INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

SISTEMAS OPERATIVOS

PROFESOR: CORTES GALICIA JORGE

CICLO ESCOLAR: 2024-2

***PRACTICA 4 – PROGRAMACIÓN BAJO EL INTÉRPRETE DE COMANDOS (SHELL) EN LINUX Y WINDOWS***

**16 DE ABRIL DE 2024**

INTRODUCCION

La programación en el intérprete de comandos, conocido como shell, representa una herramienta fundamental para interactuar con un sistema operativo a través de una interfaz de línea de comandos. A diferencia de las interfaces gráficas de usuario, que requieren la interacción con ventanas y menús, el shell permite a los usuarios ejecutar comandos directamente desde una terminal, lo que proporciona una forma rápida y eficiente de realizar diversas tareas del sistema.

Los shells están disponibles en la mayoría de los sistemas operativos, incluidos Linux y Windows, y proporcionan a los usuarios un control directo sobre el sistema. En el entorno Linux, el shell más popular y ampliamente utilizado es Bash (Bourne-Again SHell). Bash es una evolución del shell original de Unix, conocido como Bourne shell, y ofrece un intérprete de comandos completo con una amplia gama de herramientas para programar y automatizar tareas del sistema. Estas herramientas incluyen la ejecución de comandos, la redirección de entrada y salida, la manipulación de variables de entorno, la creación de scripts y la integración con otros programas.

En el caso de Windows, si bien el shell predeterminado es el Command Prompt, que utiliza un lenguaje de comando similar al de Bash pero con diferentes sintaxis y herramientas, Microsoft también ofrece PowerShell, un shell más potente y flexible. PowerShell permite la automatización de tareas del sistema mediante scripts, brindando a los usuarios la capacidad de administrar y controlar sus sistemas de manera eficiente y efectiva.

La programación en el intérprete de comandos resulta especialmente útil para automatizar tareas repetitivas o para llevar a cabo operaciones avanzadas del sistema que no son fáciles de realizar a través de una interfaz gráfica de usuario. Además, los scripts de shell son portátiles entre diferentes sistemas operativos, lo que facilita la creación de scripts una vez y su ejecución en múltiples plataformas, lo cual es una ventaja significativa para los administradores de sistemas y desarrolladores.

SECCIÓN LINUX

*1.- A través de la ayuda en línea que proporciona Linux, investigue para qué es utilizado el comando bash, mencione sus características principales, dé un resumen de su funcionalidad*

Bash, como intérprete de comandos de Unix, destaca como uno de los shells más utilizados en los entornos Unix y Linux. Sus características principales son:

* Amplio soporte para una extensa gama de comandos y utilidades presentes en Unix y Linux.
* Personalización flexible del prompt y la configuración del shell, adaptándose así a las necesidades individuales del usuario.
* Facilita la ejecución de comandos en secuencia o en conjunto mediante tuberías (pipes) y redirecciones.
* Capacidades de programación de scripts, permitiendo el uso de variables, bucles, condicionales y otros elementos esenciales.

El funcionamiento de Bash es directo: tras abrir una terminal o consola de comandos, el usuario introduce sus comandos en el prompt de Bash, que luego son ejecutados por el intérprete de comandos en el sistema operativo. Este poderoso shell habilita al usuario para ejecutar programas, editar archivos, administrar procesos y llevar a cabo tareas de administración de sistemas con eficacia. Además, la capacidad de Bash para ejecutar scripts automatizados resulta invaluable para administradores de sistemas y desarrolladores.

*2.- Revise en la ayuda en línea de Linux los siguientes comandos: date, tar, bzip2, bunzip2. Dé un resumen de sus características y funcionalidad.*

Date

Características:

* Permite visualizar la fecha y hora actuales en una variedad de formatos.
* Facilita la configuración de la fecha y hora del sistema a un momento específico.
* Ofrece soporte para la conversión de fechas y horas entre distintos formatos.

Funcionamiento:

El funcionamiento básico del comando *date* es sencillo, al ejecutarlo sin argumentos, muestra la fecha y hora actual en el formato por defecto del sistema. No obstante, si se requiere visualizar la fecha y hora en un formato particular, es posible utilizar argumentos adicionales para lograrlo.

Tar

Características:

* Permite la creación de archivos de archivo, ya sea comprimidos o sin comprimir.
* Facilita la adición de archivos a un archivo de archivo existente o la extracción de archivos de un archivo de archivo preexistente.
* Permite la compresión y descompresión de archivos utilizando una variedad de algoritmos de compresión.
* Puede ser utilizado para realizar copias de seguridad y restauraciones de datos de manera efectiva.

Funcionamiento:

El funcionamiento básico del comando *tar* para la creación de un archivo de archivo implica el uso del argumento *cvf*, seguido del nombre del archivo que se desea crear y los nombres de los archivos o directorios que se desean incluir en dicho archivo.

Bzip2

Características y funcionamiento:

* Compresión: Para comprimir un archivo, simplemente se utiliza el comando bzip2 seguido del nombre del archivo. Por ejemplo: bzip2 archivo. Esto creará un archivo comprimido con extensión ".bz2" en el mismo directorio que el archivo original.
* Descompresión: Para descomprimir un archivo comprimido con bzip2, se emplea el comando bzip2 -d archivo.bz2. Esto restaurará el archivo original en el mismo directorio que el archivo comprimido.
* Compresión múltiple: bzip2 permite la compresión de múltiples archivos simultáneamente. Para ello, se lista el nombre de los archivos que se desean comprimir después del comando bzip2. Por ejemplo: bzip2 archivo1 archivo2 archivo3. Esto generará archivos comprimidos separados para cada archivo de entrada.
* Opciones de nivel de compresión: Se pueden ajustar las opciones de compresión para controlar el nivel de compresión. El nivel de compresión predeterminado es 6, pero se puede aumentar o disminuir según las necesidades. Por ejemplo, bzip2 -9 archivo aumentará el nivel de compresión, resultando en un archivo más pequeño pero con un proceso de compresión más lento.
* Verificación de integridad: bzip2 también ofrece la capacidad de verificar la integridad de los archivos comprimidos. Utilizando el comando bzip2 -t archivo.bz2, se puede verificar si el archivo comprimido está dañado o no.

Bunzip2

Características:

* bunzip2 es una herramienta de línea de comandos que se utiliza para descomprimir archivos que han sido comprimidos utilizando bzip2.
* Este comando es similar al comando gunzip, que se utiliza para descomprimir archivos que han sido comprimidos utilizando el algoritmo de compresión gzip.
* bunzip2 es capaz de descomprimir archivos que han sido comprimidos con bzip2 y bzip, pero no es capaz de descomprimir archivos que han sido comprimidos con otros algoritmos de compresión, como gzip o zip.
* Este comando es capaz de manejar archivos grandes y comprimirlos de manera eficiente.
* bunzip2 no elimina el archivo original después de descomprimirlo, por lo que el archivo comprimido y el archivo descomprimido estarán presentes en el sistema después de la operación.

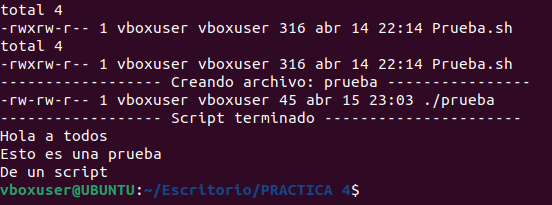
Funcionamiento:

El funcionamiento básico del comando *bunzip2* es simple: se proporciona el nombre del archivo comprimido como argumento al comando. Por ejemplo, para descomprimir un archivo llamado *archivo.tar.bz2*, simplemente se ejecuta *bunzip2 archivo.tar.bz2*. Esto restaurará el archivo original *archivo.tar* en el sistema.

*3.- Capture y ejecute el siguiente script. Observe su funcionamiento y explique.*



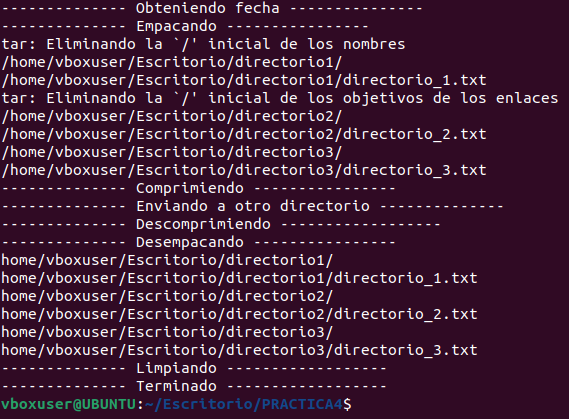
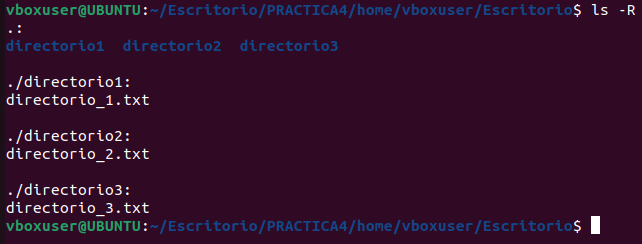
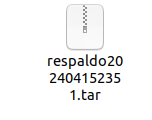
Para ejecutar el script cambie los permisos de su archivo para permitir ejecución a nivel dueño. Ejecute como un programa normal en C.



