6
Si al iniciar la máquina virtual aparece la pantalla con rayas negras o de colores, pueden verificar la solución del problema en el sitio web YouTube.[3]

[3] Beamtic. *Install Ubuntu 16.04 in VirtualBox / fix Display Corruption* YouTube, 29 de noviembre de 2016.

Se acaba de describir cómo cargar una máquina virtual mediante la opción de “Importar un servicio virtualizado”, puesto que la máquina se encontraba en un archivo de tipo .ova. Sin embargo, las máquinas virtuales pueden distribuirse en otros formatos diferentes.

A continuación, se muestra cómo importar una máquina virtual que se encuentra en un archivo de tipo .vmdk.

1

Para ello, se pulsa en la opción “Nueva” (figura 2.53.).

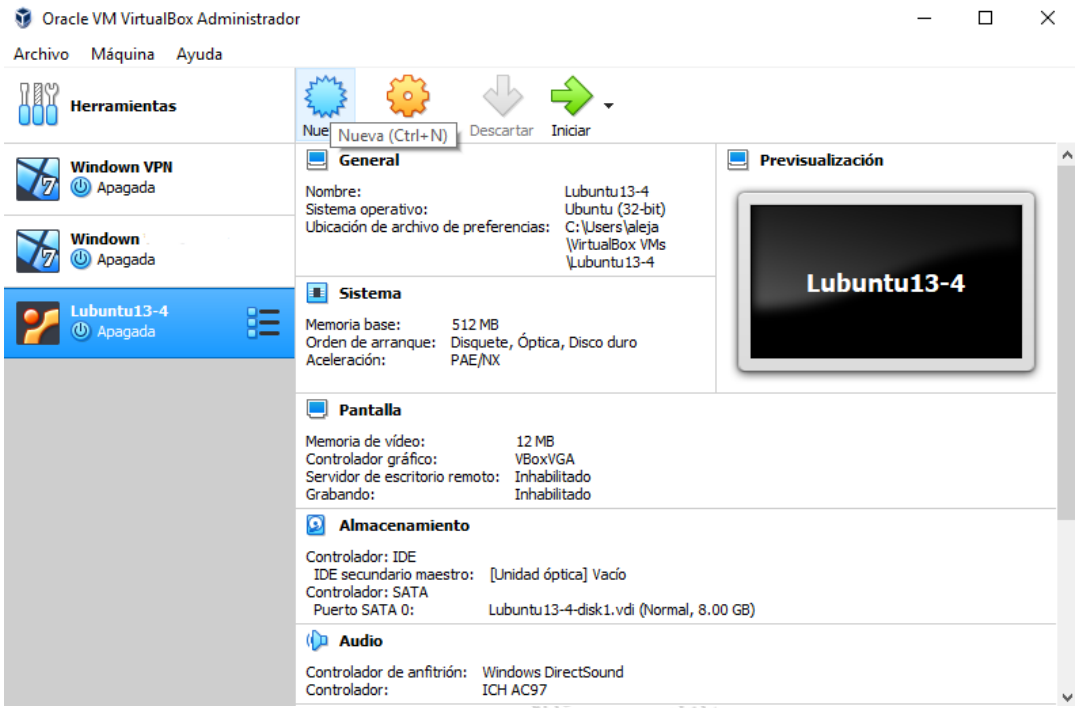

Figura 2.53. Importación de una máquina virtual tipo .vmdk

2

Aparece una nueva ventana en la que se debe configurar la máquina virtual. Se rellena la pestaña “Nombre” con uno cualquiera (por ejemplo, “Máquina 3”), en “Tipo” se selecciona “Linux” y en “Versión” se selecciona “Ubuntu (64-bit)”, tal como se ve en la figura 2.54.

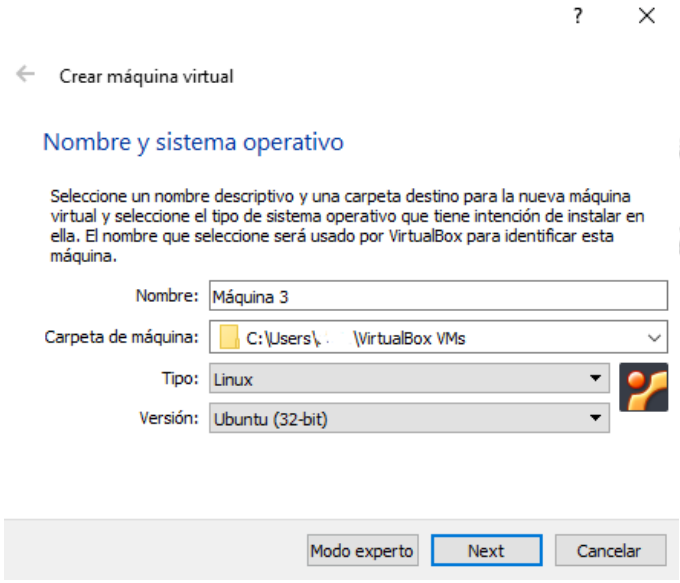

Figura 2.54. Nombre de la máquina virtual.

3

Se pulsa sobre “Next” y, en la ventana siguiente, se configura el tamaño de la memoria. 1024 Mb es un buen valor (figura 2.55.).

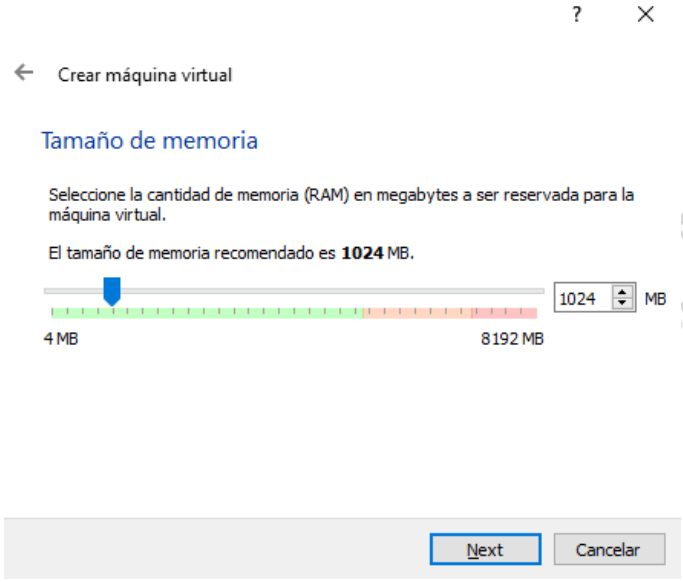

Figura 2.55. Configuración de la memoria.

A continuación, se pulsa sobre “Next” y en la ventana siguiente se selecciona la opción “Usar un archivo de disco duro virtual existente” y se pulsa sobre el icono en forma de carpeta (figura 2.56.).

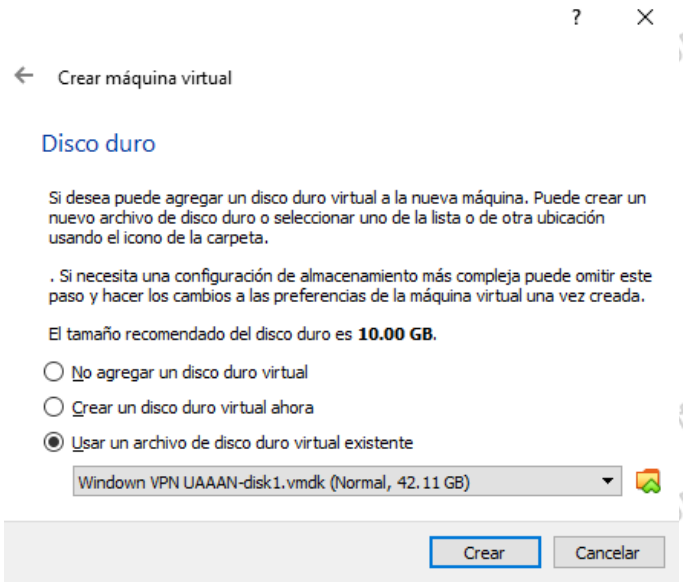

Figura 2.56. Selecciona “Usar disco ya existente”.

4

Cuando se pulsa sobre el icono de la carpeta,se pulsa en el icono agregar yaparece un navegador de archivos, en el que se buscará la máquina que se quiere cargar (figura 2.57.).

