5. Interfaz de líneas de comandos (CLI)

Los sistemas operativos han ido evolucionando en su manera de darle órdenes al sistema. Actualmente, como usuarios, solemos darle órdenes a este mediante la interfaz gráfica de usuario (IGU o GUI, graphical user interface), aunque cada vez con más frecuencia le vamos dando órdenes de voz o gestuales gracias a la NUI (del inglés, natural user interface) o interfaz natural de usuario.

Sin embargo, la primera interfaz empleada en los inicios de la informática fue la llamada CLI (es decir, command line interface) o interfaz de línea de comandos u órdenes.

Esta interfaz todavía resulta útil en la actualidad, sobre todo para los administradores y los desarrolladores, ya que les permite automatizar tareas rutinarias y repetitivas sin más que programarlas.

5.1. ¿Por qué debemos programar scripts?

Todo administrador o desarrollador tiene siempre alguna tarea que lanzar, pero quizá no puede esperar a que terminen las tareas previas para lanzar las siguientes —ya sea porque, en tal caso, se vería obligado a permanecer en su puesto de trabajo más tiempo del necesario o porque dichas tareas tendrían que ejecutarse en horas en que el resto de los usuarios han dejado de trabajar con el sistema—

Para responder a esta situación han surgido los scripts, que permiten programar tareas repetitivas y tediosas, e incluso conjuntos de ellas.

Esto es lo que ocurre, por ejemplo, con la creación, la activación, la desactivación o la eliminación de las copias de seguridad.

Sin embargo, hoy en día, los lenguajes de órdenes más avanzados nos permiten crear aplicaciones muy potentes con los que llevar a cabo cualquier tarea de manera desatendida y sin necesidad de interactuar directamente con el equipo.

5.2. Consola MS-DOS

Para poder abrir la consola de MS-DOS (del inglés, MicroSoft Disk Operating System), surgida en 1981, ejecutaremos la aplicación cmd. Esta nos abrirá una ventana con la propia consola.

En ella, encontraremos una línea de comandos acompañada de la unidad de disco en la que estemos ubicados (habitualmente, C:), la cual hace referencia a nuestro primer disco duro del sistema.

A continuación, se nos proporciona la ruta completa correspondiente a nuestra ubicación. En la siguiente figura podemos ver que se trata de la ruta /Users/raul. Por último, aparece el indicador de separación —en este caso, el símbolo '>'—.

A partir de este indicador, ya podremos lanzar las instrucciones o las órdenes que queramos. Así, por ejemplo, en la figura hemos ejecutado la orden cd, que nos muestra el directorio actual en el que nos hallamos dentro del sistema operativo Windows con el que estamos trabajando actualmente.

Lo mismo ocurre con la orden dir, que nos muestra los archivos que contiene este directorio actual, pues no le hemos asignado ningún argumento.

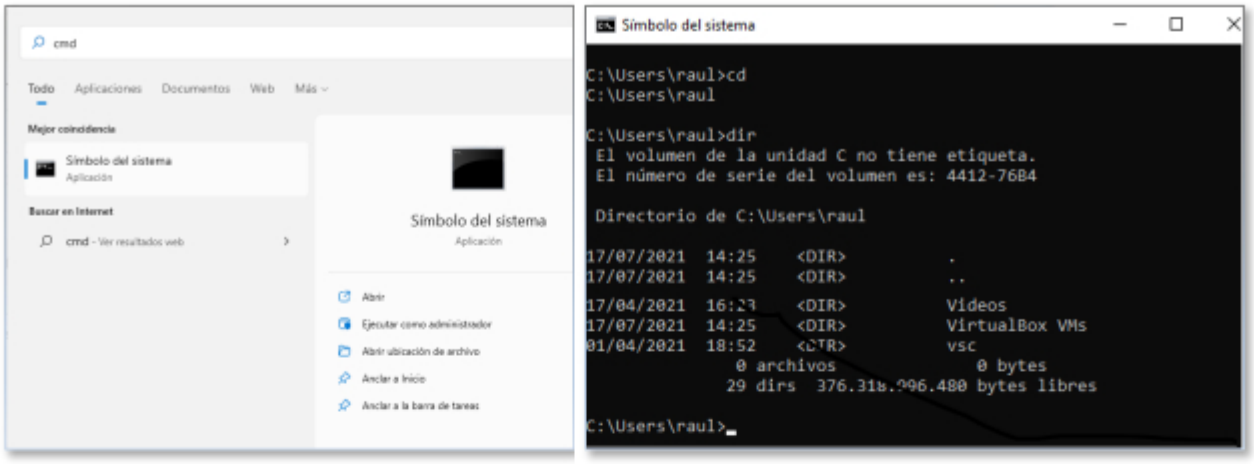

Figura 5.1. Ventana de la consola MS-DOS en Windows 11.

