Prácticas de Infraestructura Informática

Bloque 3: Infraestructuras de scripting y de gestión en plataformas Windows

Sesión 2 - Fundamentos de PowerShell (II)

Objetivos

Conocer los fundamentos del PowerShell, el lenguaje orientado a scripts de Microsoft que permite automatizar las tareas administrativas en los sistemas Windows.

Desarrollo

- La máquina base para realizar esta práctica es PLX-C-51. Arranca esta máquina e inicia sesión con el usuario local *Alumno*.

• Manejar ficheros de texto

El trabajo con ficheros de texto es esencial en cualquier entorno de trabajo orientado a la gestión de sistemas, ya que estos ficheros son habitualmente utilizados en todo tipo de operaciones de registro e inventario. PowerShell proporciona los comandos *Get-Content*, *Set-Content*, *Add-Content* y *Clear-Content* para manejar ficheros de texto.

Leer el contenido de un fichero

- Utilizando el explorador de archivos, crea el directorio *\Temp* en el volumen C de la máquina de trabajo.
- Utilizando el NOTEPAD, crea el fichero *Datos01.txt* y agrégale el siguiente contenido:

```
Primera fila
Segunda fila
Tercera fila
Cuarta fila
```

- Asegúrate de que no haya ninguna línea en blanco al final del fichero.
- Guarda el fichero.
- Posiciona la consola de PowerShell en el directorio *\Temp* utilizando el comando *Set-Location*.
- Para leer el contenido del fichero y mostrarlo en la consola, ejecuta el siguiente comando:

```
> Get-Content Datos01.txt
```

(1) EJERCICIO. Solicita ayuda sobre el comando Get-Content. Después, utiliza este comando de la forma apropiada para leer solo la primera línea del fichero *Datos01.txt* y enviarla a la consola. Escribe el comando utilizado a continuación.

Agregar contenido a un fichero

Para agregar contenido a un fichero se utiliza el comando *Add-Content*.

(2) EJERCICIO. Solicita ayuda sobre el comando *Add-Content*. Después, utiliza este comando de la forma apropiada para agregar al fichero *Datos01.txt* la siguiente cadena:

```
Quinta fila
```

Escribe el comando utilizado a continuación.

- Abre *Datos01.txt* con el NOTEPAD. ¿Es satisfactorio el resultado obtenido? Hay un aspecto que se puede mejorar. Observa que la línea "Quinta fila" no comienza en una nueva línea. Hay que introducir un carácter de nueva línea en el fichero para obtener el comportamiento deseado.
- Utilizando el NOTEPAD borra "Quinta línea". Asegúrate de que no quede ninguna línea en blanco al final del fichero.

  Ahora se repetirá el ejercicio anterior introduciendo la pareja de caracteres "Carriage return" y "New line" (en este orden) justo antes de "Quinta fila". Esta pareja de caracteres es la que utilizan los editores de texto para ubicar el cursor al comienzo de la línea siguiente. Los caracteres "Carriage return" y "New line" pertenecen a un grupo de caracteres conocidos como secuencias de escape *(scape squences)*. Estos caracteres se representan utilizando el acento invertido *(`)* seguido de otro carácter. Así el carácter "Carriage return" se representa mediante `r, y el carácter "New line", mediante *`n*.

  (3) EJERCICIO. Repite el ejercicio anterior de modo que "Quinta fila" se escriba en una nueva línea del fichero *Datos01.txt*. Escribe el comando utilizado a continuación.

- Utilizando el NOTEPAD comprueba que el resultado es el esperado.

Borrar el contenido de un fichero

Para borrar el contenido de un fichero se utiliza el comando *Clear-Content*.

- Borrar el contenido de *Datos01.txt* ejecutando el siguiente comando:

  ```
  > Clear-Content Datos01.txt
  ```
- Abrir el fichero con el NOTEPAD para comprobar que está vacío.

Establecer el contenido de un fichero

Para establecer un nuevo contenido para un fichero o reemplazar el contenido que tuviese previamente se utiliza el comando *Set-Content*.