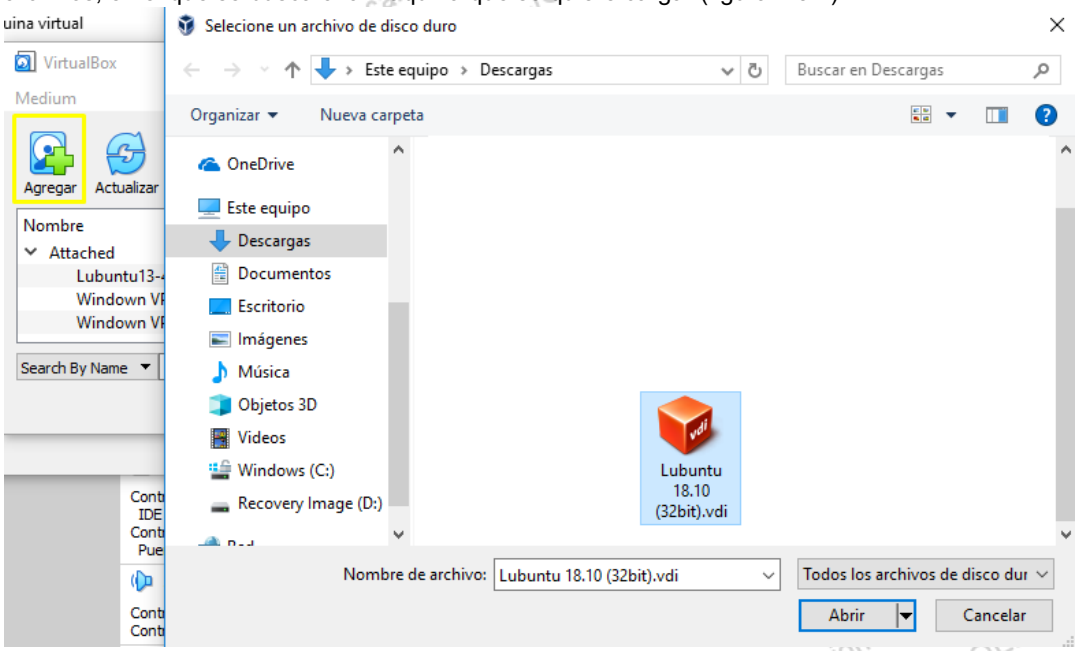

Figura 2.57. Búsqueda de la máquina virtual

5

Se pulsa sobre “Abrir” y se vuelve a la ventana anterior (figura 2.58.).

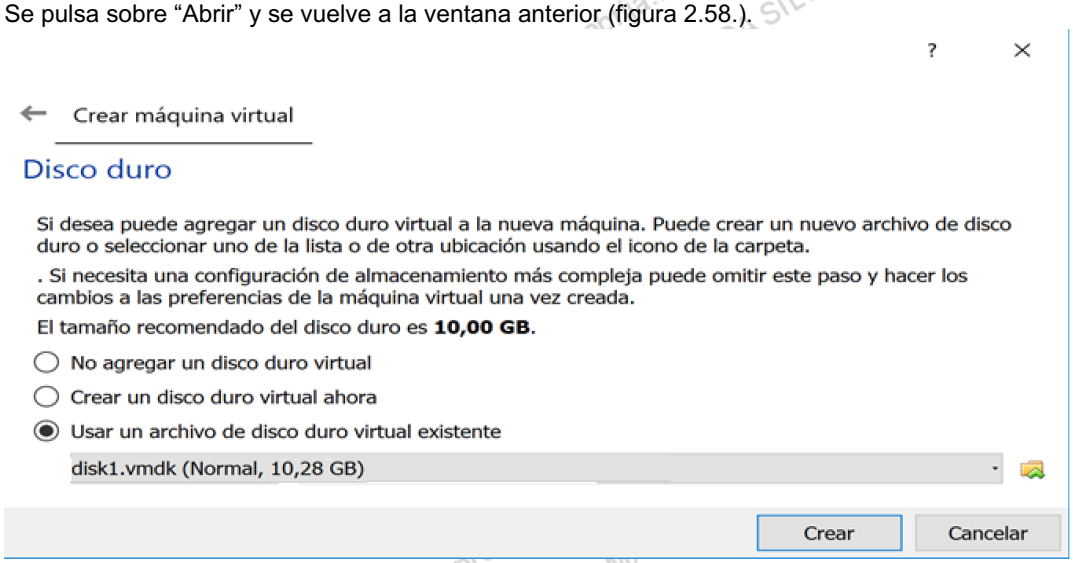

Figura 2.58. Máquina virtual seleccionada.

6

A continuación, se pulsa sobre “Crear” y aparecerá en el listado de máquinas virtuales (figura 2.59.).

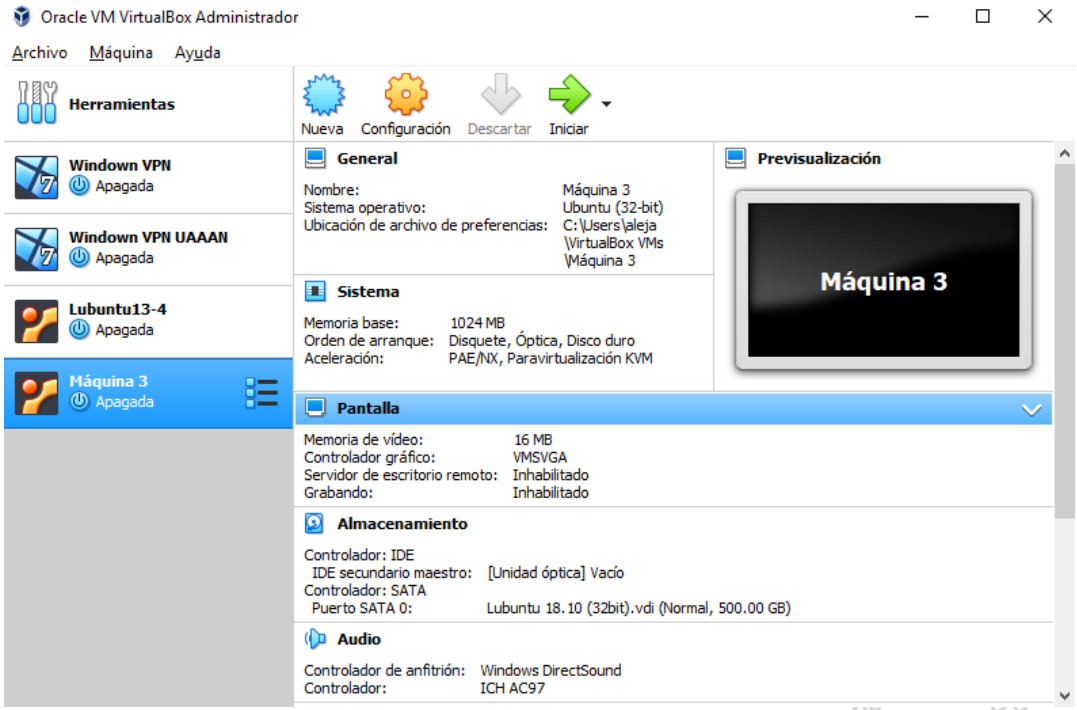

Figura 2.59. Máquina virtual importada.

VI. La Shell de comandos de Linux. Creación de scripts

6.1. La Shell de comandos

Linux es un sistema operativo similar a Unix que se distribuye como software libre. Para ilustrar los contenidos de esta sección, se va a utilizar una distribución de Linux denominada Lubuntu que se caracteriza por exigir pocos recursos hardware para su ejecución.

Linux dispone de una aplicación denominada intérprete de comandos o Shell de comandos que permite interactuar al usuario con el sistema operativo mediante la ejecución de comandos o sentencias. Hay diferentes tipos de intérpretes que se diferencian en la sintaxis de los comandos y en la forma de interaccionar con el usuario.

Para acceder al intérprete de comandos, se pulsa sobre Ctrl+Alt+T o bien se pulsa en un icono que aparece en la parte inferior izquierda, dentro de la barra de herramientas inferior del escritorio de Lubuntu (figura 2.60.).

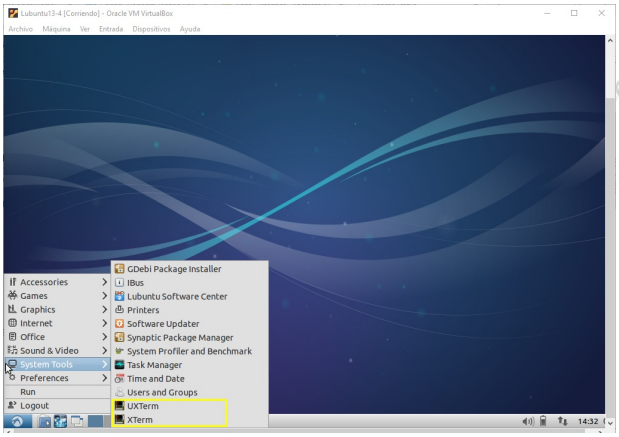

Figura 2.60. Acceso al intérprete de comandos de Linux.

Algunas características generales son:

El terminal muestra en pantalla un indicador de línea de órdenes esperando que el usuario introduzca una. El indicador finaliza con un carácter $ en el caso de usuarios normales y con # en el caso del superusuario.

Al comienzo de la línea de órdenes aparece el usuario de la forma usuario@nombre: ~$ (figura 2.61.)