Este script de shell primero limpia la pantalla, muestra el contenido del directorio actual, luego borra un archivo llamado "prueba", crea uno nuevo con ese mismo nombre y escribe algunas líneas de texto en él. Después muestra información detallada sobre el nuevo archivo y su contenido. Finalmente, muestra un mensaje de que el script ha terminado y muestra el contenido del archivo "prueba" en la terminal.

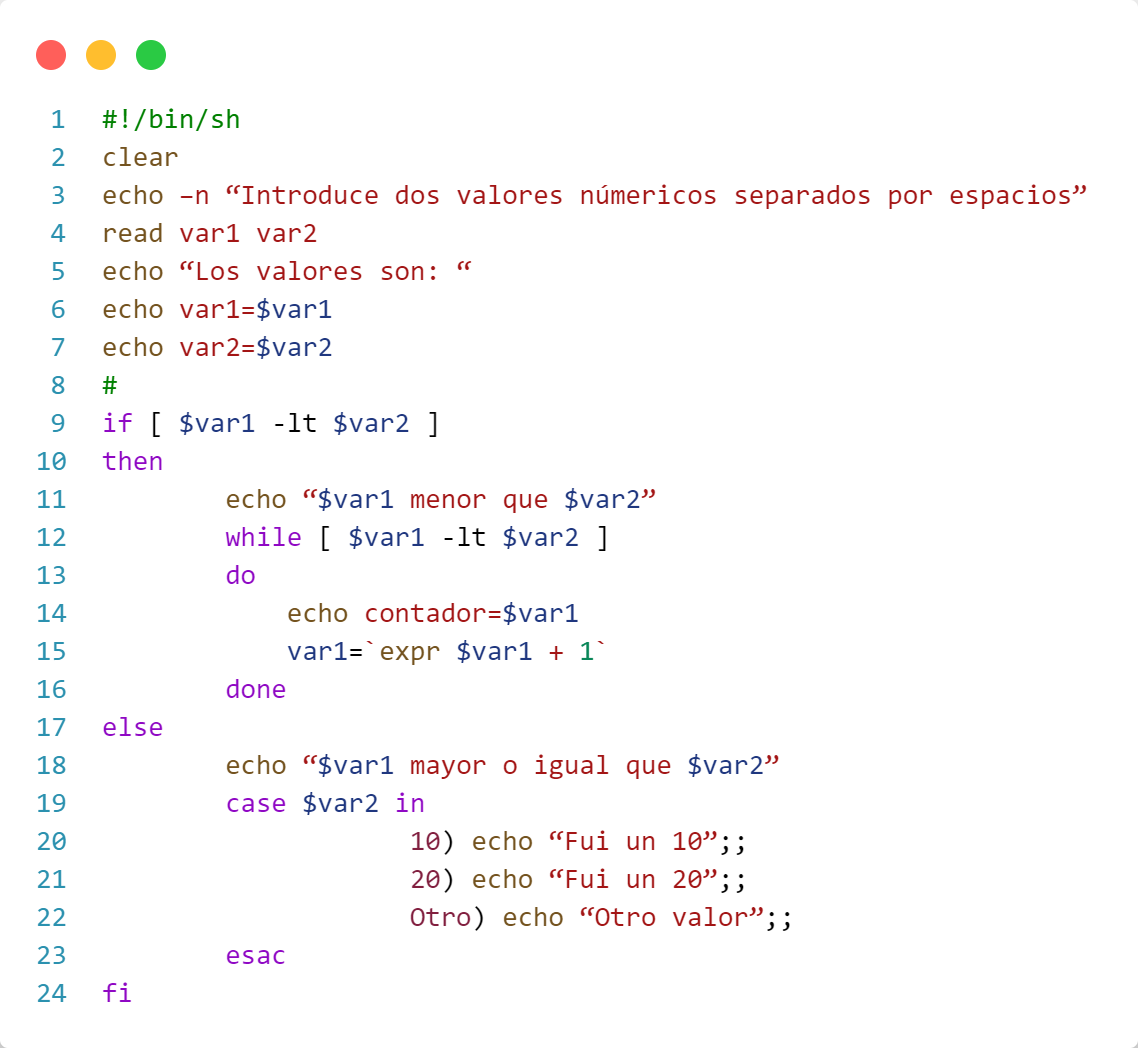
*4.- Capture y ejecute el siguiente script. Sustituya las rutas utilizadas en el script por las rutas que usted utilice. Observe su funcionamiento y explique.*

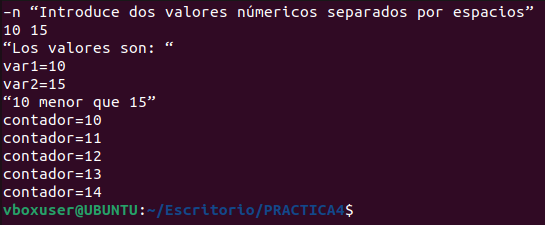
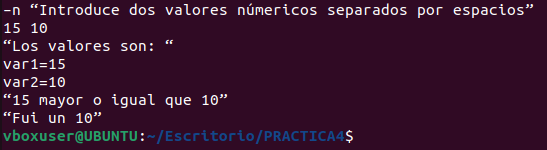




Este script de shell automatiza el proceso de respaldo de varios directorios. Primero, obtiene la fecha y hora actual y crea un archivo tar que contiene los directorios especificados. Luego, comprime este archivo usando bzip2 y lo copia a otro directorio. Después, descomprime el archivo en el directorio de respaldo y elimina los archivos temporales generados durante el proceso. Finalmente, muestra un mensaje indicando que el proceso ha terminado.

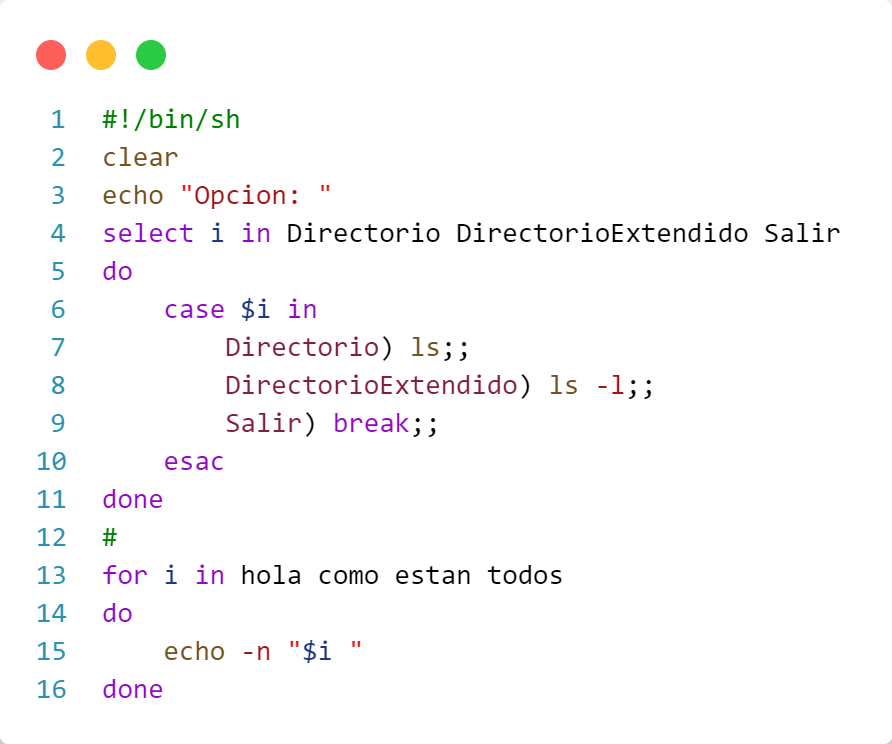
*5.- Capture y ejecute el siguiente script. Observe su funcionamiento.*

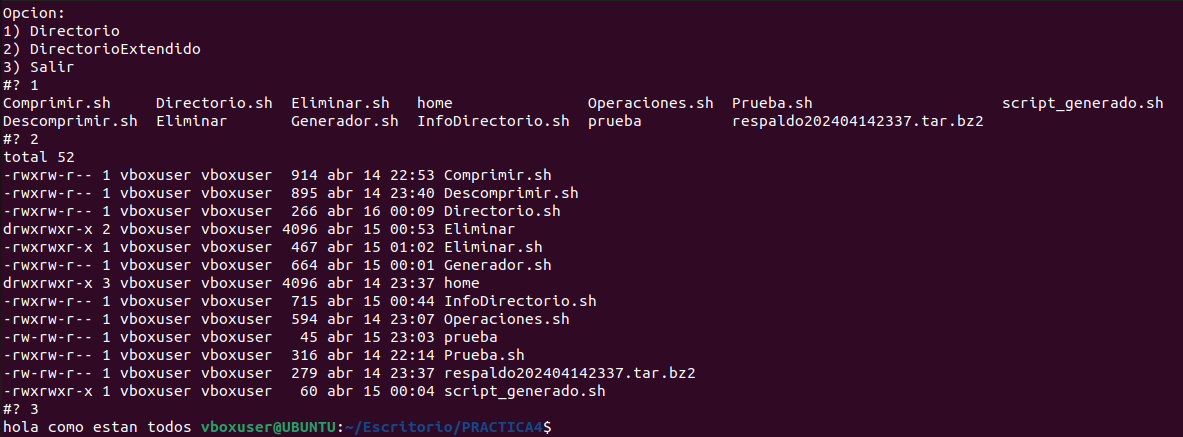


Este script de shell solicita al usuario dos valores numéricos, luego los muestra y realiza diferentes acciones según la relación entre estos valores. Si el primer valor es menor que el segundo, el script muestra cada número en un rango ascendente hasta el segundo valor. Si el primer valor es mayor o igual que el segundo, el script imprime un mensaje específico dependiendo del valor del segundo número: "Fui un 10" si es 10, "Fui un 20" si es 20, o "Otro valor" si es diferente.