(4) EJERCICIO. Solicita ayuda sobre el comando *Set-Content*. Después, ejecuta una línea de comandos que solicite al usuario que introduzca una cadena de caracteres por consola y la guarde en el fichero *Datos01.txt*. No utilices ninguna variable intermedia. En su lugar, haz uso de una canalización. El comando debe imprimir en la consola el texto "Introduce una cadena:", con objeto de guiar al usuario en la operación a realizar. Escribe la línea de comandos ejecutada a continuación.

- Abre el fichero con el NOTEPAD para verificar su contenido.
- Ejecuta otra vez el comando anterior, introduciendo una cadena diferente. Comprueba que el contenido del fichero ha sido reemplazado por la última cadena introducida.

• Ejecución remota

Windows PowerShell tiene la capacidad de ejecutar comandos remotamente. Esta característica es de gran utilidad en la gestión de los equipos de una organización, ya que permite diseñar scripts que realicen operaciones de configuración y mantenimiento en conjuntos de cualquier número de equipos.

Para realizar la ejecución remota, Powershell utiliza una infraestructura de la plataforma Windows conocida como Windows Remote Management (WinRM). Una breve introducción a WinRM está disponible a través de la URL siguiente:

https://learn.microsoft.com/en-us/windows/win32/winrm/portal

- En el explorador del sistema anfitrión, abre la URL anterior y lee los párrafos introductorios. Entonces contesta la siguiente pregunta:

  (5) PREGUNTA. WinRM es la implementación realizada por Microsoft de un protocolo estándar desarrollado por la industria TI. Indica cuál es dicho protocolo.

- Sigue el hipervínculo correspondiente al protocolo indicado en la pregunta anterior. Entonces contesta la siguiente pregunta:

  (6) PREGUNTA. Traduce el párrafo que empieza por "The intent of the protocol is…" y escríbelo a continuación:

- Ahora, sigue el hipervínculo https://dmft.org/standards/wsman, ubicado al final del apartado *Standards*. Este hipervínculo conduce a la página del protocolo WS-MAN, ubicada en el sitio de la DMTF, que es un consorcio de empresas y organizaciones de la industria TI. En la barra de navegación superior, selecciona *About DMTF*. Lee el contenido de esta página y contesta las siguientes preguntas:

  (7) PREGUNTA. ¿Cuál es el cometido del consorcio DMTF?

  (8) PREGUNTA. ¿Cuál es el objetivo de los estándares desarrollados por el consorcio DMTF?

  (9) PREGUNTA. Indica cuatro compañías del consejo de dirección *(board of directors*) del consorcio DMTF.

Habilitar la ejecución remota

Para practicar sobre la ejecución remota, además de PLX-C-51, se va a utilizar la máquina PLX-C-52. Ambas máquinas se encuentran integradas en el dominio *practicas.local*. Si bien es posible hacer gestión remota de equipos no agregados a un dominio, el trabajo dentro de un dominio facilita enormemente las operaciones remotas, ya que el dominio proporciona toda la infraestructura de autenticación requerida para dichas operaciones.

- Arranca el controlador de dominio. No es necesario que inicies sesión, pero sí esperar hasta que se muestre la pantalla de bloqueo, señal de que el controlador ha completado su arranque.
- En principio, PLX-C-51 ya está arrancada y con sesión iniciada por el usuario local *Alumno*. Cierra la sesión abierta con este usuario e inicia una nueva sesión con el *Administrador* del dominio.
- Arranca PLX-C-52 e inicia sesión con el *Administrador del dominio*.

  Vas a utilizar el cliente PLX-C-51 para gestionar remotamente el cliente PLX-C-52.

  WinRM se implementa mediante un servicio de Windows, que se llama también *WinRM* y cuyo nombre para mostrar es *Administración remota de Windows (WS-Management)*.

- En PLX-C-52, abre la consola *Servicios*. Entonces busca el servicio *Administración remota de Windows (WS-Management)*. Observa que *Tipo de inicio* se encuentra configurado en *Manual* y que el servicio se encuentra en el estado *Detenido*. Esta es la configuración estándar para este servicio en los equipos Windows con sistema operativo de tipo cliente (Win10 y Win11).

  Para que el equipo pueda ser gestionado remotamente, este servicio debe ser puesto en ejecución. No obstante, no lo vas a hacer desde la consola *Servicios*, ya que PowerShell proporciona un comando para hacer todas las configuraciones necesarias para que un equipo pueda ser gestionado remotamente. Entre dichas configuraciones se encuentra la apropiada configuración del servicio WinRM.