Cuando se escribe un comando para que se ejecute, hay que pulsar la tecla “Enter”.

Los comandos hay que teclearlos exactamente. En este sentido, las letras mayúsculas y minúsculas se consideran caracteres diferentes.

El terminal siempre está dentro de una carpeta específica y puede navegar hasta otras carpetas y gestionar los archivos. Siempre se abre desde la carpeta personal del usuario.

A continuación, se van a revisar los **principales comandos**:

Las imágenes pueden visualizarse ampliadas clicando sobre las mismas.

Cambio de directorio:

Para cambiar de directorio se usa el comando cd directorio_destino. En particular para ir al directorio superior se usa cd .. (figura 2.62.) y cd sin argumentos para volver a la carpeta personal.

Figura 2.62. Cambio de directorio.

Situación actual:

El comando pwd imprime la ruta del directorio actual (figura 2.63.).

Figura 2.63. Uso del comando pwd.

Listado del contenido de un directorio:

El comando ls muestra los nombres de los archivos y subdirectorios contenidos en el directorio en el que se está (figura 2.64.). Solo se obtienen los nombres de los archivos, sin ninguna otra información.

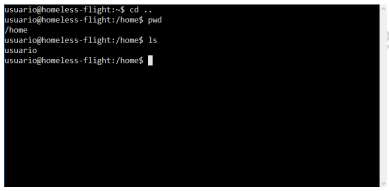

Figura 2.64. Uso del comando ls.

Admite un conjunto de opciones

- **ls -R** lista recursivamente los subdirectorios encontrados.
- **ls -a** muestra todos los archivos, incluyendo algunos que están ocultos para el usuario —aquellos que comienzan por un punto—. El archivo punto . indica el directorio actual y el doble punto .. el directorio padre, que contiene al actual.
- **ls -c** muestra ordenando por día y hora de creación.
- **ls -t** muestra ordenando por día y hora de modificación.
- **ls -r** muestra el directorio y lo ordena en orden inverso.
- **ls subdir** muestra el contenido del subdirectorio subdir.
- **ls -l filename** muestra toda la información sobre el archivo.
- **ls -color** muestra el contenido del directorio coloreado: verde para los ejecutables, azul para las carpetas, fucsia para las imágenes, rojo para los comprimidos, etc.
- **ls -l** muestra toda la información de cada archivo, incluyendo protecciones, tamaño y fecha de creación o del último cambio introducido, etc.
- Las opciones anteriores pueden combinarse. Por ejemplo, **ls -cr** muestra el directorio ordenando inversamente por fechas (figura 2.65.).

El comando **ls** admite los caracteres de sustitución ***** que representa cualquier conjunto o secuencia de caracteres y **?** que representa cualquier carácter, pero solo uno. Por ejemplo, **ls *.gif** muestra todos los nombres de archivos que acaben en **.gif** y **ls** file? muestra todos los archivos cuyos nombres empiecen por file y tengan un nombre de cinco caracteres.

Existe otro comando denominado **dir** que tiene la misma función que **ls**, pero sin mostrar tanta información.

Creación de ficheros

El comando touch actualiza los registros de fecha y hora con la fecha y hora actual de los ficheros indicados como argumento. Si el fichero no existe, el comando touch lo crea. Su uso más frecuente es para crear archivos.

La sintaxis del comando touch sigue la forma: touch fichero

Además, touch ejemplo.txt crea el archivo ejemplo.txt en el directorio actual, si este no existiera.

Creación de subdirectorios

El comando **mkdir** permite crear un nuevo subdirectorio. La sintaxis es**mkdir subdir1** donde subdir es el nombre del directorio que se va a crear (figura 2.66.).

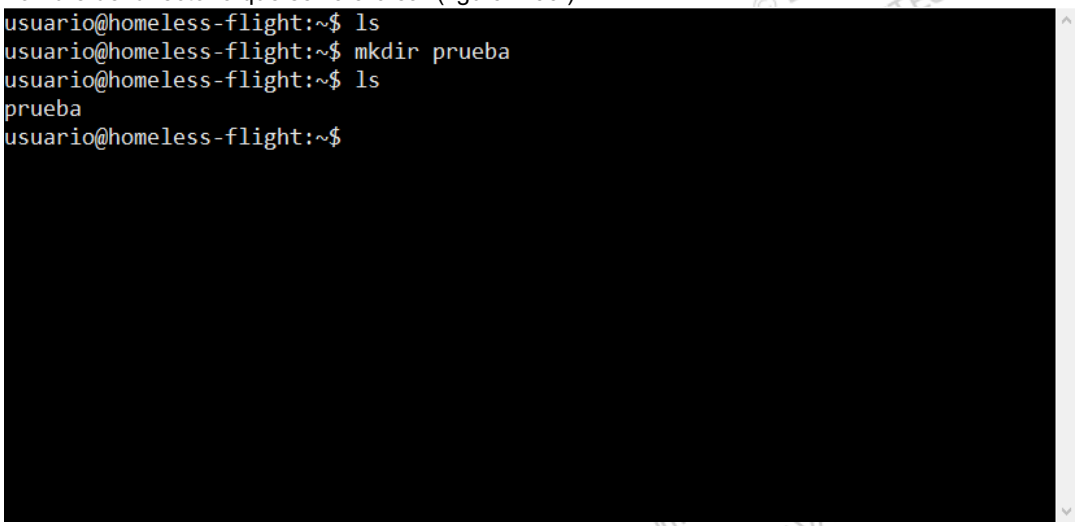

Figura 2.66. Creación de un subdirectorio.

Borrado de subdirectorios

El comando borra uno o más directorios del sistema siempre que estos subdirectorios estén vacíos. La sintaxis es **rmdir subdir1** donde subdir es el nombre del directorio que se va a eliminar (figura 2.67.).

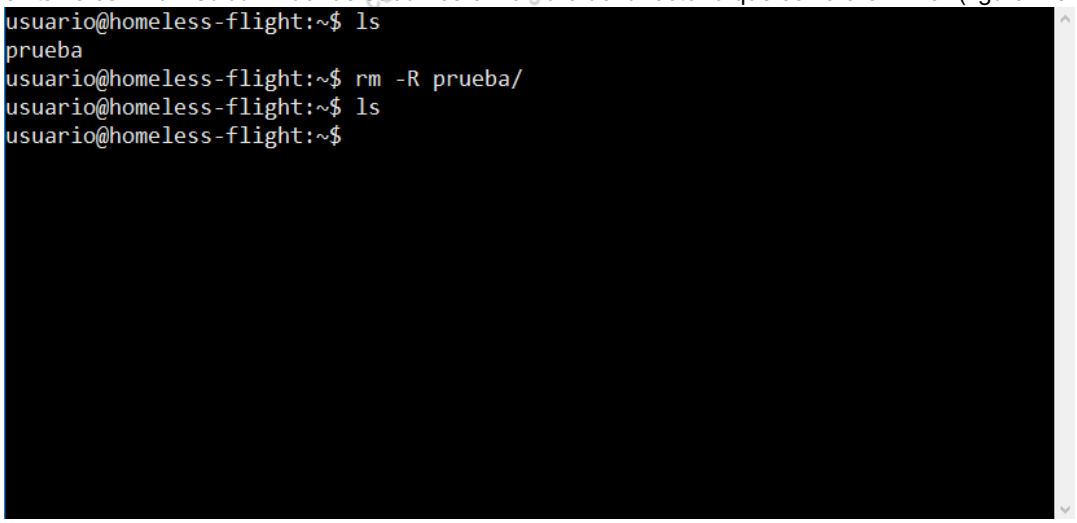

Figura 2.67. Borrado de un subdirectorio.

Copia de archivos

La sintaxis del comando es **cp file1 file2** (figura 2.68.) donde file1 y file2 indican el nombre de un archivo o la ruta al mismo. Su ejecución hace una copia de file1 y lo llama file2. Si file2 no existía, lo crea con los mismos atributos de file1 y en caso de existir su contenido es sustituido por el de file1. El archivo file2 estará en el mismo directorio que file1.

Figura 2.68. Copia de un archivo.

También se puede usar como **cp file1 file2 namedir** que hace copias de file1 y file2 en el directorio namedir (figura 2.69.).