6. Capture y ejecute el siguiente script. Observe su funcionamiento.



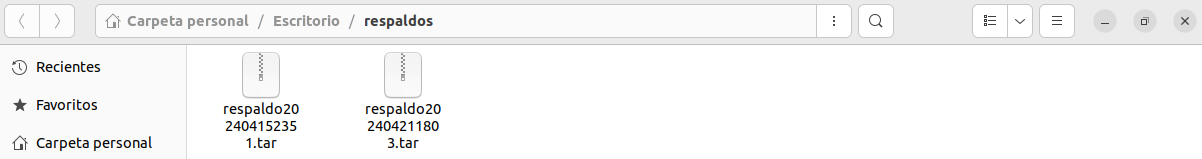


Este script de shell proporciona un menú interactivo con tres opciones: "Directorio", "DirectorioExtendido" y "Salir". El usuario puede seleccionar una opción escribiendo su número correspondiente. Dependiendo de la opción seleccionada, el script muestra el contenido del directorio actual con el comando ls si se elige "Directorio", o muestra el contenido en formato largo con ls -l si se elige "DirectorioExtendido". Si se elige "Salir", el script termina. Después del menú, el script imprime las palabras "hola como estan todos" en una línea.

7.- Programe un script para que descomprima en sus rutas originales los directorios que se han compreso con el anterior script



Archivos comprimidos:

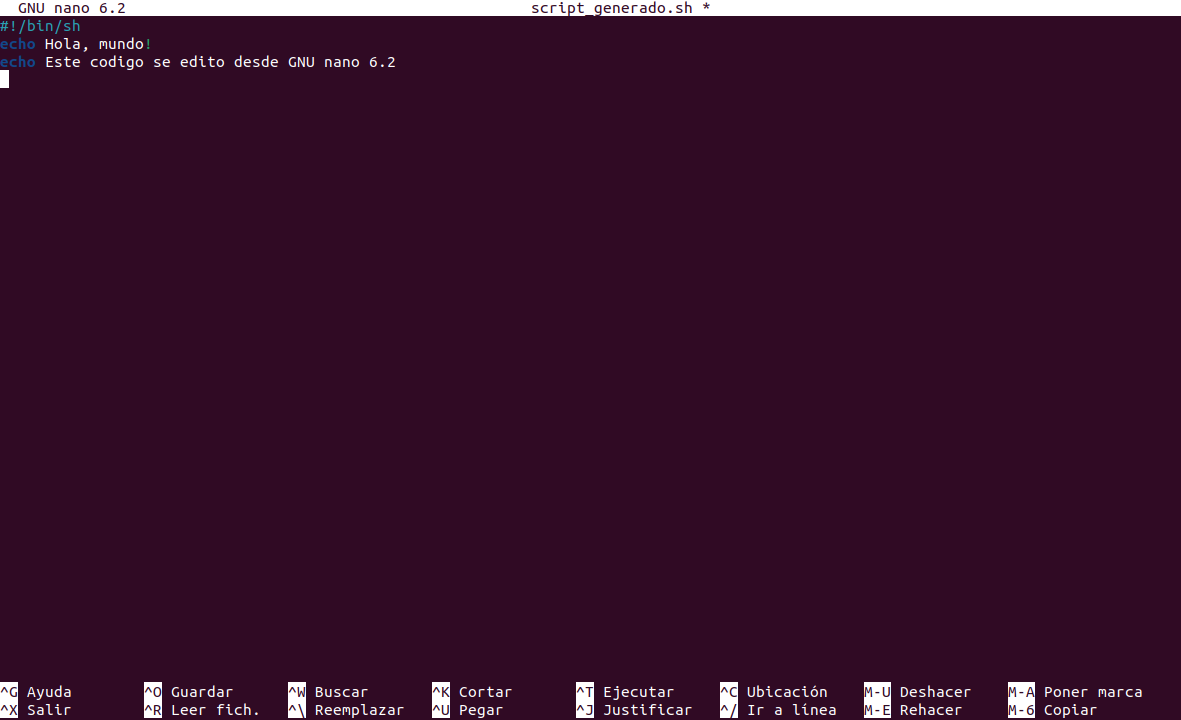
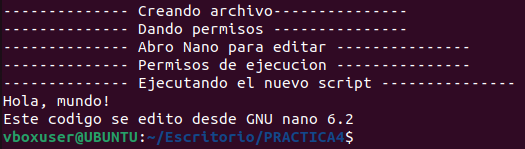


Descompresión del ultimo archivo comprimido:

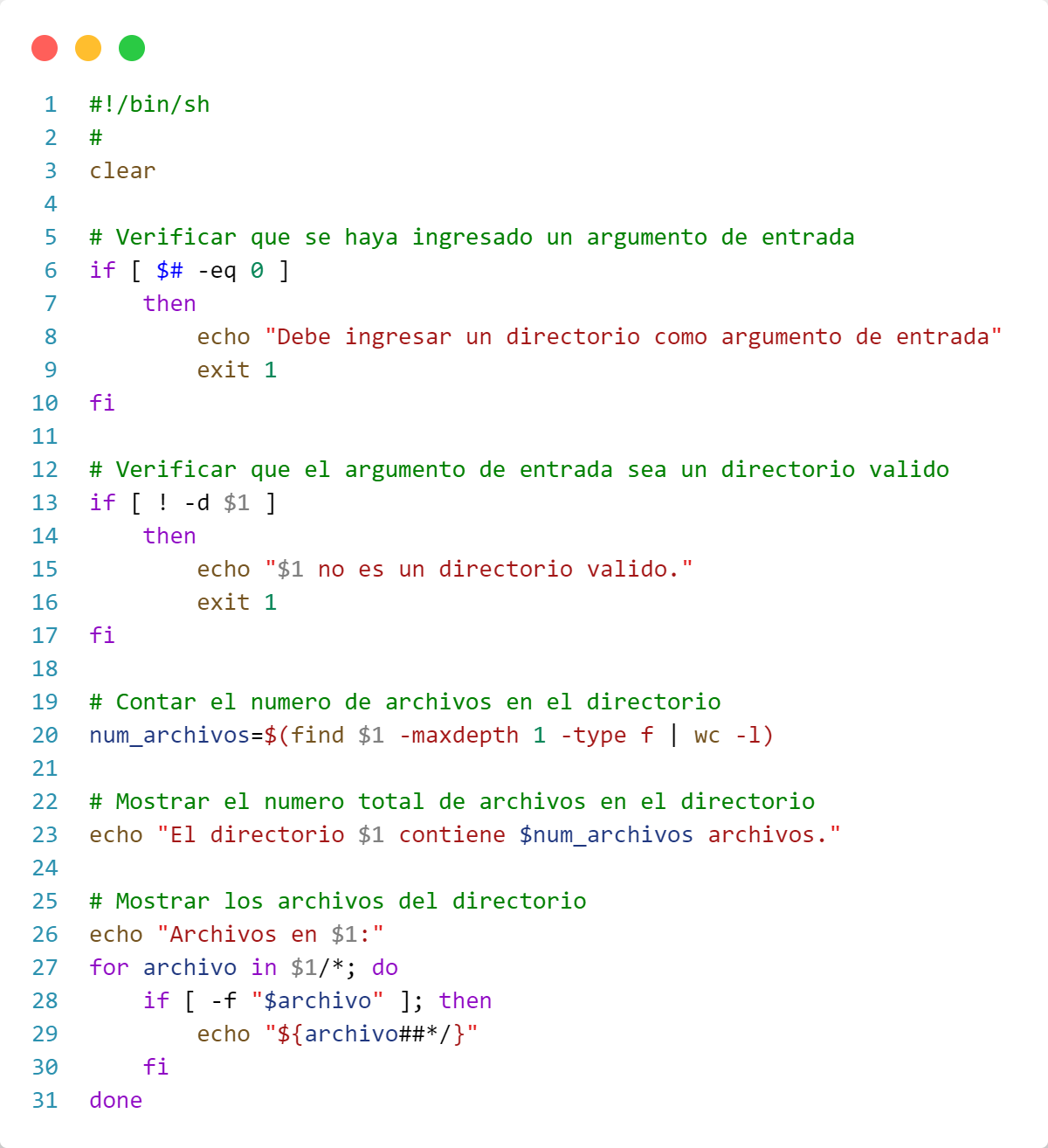


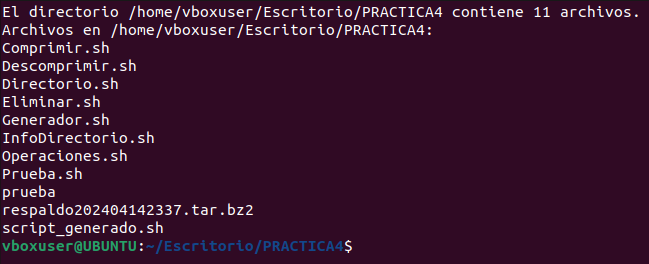
8.- Programe un script generador de scripts, es decir, que cree un archivo, que posteriormente permita editarlo y que finalmente cambie los permisos para ejecución del archivo, mandando a ejecutar el script desde el script original.