- En este punto se necesita obtener ayuda sobre el comando *Enable-PSRemoting*. Para ello, no puedes utilizar PLX-C-52, ya que en esta máquina no se han descargado los ficheros de ayuda de PowerShell. Entonces, pasa tu atención a PLX-C-51, abre la consola de PowerShell y solicita ayuda sobre el comando *Enable-PSRemoting*. Lee detenidamente el apartado *DESCRIPCIÓN* de la ayuda de este comando. En este apartado se describen las operaciones realizadas por este comando. Entonces, contesta las siguientes preguntas:

  (10) PREGUNTA. ¿Por qué no es adecuado ejecutar este comando en equipos que solo envían comandos?

  (11) PREGUNTA. ¿Qué actuación realiza el comando *Enable-PSRemoting* sobre el firewall del equipo?

- Pasa tu atención a PLX-C-52, que es la máquina en la que se requiere ejecutar *Enable-PSRemoting*. Entonces, abre una consola de PowerShell y ejecuta el comando tal y como se indica a continuación.

  ```
  > Enable-PSRemoting -force
  ```

  Tras la ejecución de este comando el equipo está listo para recibir comandos remotamente.

- Para comprobar que ahora WinRM se encuentra en ejecución, ejecuta

  ```
  > Get-Service WinRM
  ```

- Para verificar que WinRM se encuentra configurado para la administración remota del equipo, ejecuta el siguiente comando.

  ```
  > winrm quickconfig
  ```

  Deberás obtener la siguiente respuesta:

  ```
  El servicio WinRM ya está ejecutándose en esta máquina.
  WinRM ya está configurado para administración remota en este equipo.
  ```

  (12) PREGUNTA. Entre las configuraciones realizadas para habilitar la gestión remota del sistema se encuentra la apertura del firewall del equipo para una determinada característica. Abre *Panel de control -> Sistema y seguridad -> Firewall de Windows Defender -> Permitir una aplicación o una característica a través de Firewall de Windows Defender*. Observa el panel *Aplicaciones y características permitidas*. ¿Cuál es el nombre de la característica permitida tras la ejecución de *Enable-PSRemoting*?

Aspectos a tener en cuenta en la ejecución remota

- En el equipo en el que se invocan los comandos, la consola de PowerShell debe abrirse con privilegios elevados (ejecutar como *Administrador*). Solamente la cuenta *Administrador* local o la cuenta *Administrador* del dominio no requieren ejecutar con privilegios elevados, ya que, de facto, cuentan con este nivel de privilegio.

- El equipo que invoca los comandos debe presentar unas credenciales de acceso al equipo remoto. Si las credenciales no se indican explícitamente, las credenciales presentadas serán las del usuario con sesión iniciada en el sistema. Las credenciales presentadas deben corresponderse con las de un usuario que se encuentre en el grupo *Administradores* del equipo remoto. Con relación a esto, debe tenerse en cuenta que el administrador del dominio pertenece siempre al grupo local *Administradores* de todos los equipos del dominio. Para comprobar esto, en PLX-C-52 abre el grupo local *Administradores*. Entonces contesta la siguiente pregunta:

  (13) PREGUNTA. ¿Qué grupo de seguridad del dominio agregado al grupo local *Administradores* proporcionará control total del equipo al administrador del dominio?

- Cuando se ejecuta un comando remoto, cualquier fichero, directorio, o cualquier otro tipo de recurso requerido por el comando invocado remotamente debe encontrase disponible en el equipo remoto.

Invocación remota de comandos

Para ejecutar comandos remotamente se utiliza el Cmdlet *Invoke-Command*. Utilizarás este comando en PLX-C-51 para ejecutar comandos remotamente en PLX-C-52.

- En PLX-C-51 abre una consola de PowerShell. Entonces pide ayuda detallada de *Invoke-Command* (parámetro *–detailed*). Observarás que este comando es muy complejo. Para empezar, pondrás tu atención en los parámetros *–ComputerName* y *–ScriptBlock*.
- Lee la información correspondiente al parámetro *–ComputerName*.

(14) PREGUNTA. Indica las tres formas posibles de indicar el computador remoto.