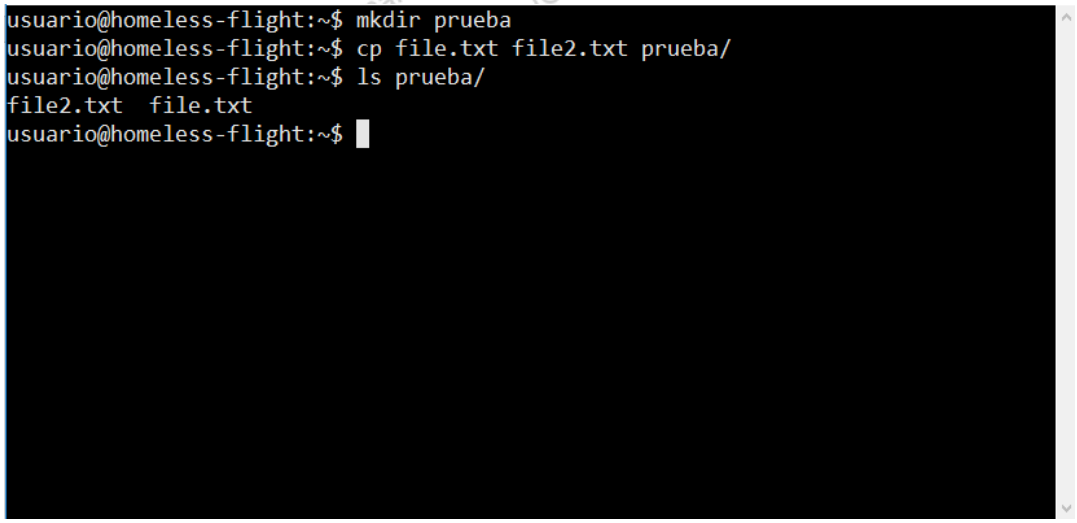

Figura 2.69. Copia de un archivo en un directorio determinado.

Traslado y cambio de nombre de archivos

La sintaxis es **mv file1 file2** y realiza la misma función que cp, pero eliminando el archivo original. Desde la visión del usuario, se cambia el nombre a file1 por file2 (figura 2.70.).

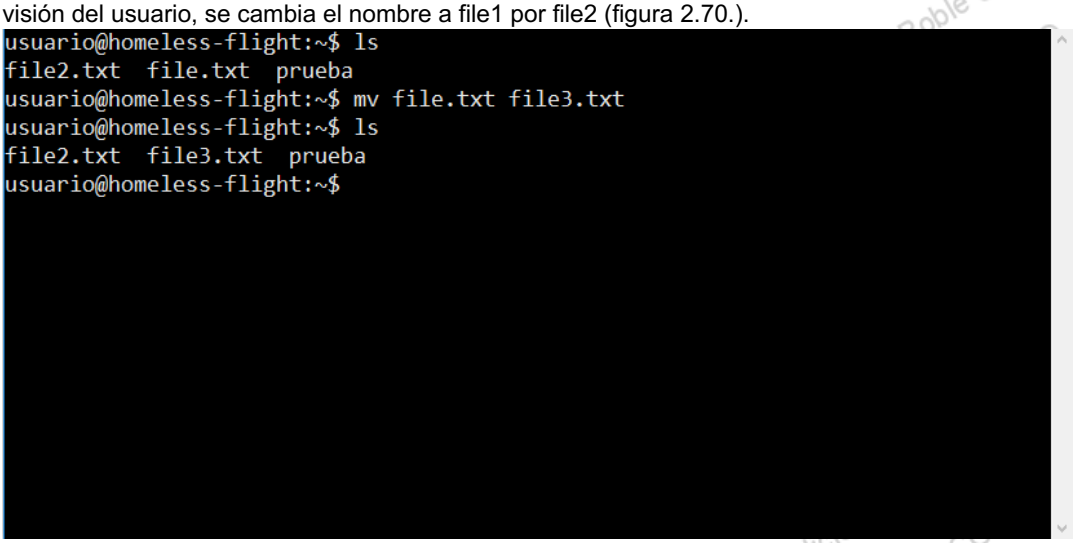

Figura 2.70. Cambio del nombre de un archivo.

Si los nombres que aparecen son de directorios, entonces el comando**mv namedir1 namedir2** cambia el nombre del subdirectorio namedir1 por namedir2.

mv file1 file2 namedir traslada uno o más archivos (file1, file2, etc.) al directorio namedir conservando el nombre.

Borrado de archivos

El comando **rm** elimina uno o más archivos de un directorio en el cual tengamos permiso de escritura. La sintaxis es **rm file1 file2** (figura 2.71.). En el comando, se pueden utilizar los caracteres de sustitución* y **?**

```
usuario@homeless-flight:~$ ls
file2.txt  file.txt  prueba
usuario@homeless-flight:~$ mv file.txt file3.txt
usuario@homeless-flight:~$ ls
file2.txt  file3.txt  prueba
usuario@homeless-flight:~$ rm file3.txt
usuario@homeless-flight:~$ ls
file2.txt  prueba
usuario@homeless-flight:~$
```

Figura 2.71. Borrado de un archivo.

Si se usa la opción **-i**, se pide confirmación para borrar cada archivo de la lista especificada:**rm -i file1 file2** (figura 2.72.).

```
usuario@homeless-flight:~$ ls
file2.txt  prueba
usuario@homeless-flight:~$ rm -i file2.txt
rm: remove regular file 'file2.txt'? y
usuario@homeless-flight:~$ ls
prueba
usuario@homeless-flight:~$
```

Figura 2.72. Borrado de un archivo con opción **-i**.

Enlaces a archivos

Un mismo archivo puede estar repetido con más de un nombre. También es posible tener un mismo archivo con varios nombres distintos y poder acceder a él desde más de un directorio. Esto último se denomina enlaces múltiples a un archivo y la forma de crearlo es mediante el comando **ln: ln file1 file2**.

De esta forma el archivo file1 tiene dos nombres: file1 y file2. Además, este comando advierte previamente si el nombre file2 está ocupado y en este caso no se ejecuta.

En el siguiente ejemplo se crea un enlace al archivo prueba.txt (figura 2.73.).

```
usuario@homeless-flight:~$ ls
prueba  prueba.txt
usuario@homeless-flight:~$ ln prueba.txt prueba2.txt
usuario@homeless-flight:~$ ls
prueba  prueba2.txt  prueba.txt
usuario@homeless-flight:~$ ls -lt
total 12
-rw-rw-r-- 2 usuario usuario    6 Feb 18 18:11 prueba2.txt
-rw-rw-r-- 2 usuario usuario    6 Feb 18 18:11 prueba.txt
drwxrwxr-x 2 usuario usuario 4096 Feb 18 17:57 prueba
usuario@homeless-flight:~$
```

Figura 2.73. Enlace a un archivo.

Los archivos enlazados a otro se borran como los archivos normales, de manera que si se borra el archivo original permanece su contenido en los archivos enlazados.

Características de un archivo

Mediante el comando **file** se puede obtener información acerca de un archivo:**file fich** (figura 2.74.).

```
usuario@homeless-flight:~$
usuario@homeless-flight:~$ file prueba.txt
prueba.txt: ASCII text
usuario@homeless-flight:~$
```

Figura 2.74. Características de un archivo.

Cambio de modo en los archivos

En Linux, los archivos tienen asociados permisos que indican las operaciones que le está permitido realizar a un usuario sobre los mismos. Los permisos de cada archivo se pueden ver con el comando **ls -l**. Para cambiar los permisos de un archivo se emplea el comando **chmod** [quien] **oper**, **permiso**, **files** donde:

quien

Indica a quién afecta el permiso que se desea cambiar. Es una combinación de las letras **u** para el usuario, **g** para el grupo del usuario, **o** para los otros usuarios, y **a** para todos los anteriores. Si no se indica nada, se supone **a**.

oper

Indica si el permiso se da usando un **+** o se quita usando un **-**.

permiso

Indica el permiso que se quiere dar o quitar. Es una combinación de las letras: **r** (leer), **w** (escribir), **x** (ejecutar). Estos permisos son diferentes cuando se aplican a directorios: **r** da la posibilidad de ver el contenido del directorio con el comando **ls**; el permiso **w** da la posibilidad de crear y borrar archivos en ese directorio; el permiso **x** autoriza a buscar y utilizar un archivo concreto.

files

Nombres de los archivos o directorios cuyos modos de acceso se quieren cambiar.

En el siguiente ejemplo, se cambian los permisos al archivo prueba.txt (figura 2.75.)

```
usuario@homeless-flight:~$ ls -lt
total 12
--w--w---- 2 usuario usuario    6 Feb 18 18:11 prueba2.txt
--w--w---- 2 usuario usuario    6 Feb 18 18:11 prueba.txt
drwxrwxr-x 2 usuario usuario 4096 Feb 18 17:57 prueba
usuario@homeless-flight:~$ chmod +x prueba.txt
usuario@homeless-flight:~$ ls -lt
total 12
--wx-wx--x 2 usuario usuario    6 Feb 18 18:11 prueba2.txt
--wx-wx--x 2 usuario usuario    6 Feb 18 18:11 prueba.txt
drwxrwxr-x 2 usuario usuario 4096 Feb 18 17:57 prueba
usuario@homeless-flight:~$
```

Cambio de propietario a un conjunto de archivos

El comando **chown** se emplea para cambiar de propietario a un determinado conjunto de archivos:**chown newowner file1 file2**...

Solo lo puede emplear el actual propietario de los mismos. Los nombres de propietario se encuentran almacenados en el archivo /etc/passwd.