A. Ayuda de órdenes y comentarios en MS-DOS

Lo cierto es que nunca llegaremos a conocer la totalidad de órdenes que existen, ni sus parámetros. Por eso, lo mejor es comenzar aprendiendo a pedir ayuda cuando lo necesitemos.

En este sentido, conviene señalar que esta consola cuenta con la orden de ayuda help. Asimismo, podemos introducir comentarios mediante la orden REM, aunque la forma más habitual es utilizar el símbolo punto y coma ';'.

En la documentación complementaria situada al final de la unidad, en la sección Material de Trabajo > Descarga de documentos, dispones del listado completo de órdenes que la orden *help* muestra por la consola cuando la ejecutamos.

B. Nuestro primer script en MS-DOS

Este es el procedimiento habitual si queremos crear un archivo batch de procesamiento por lotes con la extensión .bat, para poder ejecutarlo en esta consola.

En concreto, abriremos un editor de texto básico —como el bloc de notas (o *notepad*) de Windows— y, dentro de él, ya podremos empezar a escribir en cada línea una orden igual a la que introduciríamos en la consola.

Este sencillo script o guion de órdenes nos permitirá ejecutar más de una al mismo tiempo; así, cuando se termine de ejecutar una, podremos continuar con la siguiente. En el caso práctico ampliado que aparece más abajo, trataremos de hacer algo más elaborado, pero ahora podemos empezar con este primer ejemplo:

```
CLS                        ; Borra la pantalla.
DATE                       ; Muestra o establece la fecha.
ECHO  ¡Hola, mundo MS-DOS! ; Muestra mensajes o activa y desactiva el eco.
VOL                        ; Muestra la etiqueta del volumen y el número de serie del disco.
```

5.3. Consola PowerShell Core

Microsoft creó, en 2006, un novedoso y moderno lenguaje de órdenes orientado a objetos denominado PowerShell, que nos permite crear scripts para automatizar cualquier tarea en Windows. Sin embargo, no fue hasta 2016 cuando decidió publicar el código fuente en GitHub bajo la licencia MIT. De este modo, liberó el proyecto con un nuevo nombre: PowerShell Core. La última versión del mismo, aparecida en marzo de 2011, es la v7.1.3 Release of PowerShell.

A. Ayuda de órdenes y comentarios en PowerShell

Al igual que comentamos en el caso de MS-DOS, nunca llegaremos a conocer la totalidad de órdenes existentes ni tampoco sus parámetros.

Por eso, lo mejor es comenzar aprendiendo cómo pedir ayuda.

Debemos saber que esta consola cuenta con la orden de ayuda Get-Help. Además, podemos introducir comentarios con el símbolo de la almohadilla, '#'.

```
    Get-Help                        # Obtener ayuda genérica.
    Get-Help <comando>        # Obtener ayuda sobre una orden concreta.
    Get-Help <comando> -Full # Obtener ayuda sobre toda la documentación de
una orden.
    Get-Help <comando> -Example    # Obtener ayuda sobre un ejemplo de uso de
una orden.
```