- Lee la información correspondiente al parámetro *–ScriptBlock*.
- Ahora vas a realizar la primera invocación remota. En PLX-C-51 vas a ejecutar un comando que obtiene la información relativa al proceso *Winlogon* ejecutado en PLX-C-52. Para ello ejecuta el siguiente comando:

```
> Invoke-Command –ComputerName PLX-C-52 –ScriptBlock {get-process winlogon}
```

- Observa que en la salida generada por el comando se incluye la columna *PSComputerName*, en la que se indica el nombre de la máquina en la que se ha aplicado el comando.
- Observa el Id del proceso.
- Pasa tu atención a PLX-C-52. Entonces, en esta máquina, ejecuta

```
> Get-Process winlogon
```

- Observa que el Id del proceso obtenido al ejecutar este comando coincide con el obtenido al ejecutar el comando mediante *Invoke-Command* desde el otro ordenador.

A continuación se realizará un conjunto de ejercicios en los que se llevará a cabo una serie de operaciones de administración en PLX-C-52 mediante comandos invocados desde PLX-C-51.

- En PLX-C-52, utilizando la consola *Servicios* comprueba que el servicio *Cola de impresión* se encuentra en el estado *Iniciado*.

  (15) EJERCICIO. En PLX-C-51, invoca un comando que detenga el servicio *Cola de impresión* en PLX-C-52. Escribe el comando invocado a continuación.

- En PLX-C-52, utilizando la consola *Servicios* comprueba que el servicio se ha detenido.

  (16) EJERCICIO. En PLX-C-51, invoca un comando que ponga otra vez en marcha el servicio *Cola de impresión* de PLX-C-52. Escribe el comando invocado a continuación.

- En PLX-C-52, utilizando la consola *Servicios* comprueba que el servicio se ha puesto en ejecución.

  (17) EJERCICIO. En PowerShell, para apagar un ordenador se utiliza el comando *Stop-Computer*. Busca este comando en la ayuda de *PowerShell*. Entonces, utilizando *Invoke-Command*, invoca este comando en PLX-C-51 para apagar PLX-C-52. En el parámetro -ComputerName, para indicar el ordenador local, puedes utilizar el carácter punto (.), o bien *localhost*.

- Una vez comprobado que el equipo se apagó, arranca de nuevo PLX-C-52 iniciando sesión con el administrador del dominio.

Reflexión sobre la ejecución remota en el dominio

La simplicidad con la que estás gestionando equipos remotamente se debe a que los equipos forman parte de un dominio y a que estás utilizando el administrador del dominio para realizar las invocaciones remotas. Las credenciales de este usuario permiten realizar todo tipo de operaciones con los equipos del dominio.

Sesiones remotas

El comando *Invoke-Command* genera una sesión momentánea en el equipo remoto para ejecutar el comando invocado. En el momento en que el comando se ha ejecutado, la sesión se finaliza. Debido a ello, no se mantiene información de "estado" entre las diferentes invocaciones, por lo que las variables usadas en una invocación son desconocidas en la siguiente.

- Para comprobar esto, En PLX-C-51, ejecuta el siguiente comando:

  ```
  > Invoke-Command -ComputerName PLX-C-52 -ScriptBlock {$a=5}
  ```

  Este comando crea la variable $a en el equipo remoto y la inicializa con el valor 5.
- Para obtener el valor de *$a*, En PLX-C-51, ejecuta

  ```
  > Invoke-Command -ComputerName PLX-C-52 -ScriptBlock {$a}
  ```

  Observa que el comando no retorna ningún valor. Esto es debido a que la variable *$a* no ha persistido entre las dos invocaciones.
- En PLX-C-51, ejecuta

  ```
  > Invoke-Command -ComputerName PLX-C-52 -ScriptBlock {$a=5; $a}
  ```

  En este ejemplo sí se obtiene el valor esperado. Esto es debido a que en este ejemplo la variable *$a* se ha utilizado dentro de la misma invocación y, por tanto, dentro de la misma sesión.

Cuando se requiere ejecutar múltiples comandos interrelacionados en un sistema remoto, se debe generar una sesión en dicho sistema e invocar los comandos para la sesión abierta. Para generar sesiones se utiliza el comando *New-PSSession*.