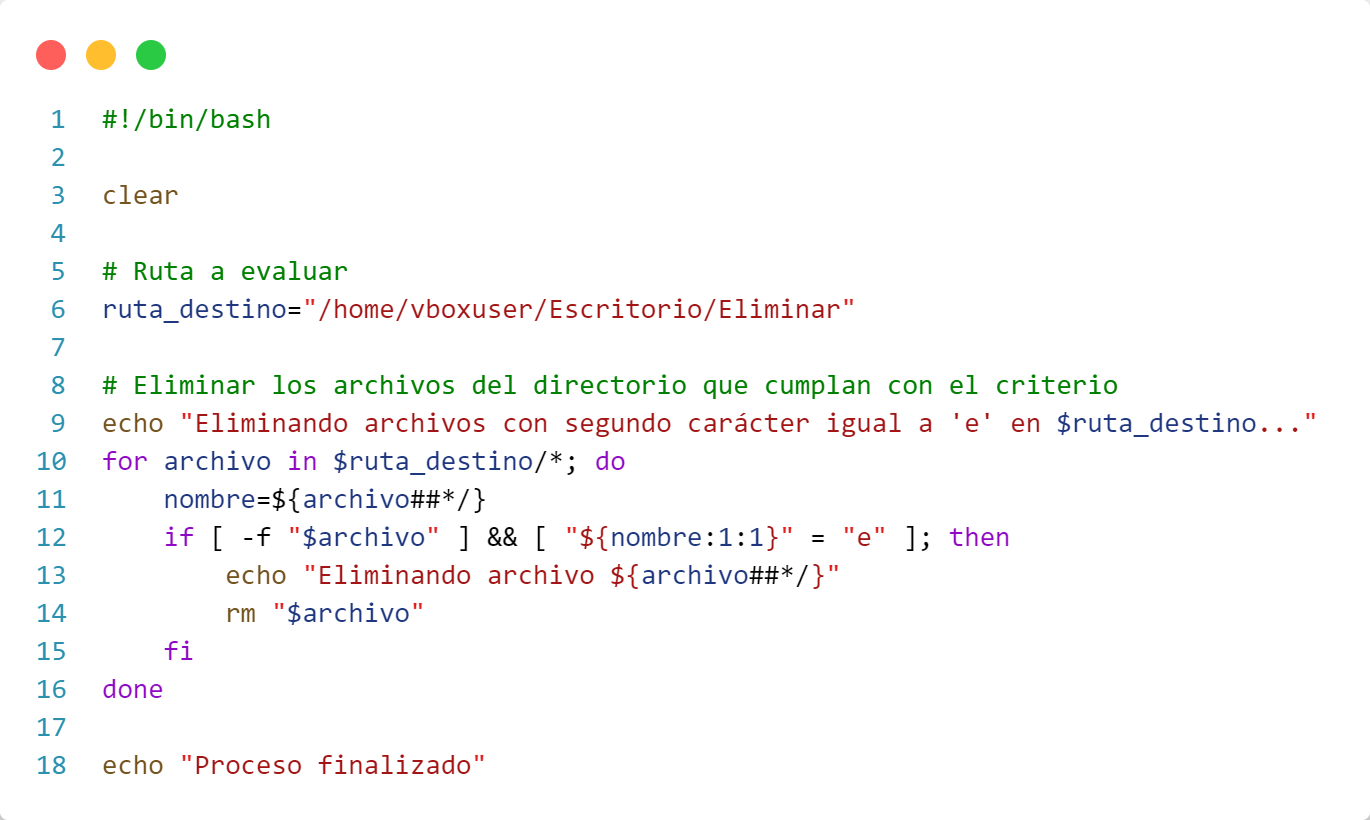


9.- Programe un script a través de las estructuras de control de flujo para que permita visualizar los archivos de un directorio que se envíe como argumento de entrada (no utilizar el comando ls). Además, el script deberá mostrar el número total de archivos en el directorio.

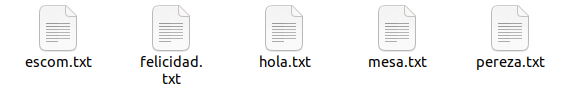




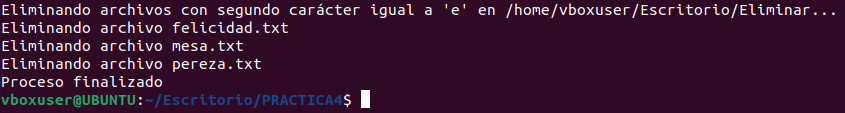
10.- Programe un script que elimine todos los archivos de un directorio especificado desde el propio script, y cuyo segundo carácter sea la letra “e”.



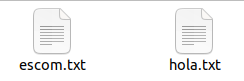
Archivos con distintos nombres:



El programa elimina los que tienen en la segunda letra una ‘e’:



El directorio solo queda con estos archivos:



SECCIÓN WINDOWS

*1.- Investigue los siguientes comandos: echo, del, date, tar, rar. Dé un resumen de sus características y funcionalidad.*

*echo:* Este comando se utiliza para mostrar mensajes en la consola o para activar y desactivar la característica de eco de comandos. Cuando se ejecuta sin parámetros, *echo* muestra la configuración actual de la característica de eco. El eco de comandos es la capacidad del sistema para mostrar en pantalla los comandos que se ingresan antes de su ejecución.

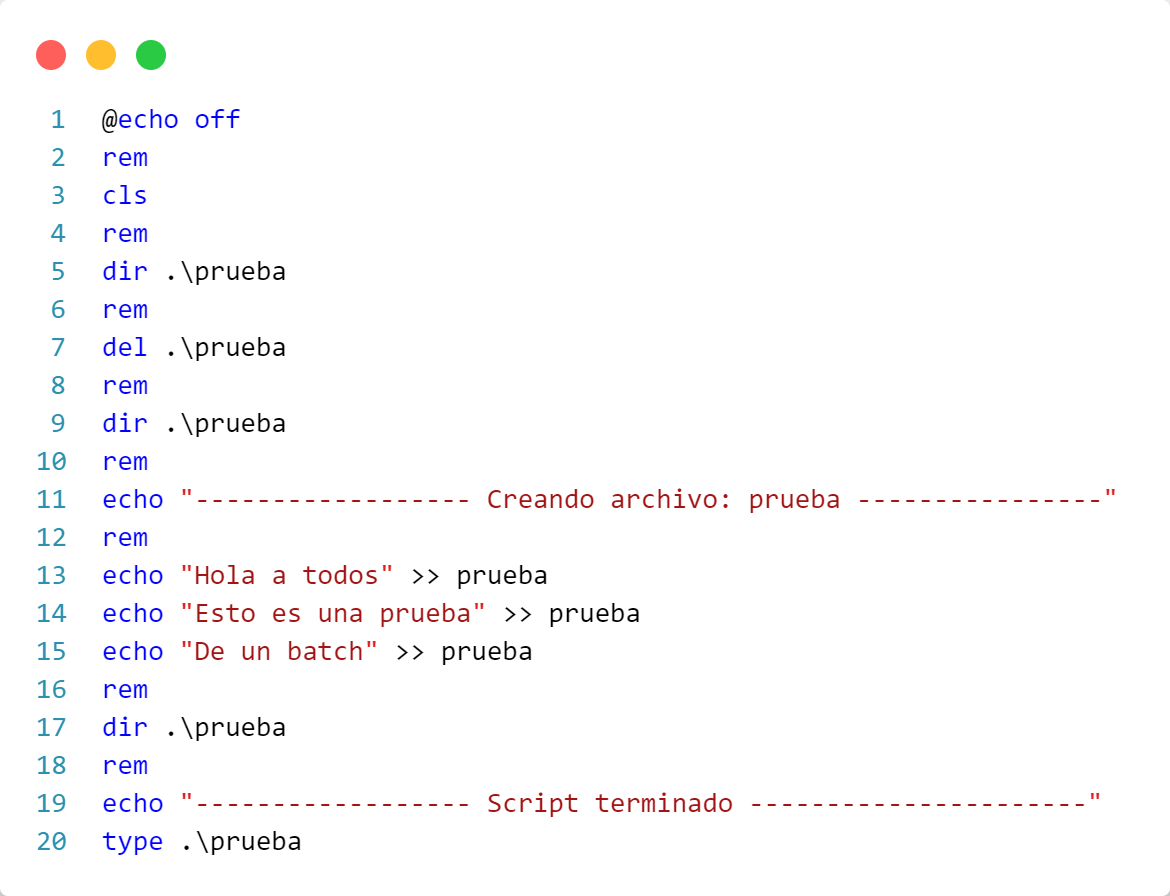
*del*: *del* es un comando que se emplea para eliminar uno o varios archivos del sistema. Esencialmente, realiza la misma función que el comando *erase*. Al ejecutar *del*, se elimina de manera permanente el archivo especificado, sin enviarlo a la papelera de reciclaje o a una carpeta de recuperación.