Cambio de grupo de un archivo

El comando **chgrp** se emplea para cambiar el grupo al que pertenece un archivo:**chgrp newgroup file1 file2...**

Los grupos de usuarios están almacenados en el archivo /etc/group.

Visualización sin formato de archivos

El comando **cat filename** permite visualizar el contenido de uno o más archivos de forma no formateada y también permite copiar uno o más archivos como apéndice de otro ya existente:

cat filename: saca por pantalla el contenido del archivo filename.

cat file1 file2…: saca por pantalla, secuencialmente y según el orden especificado, el contenido de los archivos indicados.

cat file1 file2 >file3: El contenido de los archivos file1 y file2 es almacenado en file3.

cat file1 file2 >>file3: el contenido de file1 y file2 es añadido al final de file3.

cat >file1: Acepta lo que se introduce por el teclado y lo almacena en file1 (se crea file1). Para terminar, se emplea <ctrl>d

Alternativamente, se puede usar el editorvi

El editor puede encontrarse en dos estados o modos

Comandos

En el modo de comandos, vi está esperando alguna orden (por tanto, interpreta lo que se escribe como órdenes).

Edición

En el modo de edición, vi está esperando que se escriba el texto del fichero (por tanto, interpreta lo escrito como texto).

Cuando se entra en vi, está en modo de comandos

Para pasar al modo de edición, se puede pulsar i (insertar), a (añadir), o (añadir una línea). Para pasar al modo de comandos, se puede pulsar Escape o Suprimir.

Algunos comandos de vi son

- i insertar antes del cursor
- a añadir detrás del cursor
- o añadir una línea en blanco
- x borrar un carácter
- j borrar el final de línea (une dos líneas)
- dd borra la línea completa
- u deshacer la última edición
- :q salir
- :q! salir sin guardar
- :w guardar
- :wq guardar y salir
- :set nu muestra números de línea
- :set nonu oculta números de línea

Visualización de archivos con formato

Este comando **pr file** imprime por consola el contenido de los archivos de una manera formateada, por columnas, controlando el tamaño de página y poniendo cabeceras al comienzo de las mismas:

pr file

pr file produce una salida estándar de 66 líneas por página, con un encabezamiento de 5 líneas: 2 en blanco, una de identificación y otras 2 líneas en blanco.

pr -ln

pr -ln file produce una salida de n líneas por página.

pr -F

pr -F file hace una pausa para presentar la página, hasta que se pulsa <return> para continuar.

pr -t file

pr -t file suprime las 5 líneas del encabezamiento y las del final de página.

pr -wn file

pr -wn file ajusta la anchura de la línea a n posiciones.

pr -d file

pr -d file lista el archivo con espaciado doble.

pr -h `caracteres` file

pr -h `caracteres` file, el argumento o cadena de caracteres `caracteres` se convertirán en la cabecera del listado.

pr +n file

pr +n file imprime el archivo a partir de la página n.

Se pueden combinar varias opciones en un mismo comando, como por ejemplo**pr -dt file** o cambiar la salida con **pr file1 > file2**, que crea un archivo nuevo llamado file2 que es idéntico a file1, pero con formato por páginas y columnas.

Visualización de archivos pantalla a pantalla

Los comandos **more** y **less** permiten visualizar un archivo pantalla a pantalla. El número de líneas por pantalla es de 23 de texto más una última línea de mensajes, donde aparecerá la palabra **more**. Cuando se pulsa la barra espaciadora se visualizará la siguiente pantalla. Para salir de este comando se pulsa **<ctrl>d** o **q**. La sintaxis es **more file**.

El comando **less** permite el desplazamiento a lo largo del texto empleando las teclas de cursores pudiendo ir hacia arriba o abajo de un archivo. La sintaxis es **less file**.

Búsqueda en archivos

El comando **grep 'conjuntocaracteres' file1 file2 file3** busca una palabra, clave o frase en un conjunto de directorios, indicando en cuáles de ellos la ha encontrado. Busca archivo por archivo, por turno, imprimiendo aquellas líneas que contienen el conjunto de caracteres buscado. Si el conjunto de caracteres está compuesto por dos o más palabras separadas por un espacio, se escribe entre apóstrofes **'**.

Se pueden utilizar expresiones regulares de la forma:**grep [-opcion] expresión_regular [referencia...]**

Las opciones que admite son:

c escribe el número de las líneas que satisface la condición.

i, no se distinguen mayúsculas y minúsculas.

l escribe los nombres de los archivos que contienen líneas buscadas.

n, cada línea es precedida por su número en el archivo.

s, no se vuelcan los mensajes que indican que un archivo no se puede abrir.

v, se muestran solo las líneas que no satisfacen el criterio de selección.

Algunos ejemplos:

- **grep ‘^d’ text** recupera las líneas que comienzan por d.
- **grep ‘^[^d]’ text** recupera las líneas que no comienzan por d.
- **grep -v ‘^C’ file1 > file2** quita las líneas de file1 que comienzan por C y lo copia en file2.

Impresión

El comando **lpr nombre_archivo** imprime por defecto el archivo indicado. Si se usa sin argumentos imprime el texto que se introduzca a continuación en la impresora.

Compresión de archivos

El comando **tar -cvf nombre_archivo.tar archivo1 archivo2…** agrupa varios archivos en uno solo archivo tar. Con **gzip archivo** se comprime archivo (que es borrado) y se crea un archivo con nombre archivo.gz.

Se pueden realizar las operaciones inversas, de manera que se pueden extraer los archivos mediante **tar -xpvf nombre_archivo.tar archivo1…** y descomprimir un archivo mediante **gzip -d archivo.gz**.

Dado que suele emplearse tar y gzip de forma consecutiva obteniéndose archivos con extensión tar.gz o tgz, entonces tar incluye la opción z para extraer archivos con esta extensión **tar -zxf archivo.tar.gz**.

Redirecciones

Los comandos tienen una entrada estándar (número 0) y dos salidas estándar (número 1 para la salida normal y número 2 para la salida con errores). Por defecto, la entrada y la salida estándar de un comando es la terminal, a no ser que el comando especifique los nombres de archivos que hagan de entrada y de salida, como por ejemplo el comando **cp file1 file2**.

Sin embargo, se puede redirigir la salida de un comando usando los operadores:

(**>**) redirige la salida estándar hacia el archivo indicado y en caso de no existir se crea.

(**<**) redirige la entrada estándar desde un determinado archivo.

(**>>**) redirige la salida estándar hacia otro archivo, pero añadiendo dicha salida al final de ese archivo, sin sobrescribir el contenido original.

- **date >>archivo** el archivo **archivo** contendrá información sobre todas las veces que se ha entrado en el sistema.
- **cat file1 file2 >file3** añade al archivo file2 al final de file1 y al conjunto lo llama file3.
- **cat file2 >>file1** realiza la misma operación que el anterior, pero al resultadolo llama file1.

Tuberías

Una tubería (**|**) permite comunicar la salida estándar de un comando con la entrada estándar de otro. Por ejemplo, **ls | mail juan** envía a juan una lista de los archivos del sistema. Con el operador de tubería se pueden empalmar tantos comandos como se desee.

Bifurcación

Permite redirigir la salida de un comando a un determinado archivo y que también se bifurque hacia la terminal. Para ello se usa el operador **tee,** como por ejemplo **ls | tee file**, que muestra la salida por el terminal y además la envía al archivo file. En este ejemplo, si se quiere que la salida se añada al final de file, se usaría la opción **-a**: **ls | tee -a file**.

Redirección de los errores

Los mensajes de error se dirigen a la salida número 2, que normalmente es también la terminal. En algunos casos puede interesar redirigir estos mensajes a otra salida como, por ejemplo, cuando se ejecuta un programa en segundo plano. Véanse a continuación los siguientes ejemplos:

- **gcc prueba.c 2>errores** se redirige la salida 2 hacia el archivo errores.
- **program <datos.d >resultados.r 2> & 1** se redirige la salida estándar de errores al mismo archivo que la salida estándar.

Ejecución de un programa

Existen varios comandos para gestionar la ejecución de un programa:

El carácter **&** permite realizar una ejecución en segundo plano recuperando inmediatamente el control del terminal. Para ello se añade el carácter **&** al final del comando de ejecución **program <datos.d >resultados.r** &. Como resultado aparece en el terminal el número de procesode la ejecución de este programa.

Para detener la ejecución de un proceso se puede utilizar el comando **kill númerodeproceso**.

Cuando se sale del sistema, si hay algún proceso ejecutándose en segundo plano se para, salvo que se use el comando **nohup program**. En este caso si no se utilizan redirecciones todas las salidas del programa se dirigen a un archivo llamado **nohup.out**.

El comando **nice** permite realizar ejecuciones con baja prioridad de la forma:

- **nice program &**
- **nice nohup program &**

Para darle al programa la prioridad mínima habría que invocarlo como **nice -19 program &** donde el -19 indica la mínima prioridad.