- En PLX-C-51, obtén ayuda detallada sobre el comando *New-PSSession*.

De nuevo se trata de un CmdLet bastante complejo. No obstante, el parámetro más importante es *-ComputerName*, que tiene exactamente el mismo cometido que el utilizado en el comando *Invoke-Command*.

En el caso de *New-PSSession*, toma gran relevancia el valor retornado por el comando. Se trata de un objeto de tipo *PSSession*, que será utilizado para invocar comandos dentro de esa sesión.

- Vas a repetir los comandos anteriores utilizando una sesión.
- Primero crearás una sesión remota en PLX-C-52. Para ello, en PLX-C-51, ejecuta el siguiente comando:

  ```
  > $Se = New-PSSession -ComputerName PLX-C-52
  ```

  Un objeto de tipo *PSSession* quedará guardado en la variable *$Se*.

- Para ver el contenido del objeto de tipo sesión almacenado en $Se, ejecuta

  ```
  > $Se
  ```

  Entre otros datos, se puede ver el ordenador al que se encuentra asociada la sesión.

  Ahora se utilizará el comando *Invoke-Command*, primero para crear una variable y asignarle un valor, y después, para acceder al valor de la variable. Sin embargo, ahora se utilizará *Invoke-Command* con el parámetro –Session, en vez de –ComputerName. Estos dos parámetros son incompatibles. O se utiliza uno, u otro.

- Para crear la variable, En PLX-C-51, ejecuta el siguiente comando:

  ```
  > Invoke-Command –Session $Se –ScriptBlock {$a=5}
  ```

- Para acceder al valor de la variable, en PLX-C-51, ejecuta el siguiente comando:

  ```
  > Invoke-Command –Session $Se –ScriptBlock {$a}
  ```

  Ahora sí se accede a la variable correctamente, ya que la creación y el acceso a la misma se realizan dentro de la misma sesión.

- Para conocer otros comandos relativos al manejo de sesiones, en PLX-C-51 ejecuta

  ```
  > Get-Command *Session* -Type CmdLet
  ```

- Para conocer las sesiones remotas abiertas, en PLX-C-51 ejecuta:

  ```
  > Get-PSSession
  ```

- El comando anterior te proporciona el identificador Id y el nombre de la sesión abierta en este instante.

  (18) EJERCICIO. En PLX-C-51, busca el comando necesario para eliminar sesiones y ejecútalo para eliminar la sesión abierta. Indica el comando ejecutado a continuación.

- Ejecuta el comando *Get-PSSesion*, para comprobar que la sesión se ha eliminado.

  (19) EJERCICIO. En PLX-C-51, abre una nueva sesión en PLX-C-52 y almacénala en la variable $Se. Entonces, ejecuta un comando para eliminar la sesión cumpliendo las siguientes restricciones: 1) debes utilizar el parámetro Id, y 2) debes utilizar la variable *$Se*. Indica el comando ejecutado a continuación.

- Ejecuta el comando *Get-PSSesion*, para comprobar que la sesión se ha eliminado.

  (20) EJERCICIO. En PLX-C-51, crea una nueva sesión en PLX-C-52 y almacénala en la variable *$Se*. Entonces, mediante *Invoke-Command*, ejecuta la secuencia de comandos necesarios para crear en PLX-C-52 una carpeta llamada *Temp* en el directorio raíz de la unidad C, y un fichero de texto llamado *Prueba.txt* en la carpeta *Temp*. El fichero debe contener el mensaje "PRUEBA". La secuencia debe terminar con el cierre de la sesión abierta. Cada operación a realizar se llevará a cabo en un comando *Invoke-Command* diferente. Escribe la secuencia de comandos necesarios a continuación.

- En PLX-C-52, comprueba que se ha creado la carpeta *C:\Temp*, así como el fichero *Prueba.txt* con el contenido requerido.
- A partir de este punto, la máquina PLX-C-52 no es necesaria. Apágala.
- Asimismo, el controlador de dominio, tampoco será necesario para el resto de la sesión. Apágalo.

- **Ejecución de aplicaciones externas**
  - Todo este apartado se realizará en PLX-C-51, utilizando el usuario local *Alumno*. En este momento, en PLX-C-51, el usuario que tiene iniciada sesión es el *Administrador del dominio*. Cierra la sesión abierta con este usuario e inicia una nueva sesión con *Alumno*.