*date*: Este comando se utiliza para mostrar la fecha actual del sistema o para establecer una nueva fecha. Cuando se ejecuta sin parámetros, *date* simplemente muestra la fecha actual del sistema y solicita al usuario que ingrese una nueva fecha si desea modificarla. Es una herramienta fundamental para administrar y mantener la precisión temporal del sistema.

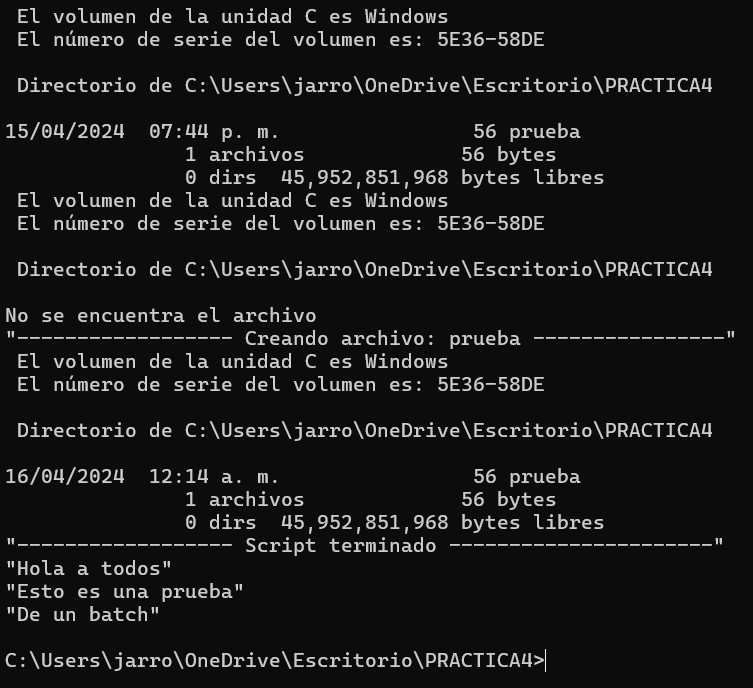
*tar*: *tar* es un comando utilizado para archivar y recuperar archivos en y desde un único archivo, conocido como "tarfile". Aunque su nombre se deriva de "tape archive" (archivo de cinta), el *tarfile* puede ser cualquier tipo de archivo en el sistema. Permite la compresión y la agrupación de múltiples archivos y directorios en un solo archivo, facilitando su transporte o respaldo.

*rar*: Este comando se encarga de comprimir o descomprimir archivos directamente desde la consola de comandos. La herramienta *rar* es especialmente útil para la compresión de archivos grandes o la creación de archivos comprimidos con múltiples archivos y directorios. Su funcionalidad versátil lo convierte en una herramienta valiosa para la gestión de archivos en entornos de línea de comandos.

*2. Capture y ejecute el siguiente batch. Guarde el archivo con extensión .bat. Observe su funcionamiento y explique.*



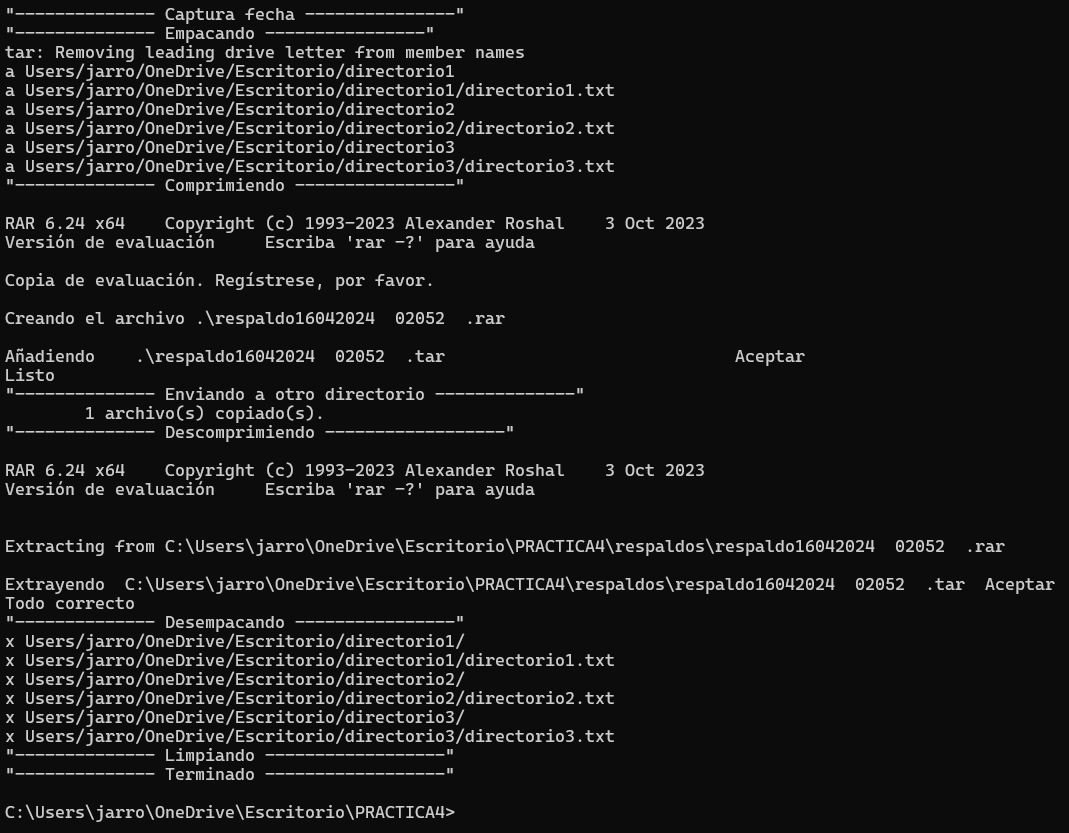
Para ejecutar escriba desde la línea de comando >nombre\_archivo.bat.



Este script de lotes para Windows inicia limpiando la pantalla y luego busca un archivo llamado "prueba" en el directorio actual. Si encuentra este archivo, lo elimina y muestra el contenido actualizado del directorio. Luego, imprime un mensaje indicando que está creando el archivo "prueba" y agrega tres líneas de texto al mismo. Después, muestra la información actualizada del archivo y finaliza el script mostrando su contenido completo en la terminal.

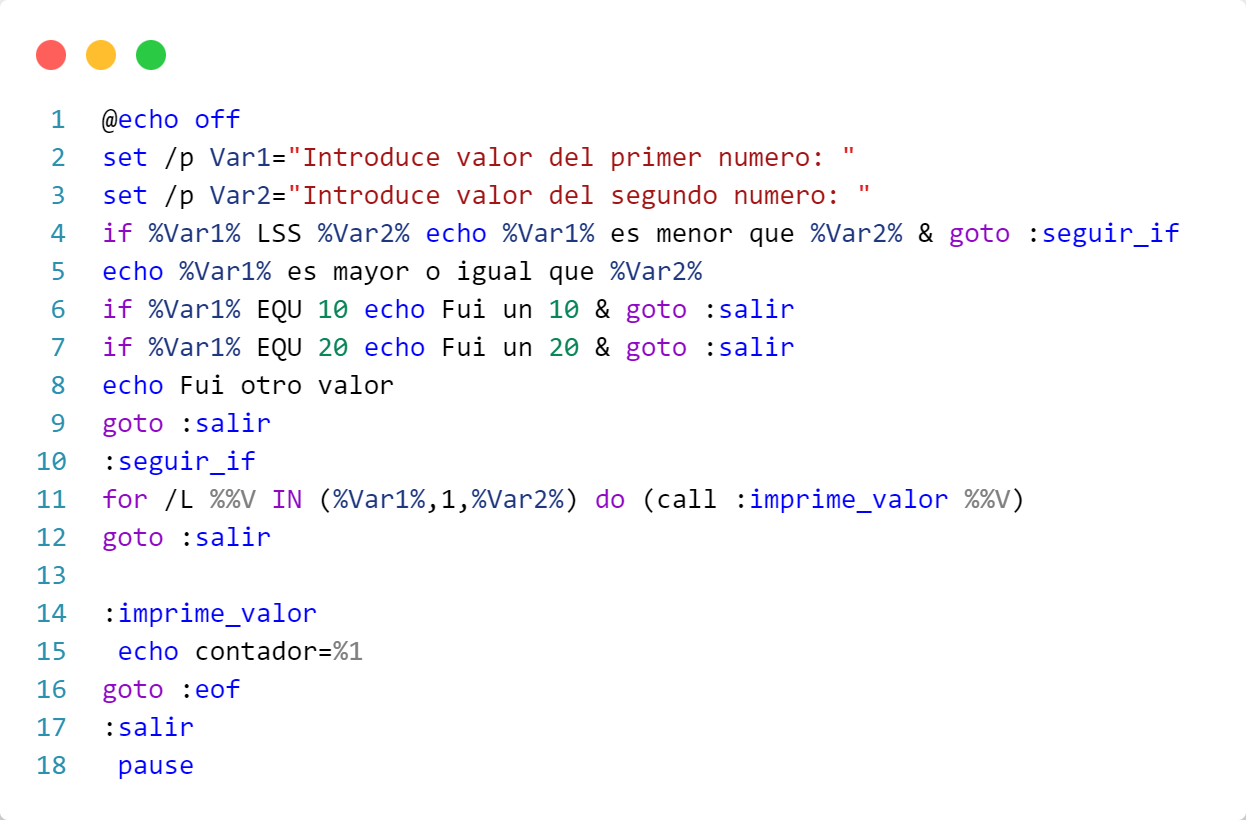
3. Capture y ejecute el siguiente batch. Sustituya las rutas utilizadas en el batch por las rutas que usted utilice. Deberá tener instalado el programa Winrar para utilizar el comando rar en el script. Observe su funcionamiento y explique.