El comando **time**, precediendo a cualquier otro comando, suministra información acerca del tiempo total empleado en la ejecución, del tiempo de CPU utilizado por el programa del usuario y del tiempo de CPU consumido en utilizar recursos del sistema. Por ejemplo, **time gcc prueba.c** ofrece información sobre el tiempo de compilación y montaje del programa prueba.c.

El comando **top** muestra una lista de los procesos que se están ejecutando. Sus principales opciones son **u,** que muestra los procesos que pertenecen a un determinado usuario), **k**, que elimina un proceso, y **h**, que muestra la ayuda del programa.

Otros comandos

En las siguientes tablas (Tabla 2.1.), se muestran otros comandos interesantes del intérprete de comandos de Linux.

Tabla 2.1. Comandos de Linux.

Comando	Significado
date	Muestra el día y la hora.
cal	Muestra el calendario. Tiene diversas opciones. Por ejemplo, cal 1945 mostraría el calendario del año 1945.
who	Indica qué usuarios tiene el ordenador en ese momento, en qué terminal están y desde qué hora.
whoami	Indica cuál es la terminal y la sesión en la que se está trabajando.
uptime	Indica cuál es la terminal y la sesión en la que se está trabajando.
hostname	Muestra el nombre de la máquina.
bc	Abre una calculadora de texto. Para salir se introduce **quit.**
lshw	Muestra todas las características del hardware.
history	Muestra los comandos usados por el usuario en orden cronológico.
fc -l	Muestra los últimos comandos utilizados por el usuario.
man comando	Muestra el manual de un comando con sus modificadores y argumentos.
lspci	Muestra los diferentes dispositivos PCI.
umask	Muestra los permisos con los que el usuario creará sus directorios por defecto.
uname -r	Muestra la version del kernel.
lsusb	Muestra información de los dispositivos que tengo conectados en los puertos USB.
xkill	Permite cerrar un programa bloqueado.
groups	Muestra los grupos del sistema a los que pertenece un usuario.
clear	Limpia la consola.
head	Indicando un número n y un nombre de fichero, muestra las n primeras líneas del fichero indicado.

6.2. Creación de scripts

Un guión o script es un archivo de texto que contiene comandos de la Shell que se interpretarán cuando se ejecute el script. Se caracterizan por:

1
Pueden tener cualquier nombre.
2
El archivo debe tener permiso de ejecución.
3
Para ejecutarlo se debe indicar el nombre del intérprete y el nombre del guión.

4
Las instrucciones del guión se procesan por orden, una por línea, de manera que los saltos de línea se interpretan como un “INTRO”.
5
Las órdenes podrían aparecer en una misma línea separadas por punto y coma.

Los guiones pueden tener una línea inicial que indica el tipo de intérprete que se quiere utilizar para ejecutar los comandos. Por ejemplo #!/bin/bash indica que debe usarse el intérprete Bourne-Again Shell.

Asimismo, un guión puede contener comentarios que se marcan con el carácter # al inicio del texto del comentario.

Los **principales elementos de un script** son:

Variables y parámetros

Para definir una variable se usa la estructura variable=valor donde variable es el nombre de la misma. Hay que tener en cuenta lo siguiente:

No puede haber un espacio entre el nombre de la variable, el signo = y el valor.
Si se desea que el valor contenga espacios, es necesario utilizar comillas.
Para recuperar el valor de una variable hay que anteponerle a su nombre el carácter $. Por ejemplo, para visualizar el valor de una variable echo $variable.

En el siguiente ejemplo, se define la variable Prueba=”ls -l” y a continuación se invoca con $Prueba (figura 2.76.):

```
usuario@homeless-flight:~$ test='ls -lt'
usuario@homeless-flight:~$ $test
total 12
--wx-wx--x 2 usuario usuario    6 Feb 18 18:11 prueba2.txt
--wx-wx--x 2 usuario usuario    6 Feb 18 18:11 prueba.txt
drwxrwxr-x 2 usuario usuario 4096 Feb 18 17:57 prueba
usuario@homeless-flight:~$
```

Figura 2.76. Ejemplo de variable.

Para eliminar una variable, se usa el comando **unset**. Por ejemplo (figura 2.77.):

```
usuario@homeless-flight:~$ test='ls -lt'
usuario@homeless-flight:~$ $test
total 12
--wx-wx--x 2 usuario usuario    6 Feb 18 18:11 prueba2.txt
--wx-wx--x 2 usuario usuario    6 Feb 18 18:11 prueba.txt
drwxrwxr-x 2 usuario usuario 4096 Feb 18 17:57 prueba
usuario@homeless-flight:~$ unset test
usuario@homeless-flight:~$ $test
usuario@homeless-flight:~$
```

Figura 2.77. Eliminación de una variable.

Existen algunas variables definidas en la Shell. En la siguiente tabla se muestran las principales variables (tabla 2.2.).

Tabla 2.2. Algunas variables predefinidas.

Nombre de variable	Significado
LOGNAME	Nombre del usuario.
HOME	Directorio de trabajo del usuario actual.
PATH	Caminos usados para ejecutar órdenes o programas.
PWD	Directorio activo.
TERM	El tipo de la terminal actual.
SHELL	Shell actual.
$?	Contiene el valor de salida de la última orden ejecutada. El valor de ‘0’ indica que no ha habido errores, mientras que otros valores indican errores.

Un guión puede recibir parámetros en la línea de órdenes para considerarlos durante su ejecución. Los parámetros recibidos se guardan en una serie de variables que el script puede consultar cuando lo necesite. Los nombres de estas variables son: $1 $2 $3 ... ${10} ${11} ${12} ... de manera que:

La variable $0 contiene el nombre con el que se ha invocado al script, es decir, el nombre del programa.

$1 contiene el primer parámetro.

$2 contiene el segundo parámetro.

. . .

A continuación, en el siguiente ejemplo de un guión Shell se muestran los valores de los parámetros (figura 2.78.).

Nombre de archivo: *script*

```
echo el nombre del programa es $0

echo el primer parámetro es $1

echo el segundo parámetro es $2
```

```
usuario@homeless-flight:~$ bash script parámetro1 parámetro2
el nombre del programa es script
el primer parámetro es parámetro1
el segundo parámetro es parámetro2
usuario@homeless-flight:~$
```

Figura 2.78. Ejemplo de usos de parámetros.

La orden shift mueve todos los parámetros una posición a la izquierda, esto hace que el contenido del parámetro $1 desaparezca y sea reemplazado por el contenido de $2, $2, que es reemplazado por $3, etc. Además, existen algunas variables especiales tales como $# que contienen el número de parámetros que ha recibido el script, y $* o $@ contienen todos los parámetros recibidos.

A continuación, se describen algunas reglas de la evaluación de variables:

$var o ${var}
$var o ${var}: representa el valor de la variable o nada si la variable no está definida.
${var-$val}
${var-$val}: devuelve el valor de var si está definida, si no val.
${var=val}
${var=val}: valor de var si está definida, si no val y el valor de var pasa a ser val.

${var?message}
${var?message}: devuelve $var si está definida y si no, imprime el mensaje en el terminal del Shell. Si el mensaje está vacío imprime uno estándar.

${var+$val}
${var+$val}: devuelve $var está definida, si no, nada.

El siguiente ejemplo muestra cómo podemos usar una variable, asignándole un valor en caso de que no esté definida (figura 2.79.).

Nombre de archivo: *script2*

```
echo el valor de var1 es $var1

echo el valor de var1 es $(var1=5)

echo el valor de var1 es $var1
```

```
usuario@homeless-flight:~$ bash script2
el valor de var1 es
el valor de var1 es 5
el valor de var1 es 5
usuario@homeless-flight:~$
```

Figura 2.79. Ejemplo de uso de parámetros

A continuación, en el siguiente script Shell, se muestra el directorio actual y el usuario logeado.

Nombre de archivo: *script3*

```
#!/bin/bash

echo "El directorio actual es:"

pwd

echo "Usuario logeado:"

#Comando que muestra el usuario que lo ejecuta

whoami
```

Comillas dobles y simples

Pueden usarse comillas simples o dobles cuando se asignan caracteres que contengan espacios y caracteres especiales. Sin embargo, forzosamente deben usarse comillas dobles si una variable hace referencia al valor de otra.

El siguiente script Shell concatena el valor de una variable con otra.

```
#!/bin/bash

var="Hola Linux"

var2="$var y Scripts de Shell"

echo $var
```

Usando comillas simples el siguiente script Shell no concatena el valor de una variable con otra.

```
#!/bin/bash

var='Hola Linux'

var2='$var y Scripts de Shell'

echo $var2
```

Arrays

Un array es una colección de elementos del mismo tipo, dotados de un nombre, y que se almacenan en posiciones contiguas de memoria. El primer elemento del array está numerado con el 0. No hay un tamaño límite para un array y la asignación de valores se puede hacer de forma alterna. La sintaxis para declarar un array es la siguiente:

Crear e inicializar un array: nombre_array=(val1 val2 val3 ...).