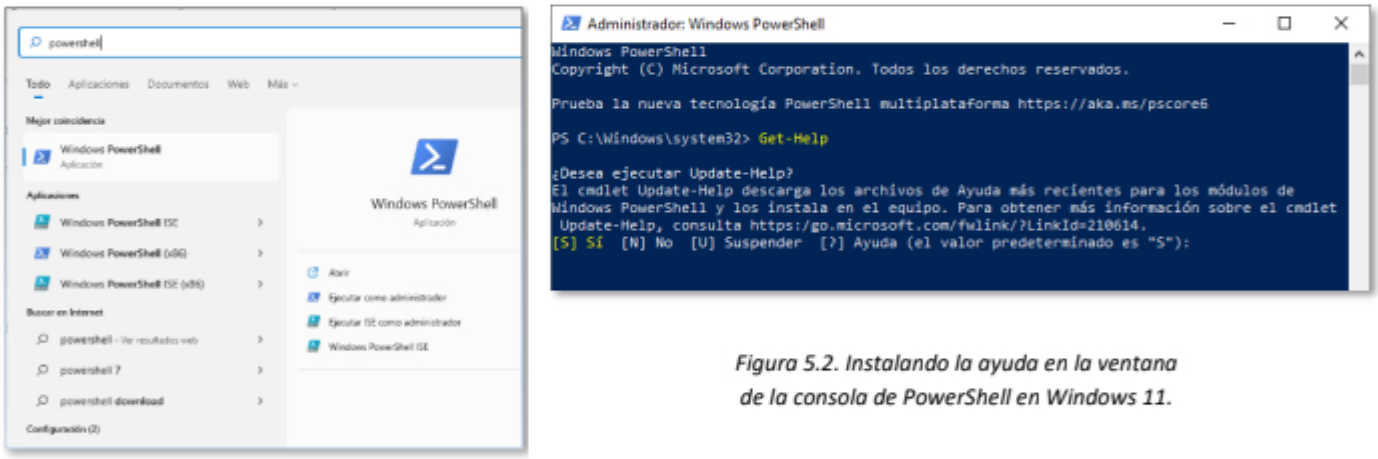

Figura 5.2. Instalando la ayuda en la ventana de la consola de PowerShell en Windows 11.

B. Órdenes de GNU/Linux en PowerShell

Debemos comentar que, desde la liberalización de este proyecto, se han ido creando distintos alias con órdenes nativas de los intérpretes de órdenes como Shell, bash u otros.

Estas órdenes también pueden comprenderse desde powershell y, por tanto, ejecutarse. Así ocurre, por ejemplo, si ejecutamos la orden ls —que es la equivalente al dir de MS-DOS— o si lanzamos en powershell la instrucción Get-ChildrenItem, que facilita la adaptación a la misma por parte de los profesionales más habituados a utilizar GNU/Linux.

Para empezar a conocer mejor todo esto, conviene que empieces por la orden *man*, empleada para mostrar los manuales de ayuda del resto de órdenes.

C. Nuestro primer script en PowerShell ISE

En Powershell, si queremos crear guiones o scripts con varias órdenes, también tendremos que crear un archivo de proceso por lotes dotado de una extensión .ps1. Esto nos permitirá, más tarde, ejecutar las órdenes en su consola.

Este archivo podríamos crearlo con ayuda de un editor de texto básico como el bloc de notas (*notepad*) de Windows.

Sin embargo, en Windows disponemos de una aplicación más potente para crearlos fácilmente con ayuda inteligente a la hora de escribirlos, como ocurre en los IDE actuales.

Esta aplicación es la versión ISE de PowerShell.

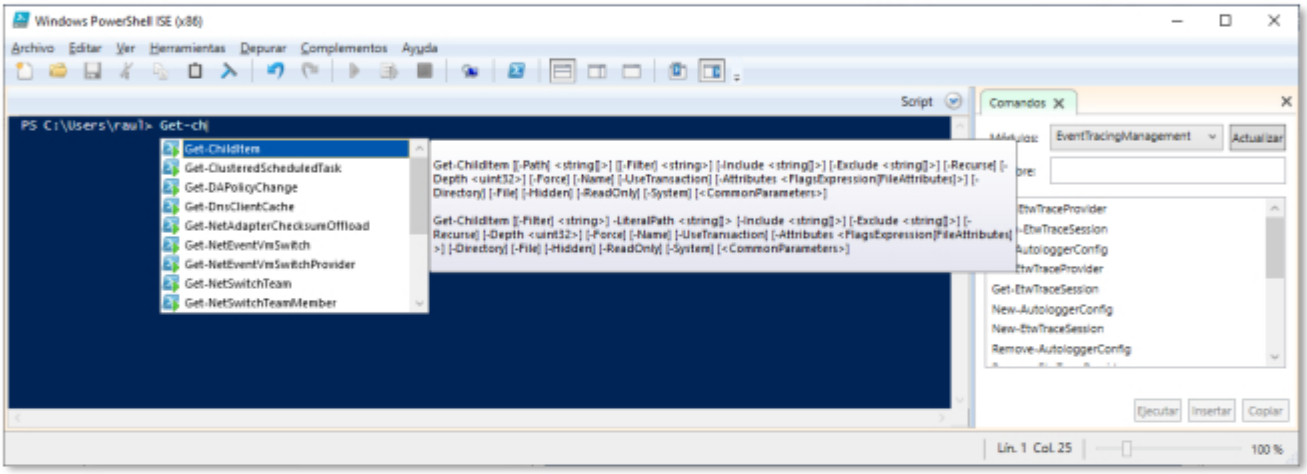

Figura 5.3. Ventana de la aplicación de la consola de PowerShell ISE.