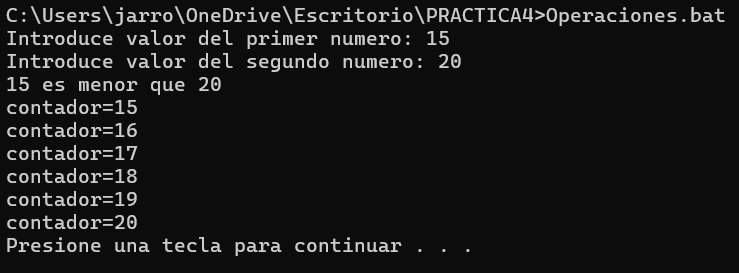
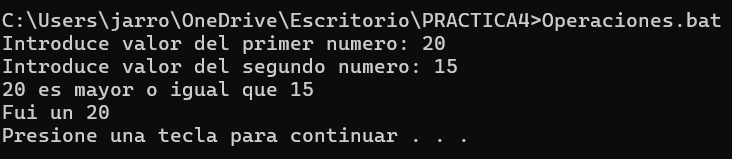


Este script de lote (batch) para Windows automatiza el proceso de respaldo de varios directorios. Primero, captura la fecha y la hora actual y las utiliza para generar un nombre único para el respaldo. Luego, utiliza el comando tar para empaquetar los directorios especificados en un archivo tar. A continuación, utiliza el programa WinRAR para comprimir el archivo tar en formato RAR. Posteriormente, copia el archivo comprimido a otro directorio. Después, descomprime el archivo RAR en el directorio de respaldo. Finalmente, desempaqueta el archivo tar dentro del directorio de respaldo, elimina los archivos temporales generados durante el proceso y muestra un mensaje indicando que el proceso ha terminado.

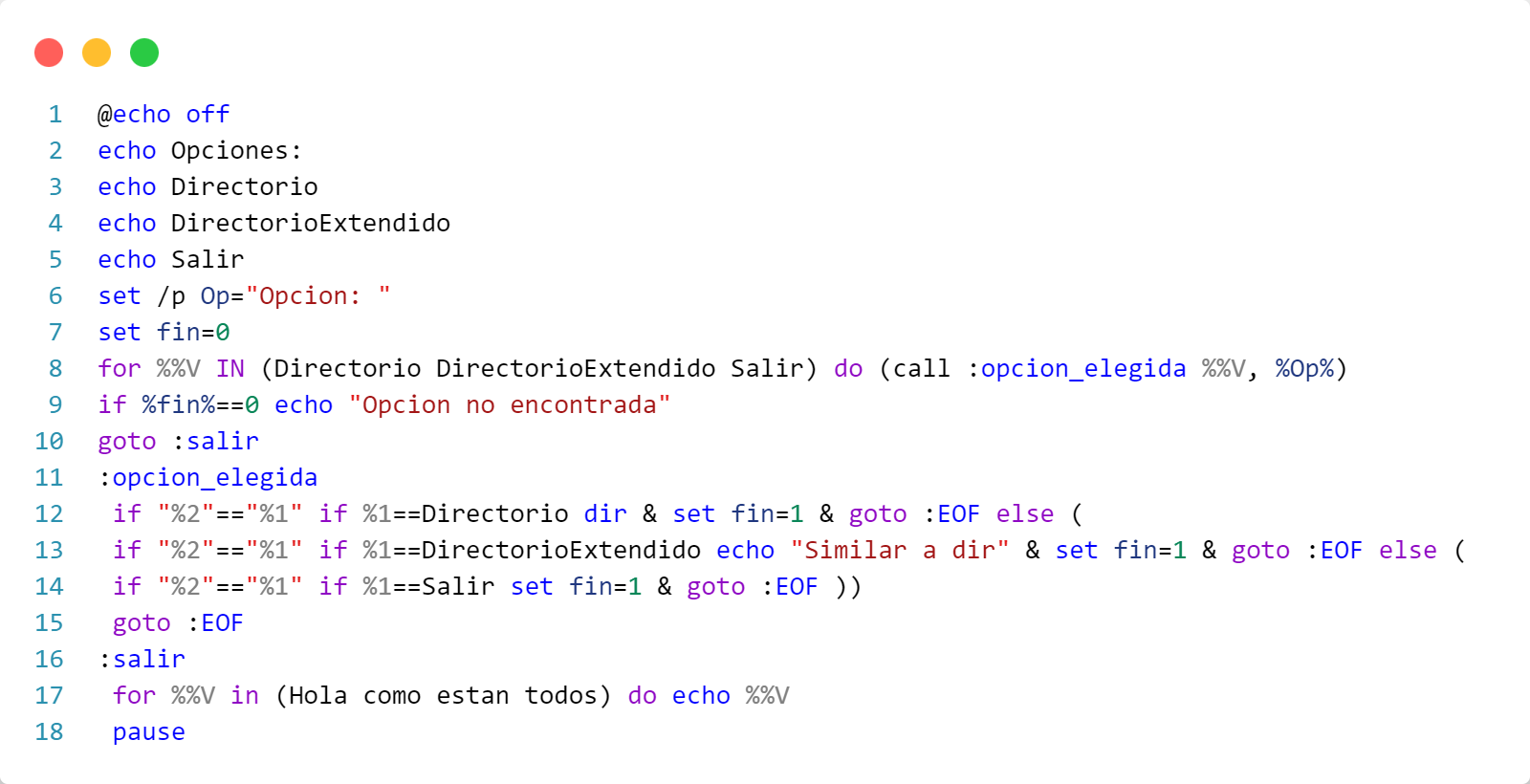
4.- Capture y ejecute el siguiente script. Observe su funcionamiento.



Este script de lotes para Windows solicita al usuario que ingrese dos números. Luego, compara los números ingresados y ejecuta diferentes acciones según las comparaciones. Si el primer número es menor que el segundo, muestra todos los números en el rango entre ellos. Si el primer número es mayor o igual que el segundo, comprueba si el primer número es igual a 10 o 20, imprimiendo un mensaje correspondiente en cada caso. Si el primer número no es ni 10 ni 20, imprime un mensaje indicando que fue otro valor. Después de ejecutar las acciones correspondientes, el script se detiene y espera a que el usuario presione una tecla para salir.