Asignar un valor al elemento x del array: nombre_array[x]=valor.

Acceder al elemento x: ${nombre_array[x]}.

Consultar todos los elementos: ${#nombre_array[*]} o ${#nombre_array[@]}.

Es preciso tener en cuenta lo siguiente:

1

Si al referenciar a un array no se utiliza subíndice se considera que se está referenciando a su primer elemento.

2

Para conocer el tamaño en bytes del array se utiliza #${nombre_array[x]}, donde x puede ser un subíndice, o bien los caracteres * o @.

A continuación, el siguiente script define dos arrays, que son animales mamíferos y animales ovíparos:

```
#!/bin/bash

mamifero[0]='perro'

mamifero[1]='gato'

mamifero[2]='león'

mamifero[3]='mono'

oviparo=("loro" "gallina" "pato" "ganso" "abeja")

echo ${mamifero[0]}

echo ${mamifero[1]}

echo ${mamifero[2]}

#imprimir elementos de un array

echo ${oviparo[*]}

#imprimir total de elementos del array

echo ${#oviparo[*]}
```


Órdenes internas de la Shell

Una orden interna de la Shell es una orden que el intérprete implementa y que ejecuta sin llamar a programas externos.

echo: envía una cadena a la salida estándar. Por ejemplo:

```
#!/bin/bash

echo Esto es una cadena
```

read: lee una cadena de la entrada estándar. Por ejemplo:

```
#!/bin/bash

echo -n "Introduzca un valor para var1: "

read var1

echo "var1 = $var1"
```

La orden read puede leer diferentes variables a la vez. También se puede combinar el uso de read con echo para mostrar un prompt que indique qué es lo que se está pidiendo. Hay una serie de caracteres especiales para usar en echo y que permiten posicionar el cursor en un sitio determinado:

_ nb

Retrocede una posición (sin borrar).

_ nf

Alimentación de página.

_ nn

Salto de línea.

_ nt

Tabulador.

Para que echo reconozca estos caracteres es necesario utilizar la opción .-e. y utilizar comillas dobles:

```
$ echo -e "Hola \t ¿cómo estás?"

hola como estás
```

cd

Cambia de directorio.

pwd

Devuelve el nombre del directorio actual.

let arg [arg]

Cada arg es una expresión aritmética a ser evaluada: $ let a=$b+7. Si el último arg se evalúa a 0, let devuelve 1; si no, devuelve 0.

exit
Finaliza la ejecución del guión, recibe como argumento un entero que será el valor de retorno.
true y false
Devuelven 0 y 1 siempre, respectivamente. El valor 0 se corresponde con true y cualquier valor distinto de 0 con false.

Estructuras de control

IF-CASE

Indica que determinadas órdenes solo se ejecuten cuando se cumplan unas condiciones concretas. Para ello se utilizan las estructuras if o case.

Estructura if
if [expresión] then órdenes a ejecutar si se cumple la condición elif [expresión] then órdenes a ejecutar si se cumple la condición else órdenes a ejecutar en caso contrario fi

El siguiente script, pregunta cuál es el nombre del usuario actual y lo valida

```
#!/bin/bash
echo -n "Escribe tu nombre de usuario: "
read usuario
if [ "$usuario" = "$USER" ]; then
    echo "Buen día, $usuario."
else
    echo "¡No eres el usuario $usuario!"
fi
```

Ejemplo básico de la sentencia if, script que verifica si el directorio actual es /root

```
#!/bin/bash
directorio=$(pwd)    #Asiga la salida del comando pwd a la variable directorio
if [ "$directorio" = "/root" ]; then
    echo "Directorio actual: $directorio"
else
    echo "Directorio actual diferente de: /root"
fi
```

Estructura case

```
case $var in
v1) …  #Acción a realizar si var toma el valor v1
;;
v2|v3) …#Acción a realizar si var toma el valor v2 o v3
```

;;

*) …# Caso por defecto

;;

esac

Por ejemplo, el siguiente script solicita un número e imprime el valor con caracteres:

```
#!/bin/bash
echo -n "Escribe un número entre 1 y 5: "
read x
case $x in
    1) echo "Tecleaste el número uno.";;
    2) echo "Tecleaste el número dos.";;
    3) echo "Tecleaste el número tres.";;
    4) echo "Tecleaste el número cuatro.";;
    5) echo "Tecleaste el número cinco.";;
    *) echo "Error, debias escribir un número entre 1 y 5";;
esac
```

Bucles

Permite ejecutar bloques de órdenes de forma iterativa dependiendo de una condición. Para ello se utilizan las estructuras if o case.

Estructura while

while [expresión] # Mientras la expresión sea cierta ...

do

...

done

Por ejemplo, un script que lee del teclado un número y lo suma con sus antecesores

```
#!/bin/bash

echo -n "Escribe un número: "

read numero

let suma=0

let contador=1

while [ $contador -le $numero ]; do# le -> less than

    let "suma = $suma + $contador"

    let "contador = $contador + 1"

done

echo "La suma del 1 al $numero es: $suma"
```

Otro ejemplo sería el de un script que calcula la factorial de un número usando el bucle while, recibe el número como parámetro

```
#!/bin/bash

contador=$1

numero=$contador

factorial=1

while [ $contador -gt 0 ] # gt -> greater than

do

  factorial=$(( $factorial * $contador ))

  contador=$(( $contador - 1 ))

done

echo "El factorial de $numero es $factorial"
```

Estructura until

El bucle until es similar al bucle while, la diferencia es que el código se ejecuta mientras la expresión se

evalúe como falsa.

until [expresión] # Mientras la expresión sea falsa ...

do

...

done

Por ejemplo, un script que imprime 5 veces el mensaje “Hola mundo”:

```
#!/bin/bash
contador=1
until [ $contador -gt 5 ] # gt -> greater than
do
    echo "Hola mundo $contador vez."
    contador=$(( contador+1 ))
done
```

Estructura for

for var in lista #Por cada valor en la estructura lista se ejecuta una iteración.

do

órdenes a ejecutar

done

Por ejemplo:

```
for i in 10 30 70
do
echo Mi número favorito es $i
done
```

O un script que imprime cadenas utilizando un bucle for:

```
#!/bin/bash

for x in Cabeza cara orejas ojos nariz boca; do

 echo "El valor de la variable x es: $x"

 sleep 1

done
```

Break y Continue

Sirven para interrumpir la ejecución secuencial del cuerpo del bucle:

- break transfiere el control a la orden que sigue a done, haciendo que el bucle termine antes de tiempo.

Loop while que imprime del número 0 al 9 utilizando break se termina el ciclo si el contador es igual a 5:

```
#!/bin/bash

contador=0

while [ $contador -lt 10 ]

do

  echo $contador

  if [ $contador -eq 5 ]

  then

    break

  fi

  contador=`expr $contador + 1`

done
```

- continue transfiere el control a done, haciendo que se evalúe de nuevo la condición, prosiguiendo el bucle.

Por ejemplo, un script que lee un archivo y solo imprime las líneas con menos de 20 caracteres:

```
#!/bin/bash

while read buf

do

   cuenta=`echo $buf | wc -c`

   if [ $cuenta -gt 20 ]

   then

      continue

   fi

   echo $buf

done < $1
```

En ambos casos, las órdenes del cuerpo del bucle siguientes a estas sentencias no se ejecutan. Lo normal es que formen parte de una sentencia condicional, como if.

La orden test

Test es una orden que permite evaluar si una expresión es verdadera o falsa. Se usan en la estructura if/then/else/ para determinar qué parte del script se va a ejecutar. La sintaxis de test puede ser una de las dos que se muestran a continuación:

test expresión

[expresión]

Operadores para números

-eq	Igual a
-ne	Distinto de
-lt	Menor que
-le	Menor o igual que
-gt	Mayor que
-ge	Mayor o igual que

Tabla 2.3. Operadores para números.

```
if [ $e -eq 1 ]
then
echo Vale 1
else
echo No vale 1
fi
```

Operadores para cadenas

-z cadena	Verdad si la longitud de cadena es cero
-n cadena o cadena	Verdad si la longitud de cadena no es cero
cadena1 == cadena2	Verdad si las cadenas son iguales. Se puede emplear = en vez de ==
cadena1 != cadena2	Verdad si las cadenas no son iguales
cadena1 < cadena2	Verdad si cadena1 se ordena lexicográficamente antes de cadena2 en la localización en curso
cadena1 > cadena2	Verdad si cadena1 se ordena lexicográficamente después de cadena2 en la localización en curso

Tabla 2.4. Operadores para cadenas.

```
#! /bin/bash
S1=’cadena’
S2=’Cadena’
if [ $S1!=$S2 ];
then
echo "S1(’$S1’) no es igual a S2(’$S2’)"
fi
if [ $S1=$S1 ];
then
echo "S1(’$S1’) es igual a S1(’$S1’)"
fi
```