En ella, podemos seguir ejecutando órdenes de manera interactiva, pero la gran potencia de la aplicación se debe a que nos permite crear nuestros scripts.

Para comprobarlo, vamos a crear un nuevo archivo, introduciendo las instrucciones que queramos, y observar cómo funciona nuestro primer script en MS-DOS con powershell.

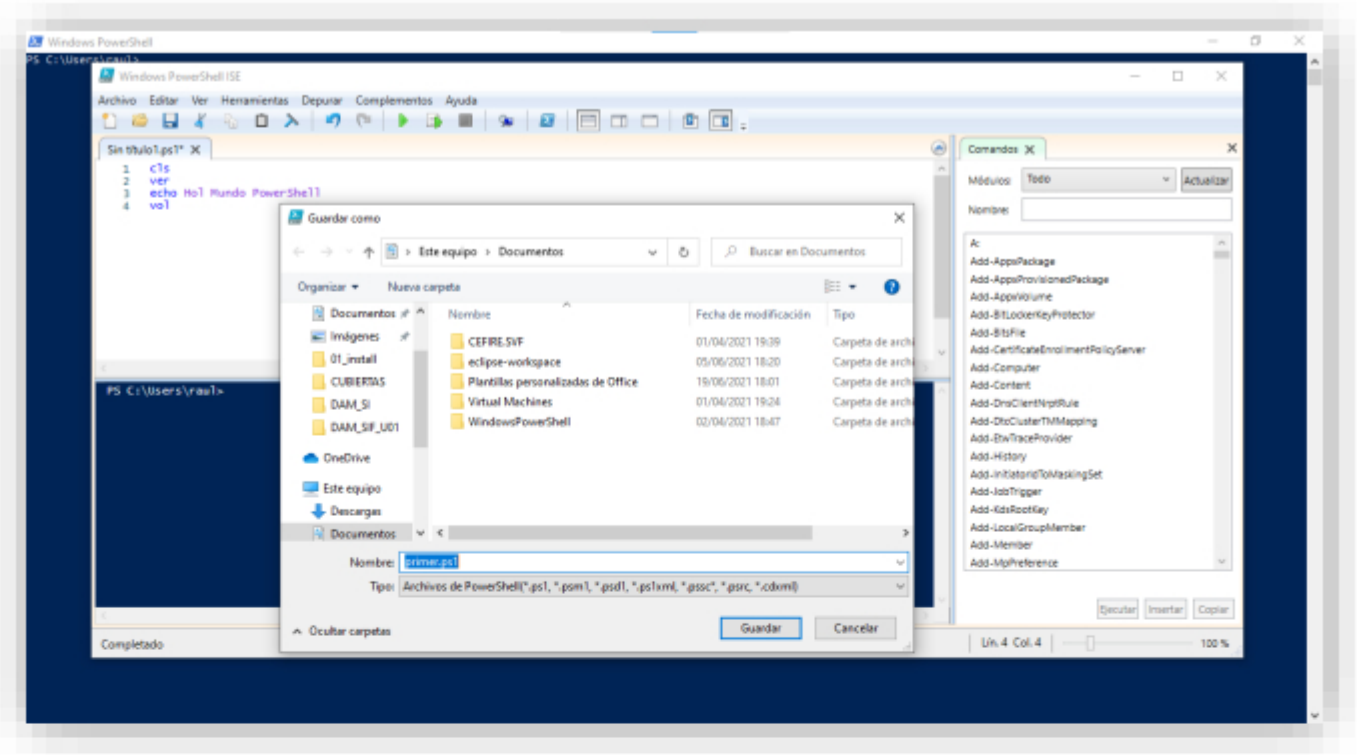

Figura 5.4. Guardando nuestro primer script en la consola de PowerShell ISE.

Realmente han ido apareciendo más tipos de extensiones en el ecosistema de scripts de poweshell, como podemos ver en la figura.

En la siguiente tabla se indica su función correspondiente.

Extensión script powershell	Descripción de su función
*.ps1	Archivo de script.
*.psm1	Archivo de módulo de script.
*.psd1	Archivo de datos de script.
*.ps1xml	Un archivo XML que define datos de tipo extendido.
*.pssc	Archivo de configuración de sesión.
*.psrc	Archivo de funcionalidad de roles.
*.cdxml	Archivo XML que encapsula el Modelo de información común (o CIM, Common Information Model).

Tabla 1. Diferentes tipos de archivos del ecosistema de scripts en PowerShell.

Para más información, puedes consultar este articulo sobre cómo escribir y ejecutar scripts en Windows PowerShell ISE.