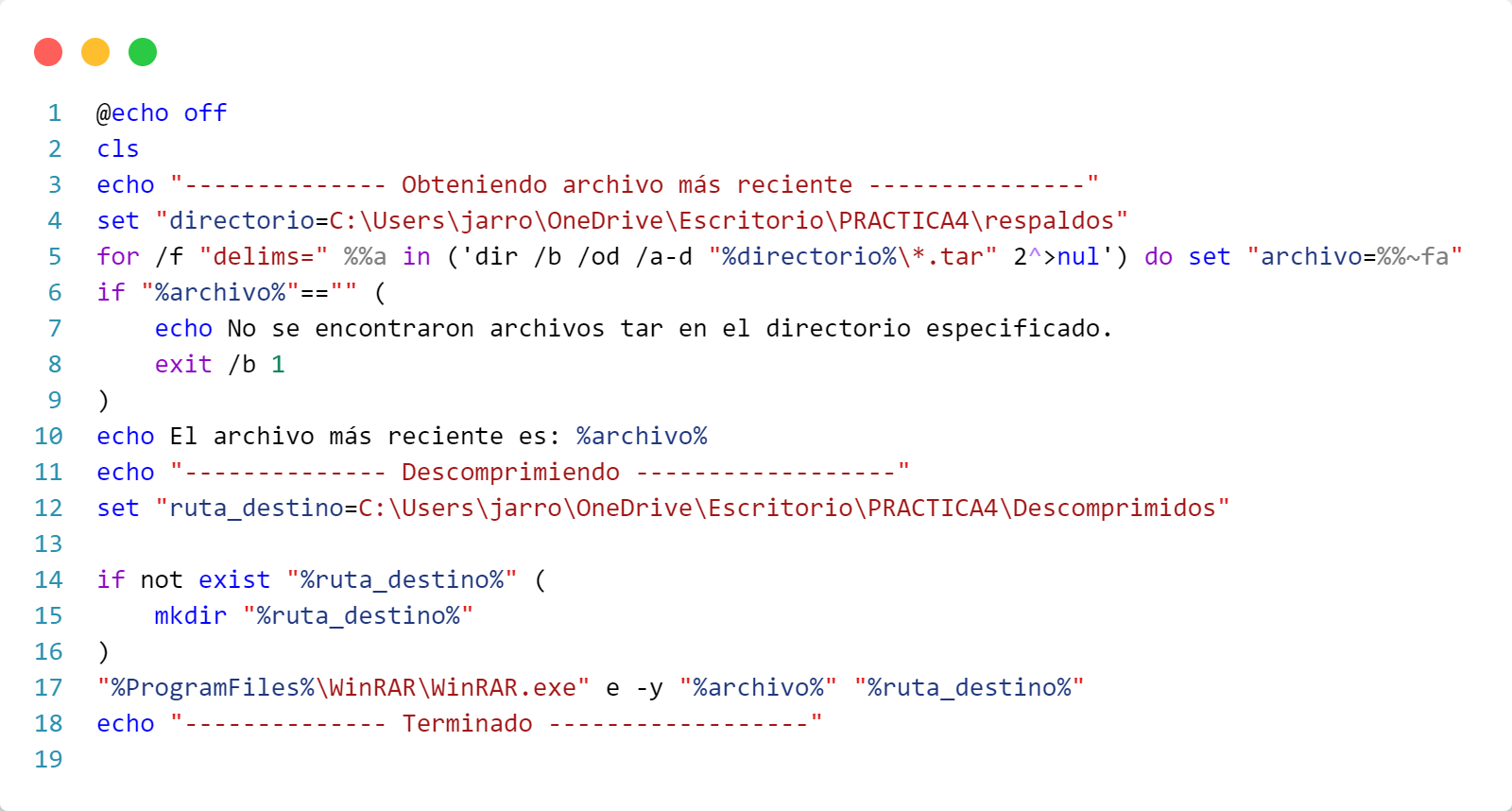
5.- Capture y ejecute el siguiente script. Observe su funcionamiento.

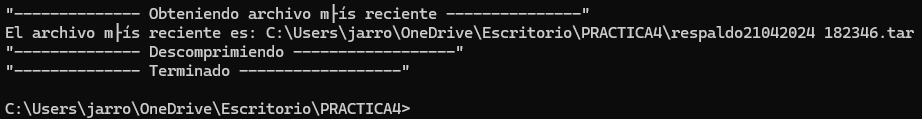


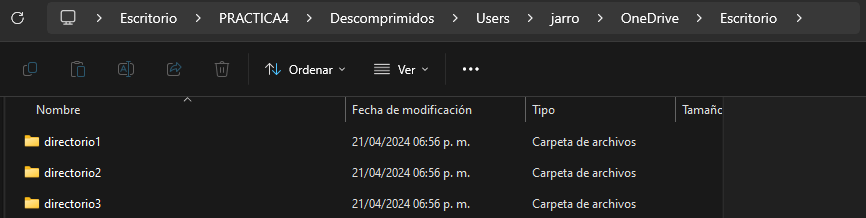


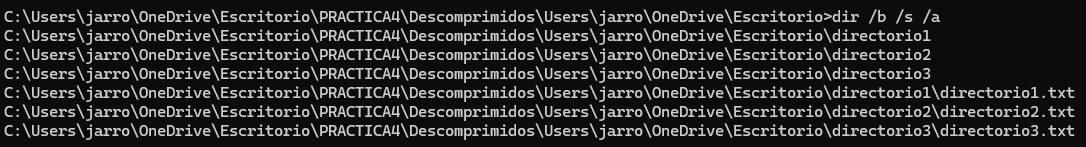
Este script de lotes para Windows presenta un menú con tres opciones: "Directorio", "DirectorioExtendido" y "Salir". El usuario elige una opción ingresando su nombre, y el script ejecuta acciones correspondientes a esa opción. Si se elige "Directorio", muestra el contenido del directorio actual. Si se elige "DirectorioExtendido", imprime "Similar a dir". Si se elige "Salir", termina el script. Si ninguna opción coincide con la entrada del usuario, se muestra "Opción no encontrada". Luego, el script imprime las palabras "Hola", "como", "están" y "todos" en líneas separadas antes de pausar la ejecución y esperar a que el usuario presione una tecla para salir.

6.- Programe un script para que descomprima en sus rutas originales los directorios que se han compreso con el anterior script