Operadores sobre ficheros

-e fichero	El fichero existe
-r fichero	El fichero existe y tengo permiso de lectura
-w fichero	El fichero existe y tengo permiso de escritura
-x fichero	El fichero existe y tengo permiso de ejecución
-f fichero	El fichero existe y es regular
-s fichero	El fichero existe y es de tamaño mayor a cero
-d fichero	El fichero existe y es un directorio

Tabla 2.5. Operadores sobre ficheros.

```
if test -f "$1" # ¿ es un fichero ?
then
more $1
elif test -d "$1" # ¿ es un directorio ?
then
(cd $1;ls -l|more)
else # no es ni fichero ni directorio
echo "$1 no es fichero ni directorio"
fi
```

Operadores lógicos

-o	OR
-a	AND
!	NOT
\(	Paréntesis izquierdo
\)	Paréntesis derecho

Tabla 2.6. Operadores lógico

Funciones

Declaración

Para declarar una función:

function nombre:

{ mi_código }

Llamar a la función es como llamar a otro programa, solo hay que escribir su nombre.

```
# Se define la función hola
function hola {
echo ¡Hola!
}

hola # Se llama a la función hola
```

Definición con parámetros

Se pueden definir funciones con parámetros:

```
#!/bin/bash
function e {
echo $1
}
```

Se podría invocar como:

```
e Hola
```

Ejemplos

Algunos ejemplos más de scripts:

Ejemplo 1. Guión que solicita confirmación si va a sobrescribir un archivo cuando se use la orden cp.

```
#!/bin/bash

if [ -f $2 ]

then

echo "$2 existe. ¿Quieres sobreescribirlo? (s/n)"

read sn

if [ $sn = "N" -o $sn = "n" ]

then

exit 0

fi

fi

cp $1 $2
```

Ejemplo 2. Guión que liste los directorios existentes en el directorio actual.

```
#!/bin/bash

for archivo in *

do

test -d $archivo && ls $archivo

done
```

Ejemplo 3. Guión que borra con confirmación todos los archivos indicados como argumentos en la línea de comandos.

```
#!/bin/bash

while test "$1" != ""

do

rm -i $1

shift

done
```

Ejemplo 4. Guión que evalúa la extensión de un archivo. Si esta se corresponde con .txt., copia el archivo al directorio ~/copias. Si es otra la extensión presenta un mensaje.

```
case $1 in

*.txt)

cp $1 ~/copias/$1

;;

*)

echo "$1 extensión desconocida"

;;

Esac
```

Ejemplo 5. Guión que pone el atributo de ejecutable a los archivos pasados como argumento.

```
for fich in $@

do

if test -f $fich

then

chmod u+x $fich

fi

done
```

Ejemplo 6. Guión que lee dos números del teclado e imprime su suma.

```
#!/bin/bash

echo "Introduzca un número \n"

read numero1

echo "Introduzca otro número \n"

read numero2

let respuesta=$numero1+$numero2

echo "$numero1 + $numero2 = $respuesta \n"
```

Ejemplo 7. Guión que liste los directorios existentes en el directorio /root.

```
#!/bin/bash
# Lista todos los ficheros del directorio /root
for fichero in /root
do
    ls -l "$fichero"
done
```

Ejemplo 8. Guión que liste los enlaces simbólicos del directorio que se indique.

```
#!/bin/bash
echo "Escribe el nombre del directorio: "
read directorio
echo "Enlaces simbolicos en el directorio $directorio "
for fichero in $( find $directorio -type l )
do
    echo "$fichero"
done
```

Ejemplo 9. Usando funciones, guión verifica si existe un fichero en el directorio /root.

```
#!/bin/bash
function check() {
    if [ -e "/root/$1" ]
    then
        return 0
    else
        return 1
    fi
}
echo "Introduzca el nombre del archivo: "
read fichero
if check $fichero
    then
        echo "El fichero $fichero existe."
    else
        echo "El fichero $fichero no existe."
fi
```

Ejemplo 10. Guión que recibe como parámetro un usuario y verifica si existe en el sistema operativo.

```
#!/bin/bash

# Parámetros

usuario=$1

es_usuario=`grep $usuario /etc/passwd`

if test -z "$es_usuario"

then

    echo "$usuario no es usuario del sistema operativo"

else

    echo "$usuario si es usuario del sistema operativo"

fi
```


VII. Resumen

En esta unidad, se ha introducido el concepto de máquina virtual como un mecanismo que facilita la ejecución de una simulación de un sistema operativo diferente al que se tiene instalado por defecto. Para poder crear y ejecutar una máquina virtual, se requiere de un software específico. Las dos herramientas más importantes para este fin son Virtual Box y VMware.

En esta unidad se ha descrito cómo usar la herramienta Virtual Box para crear, ejecutar y configurar máquinas virtuales. Para ello, se ha creado una máquina virtual a partir de una distribución de Linux denominada Lubuntu.

Asimismo, se ha descrito cómo se puede importar una máquina virtual creada por terceros. Para este caso, se ha utilizado una máquina virtual de una distribución de Linux de tipo Lubuntu.

La última parte de la unidad se ha dedicado a describir la Shell de comandos de Linux, habiéndose revisado aquellos que permiten una administración básica del sistema.

VIII. Caso práctico

Se pide

Crear un script que muestre el siguiente menú:

Ver directorio actual.....[1]

Copiar archivos.....[2]

Editar archivos.....[3]

Imprimir archivo.....[4]

Salir del menú.....[5]"

Donde

- Si elige la primera opción, se mostrarán los archivos del directorio actual.
- Si elige la segunda opción, se le pedirá el nombre del archivo que se quiere copiar y el nombre del archivo donde se quiere copiar.
- Si elige la tercera opción, se le pedirá el nombre del archivo que se desea editar y se abrirá el editor vi para editarlo.
- Si elige la cuarta opción, se le pedirá el nombre del archivo que se desea imprimir y se imprimirá el mismo.
- Si elige la quinta opción, se saldrá del script.

Solución

```
while true

do

clear

echo "

Ver directorio actual.....[1]

Copiar ficheros.....[2]

Editar ficheros.....[3]

Imprimir fichero.....[4]

Salir del menú.....[5]"
```

```
read i

case $i in

1) ls -l|more; read z

;;

2) echo "Introduzca [desde] [hasta]"

read x y

cp $x $y

read x

;;

3) echo "¿Nombre de fichero a editar?"

read x;

vi $x

;;

4) echo "¿Nombre de fichero a imprimir?"

read x

lpr $x

;;

5) clear; break

;;

esac

done
```

Recursos

Bibliografía

- **Advanced Linux Programming**: Mitchell, Mark et al. Advanced Linux Programming. Indianapolis, IN: New Riders Publishing
- **Bash Guide for Beginners**:

  Garrels, Machtelt. Bash Guide for Beginners. 2008. [En línea] URL disponible enhttp://www.tldp.org/LDP/Bash-Beginners-Guide/Bash-Beginners-Guide.pdf
- **Beginning Linux Programming**: Matthew, Neil y Stones, Richard. Beginning Linux Programming. Ed. Wiley
- **Guía oficial de primeros pasos con VirtualBox**:

  Guía oficial de primeros pasos con VirtualBox. [En línea] URL disponible enhttps://www.virtualbox.org/manual/ch01.html
- **Linux Kernel Development**: Love. Robert. Linux Kernel Development. Addison Wesley
- **User Manual** :

  User Manual. [En línea] URL disponible en https://www.virtualbox.org/manual/UserManual.html

Glosario.

- **Exportar una máquina virtual**: es la acción de exportar una máquina virtual creada para distribuirla entre otras personas.
- **Importar/Cargar máquina virtual**: es la acción de importar una máquina virtual que ha sido creada por un tercero a un sistema de gestión de máquinas virtuales para poder usarla.
- **Lubuntu**: es una distribución de Linux que requiere pocos recursos hardware.
- **Máquina virtual**: es una simulación de un sistema operativo. Físicamente es un archivo con varias extensiones posibles tales como .ova, .vmdk, etc.
- **Script o guión**: es un archivo de texto que contiene comandos de la Shell que se interpretarán cuando se ejecute el script.
- **Shell o intérprete**: aplicación que permite interactuar al usuario con el sistema operativo mediante la ejecución de comandos o sentencias.
- **Sistema operativo anfitrión**: sistema operativo sobre el que se ejecuta una máquina virtual.
- **Sistema operativo huésped**: sistema operativo virtualizado que se ejecuta sobre el sistema operativo anfitrión.
- **Sistema operativo virtualizado**: es un sinónimo de máquina virtual.
- **Software multiplataforma**: es un software que puede ejecutarse en más de un sistema operativo.

- **Virtual Box**: software gratuito para crear máquinas virtuales.
- **Virtualizar**: es la acción de crear una máquina virtual.
- **Vmware**: software con diferentes versiones de licencias para crear máquinas virtuales.