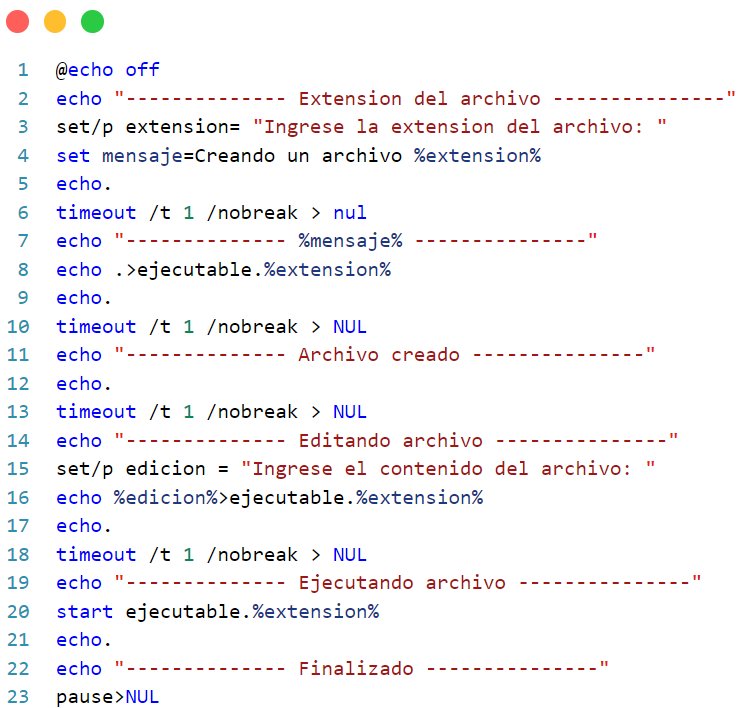


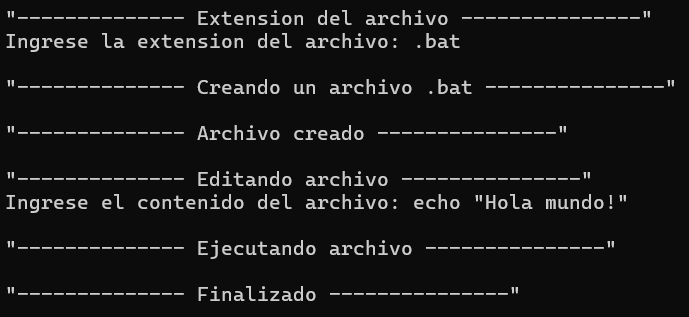


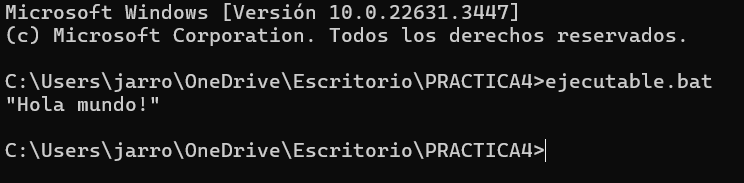




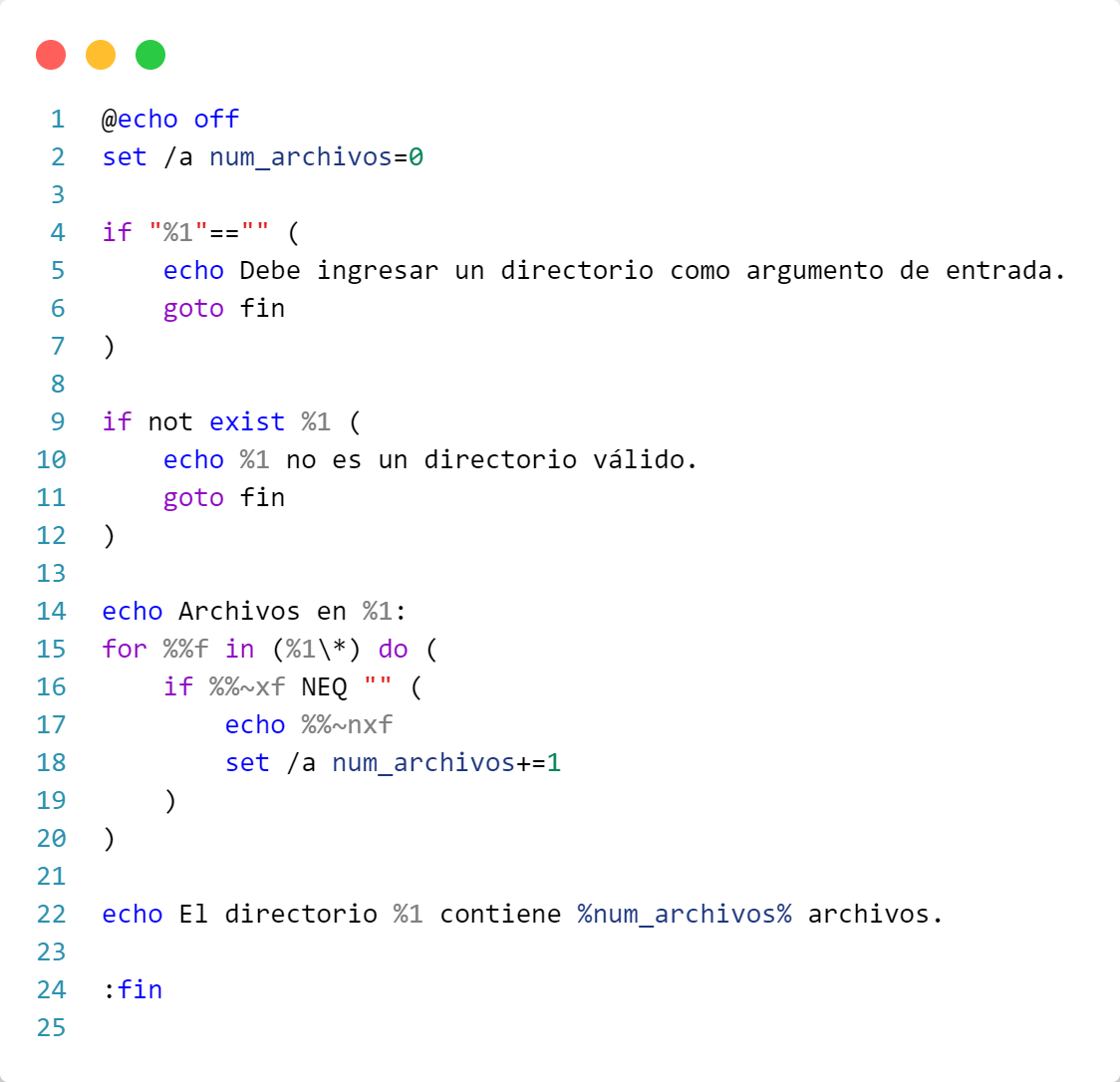
7.- Programe un script generador de scripts, es decir, que cree un archivo, que posteriormente permita editarlo y que finalmente cambie los permisos para ejecución del archivo, mandando a ejecutar el script desde el script original.

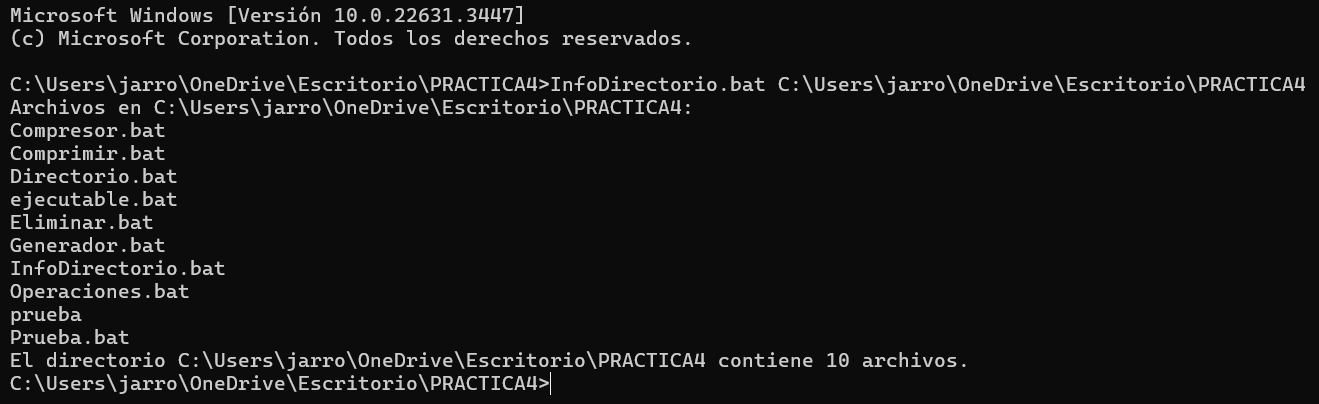




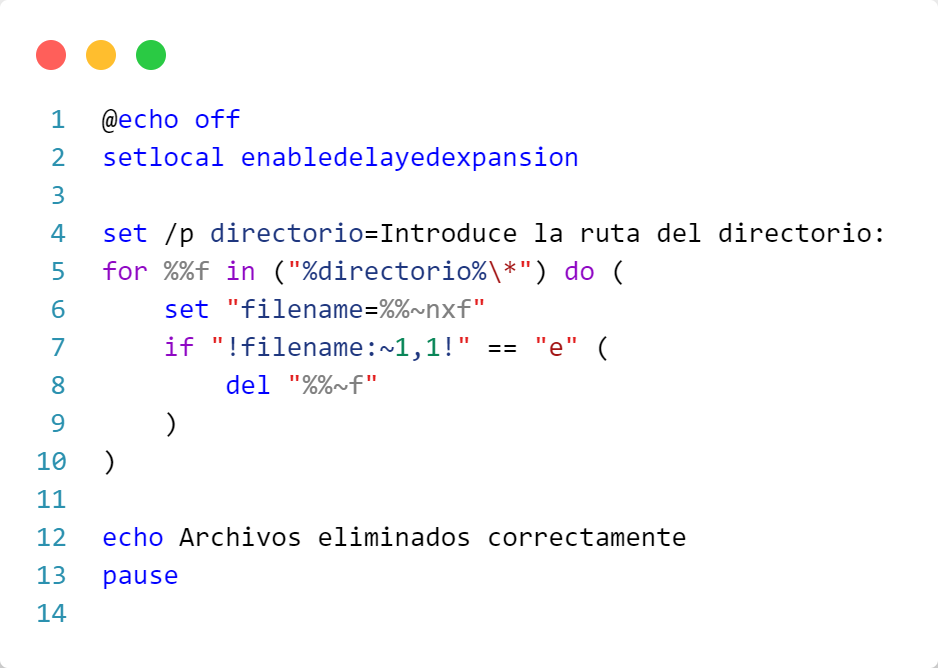


8.- Programe un script a través de las estructuras de control de flujo para que permita visualizar los archivos de un directorio que se envíe como argumento de entrada (no utilizar el comando ls). Además, el script deberá mostrar el número total de archivos en el directorio.

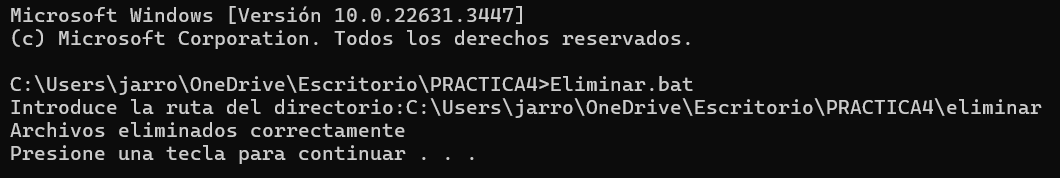


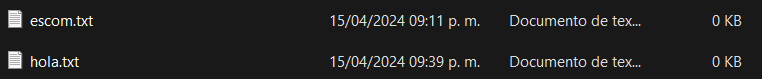


9.- Programe un script que elimine todos los archivos de un directorio especificado desde el propio script, y cuyo segundo carácter sea la letra “e”.









**CONCLUSION**

En conclusión, la programación bajo el intérprete de comandos (shell) en sistemas operativos como Linux y Windows representa una técnica sumamente valiosa para interactuar con el sistema operativo utilizando una interfaz de línea de comandos en lugar de una interfaz gráfica de usuario convencional. Los shells, como Bash en Linux y PowerShell en Windows, ofrecen una amplia gama de herramientas y funcionalidades avanzadas que permiten la programación y la automatización eficientes de tareas del sistema.

Dominar la programación en el intérprete de comandos es una habilidad esencial para administradores de sistemas, desarrolladores de software y cualquier persona que busque mejorar su eficiencia en la realización de tareas del sistema. Los scripts de shell son altamente portátiles entre diferentes sistemas operativos, lo que significa que pueden crearse una vez y ejecutarse en múltiples plataformas, lo que facilita la gestión de sistemas heterogéneos.

En definitiva, la programación en el intérprete de comandos es una herramienta poderosa y flexible que puede mejorar significativamente la productividad y la eficiencia en el trabajo con sistemas operativos Linux y Windows. Además de simplificar tareas cotidianas, la automatización a través de scripts de shell permite a los usuarios concentrarse en tareas más complejas y estratégicas, optimizando así los procesos y recursos del sistema de manera efectiva.