Ejecución mediante la especificación de la ruta completa

Hasta ahora solo se ha utilizado la consola de PowerShell para ejecutar CmdLets. Sin embargo, al igual que otras consolas de comandos, la consola de PorwerShell también permite ejecutar aplicaciones externas a PowerShell.

- Para ejecutar una aplicación externa, se introduce en la línea de comandos de PowerShell el nombre de la aplicación a ejecutar junto con su ruta completa. Por ejemplo, supón que se desea ejecutar el programa *Notepad.exe*. La ruta completa de este programa es la siguiente: *\Windows\system32\Notepad.exe*. Por consiguiente, para ejecutar este programa desde PowerShell, ejecuta la siguiente línea de comandos:

```
> \Windows\system32\NOTEPAD.exe
```

La variable de entorno *PATH*

El sistema operativo mantiene un conjunto de variables con información de configuración del sistema. Estas variables reciben el nombre de variables de entorno *(environment variables)*.

Entre las variables de entorno se encuentra la variable *Path*. Esta variable mantiene las rutas de búsqueda por defecto del sistema, es decir, cuando se ejecuta un programa desde una consola de comandos sin especificar la ruta del programa, se busca dicho programa en las rutas indicadas en la variable *Path*. Si se encuentra en estas rutas, el programa se ejecuta. De esta forma, la variable PATH facilita la ejecución de programas habituales, evitando tener que teclear la ruta completa de estos programas.

- Para manejar las variables de entorno, PowerShell proporciona la unidad *Env*. Para ver las unidades disponibles ejecuta el siguiente comando:

```
> Get-PSDrive
```

- Observa entre ellas la unidad *Env*.
- Para posicionar la consola en esta unidad, ejecuta

```
> Set-Location Env:
```

- Para ver el contenido de esta unidad, es decir, para ver las variables de entorno, ejecuta

```
> Get-ChildItem
```

- Entre todas las variables, puedes observar la variable *Path*. Sin embargo, el contenido de esta variable no se ve completamente, debido al formato de salida por defecto del comando *Get-ChildItem*. Para ver el contenido completo, se puede pasar la salida generada por *Get-ChildItem* a *Format-List*. Para ello ejecuta el siguiente comando:

```
> Get-ChildItem Path | Format-List
```

- Entre las rutas contenidas en la variable *Path* (separadas por el carácter ';') se observa *C:\Windows\Sytem32*. En esta ruta es donde se encuentra el programa *Notepad.exe*. Debido a ello, no será necesario escribir la ruta completa de este programa para ejecutarlo.
- Primero, vuelve a posicionar la consola en el volumen C. Para ello, ejecuta

```
> Set-Location C:
```

- Finalmente, ejecuta *Notepad.exe* sin incluir su ruta.

```
> Notepad.exe
```

- **Scripts**

  Los scripts de PowerShell son ficheros de texto que contienen secuencias de comandos y expresiones de PowerShell. Cualquier operación que se pueda realizar en la línea de comandos puede ser incluida en un script. Los scripts de PowerShell deben tener obligatoriamente la extensión `ps1` y pueden ser escritos mediante cualquier editor que genere texto plano, como por ejemplo el *Notepad*.

  Carpeta para el almacenamiento de scripts

  - Será conveniente crear una carpeta para almacenar los scripts que se programarán en la realización de estas prácticas. Se creará esta carpeta utilizando la consola de PowerShell. En este momento, la consola debe estar ubicada en el directorio raíz del volumen C. Si no es así, primero de todo, ubica la consola en esta posición. A continuación, ejecuta el comando siguiente:

    ```
    > New-Item Scripts -Type Directory
    ```

  - Ahora posiciona la consola en el directorio creado ejecutando el comando siguiente:

    ```
    > Set-Location Scripts
    ```

  Mostrar las extensiones de los archivos

  Por defecto, el sistema operativo está configurado para NO mostrar las extensiones de archivo para los tipos de archivos conocidos. No obstante, es posible que ya hayas cambiado esta configuración en sesiones anteriores. No mostrar las extensiones de archivo resulta un tanto incómodo cuando se manejan scripts que son de tipo texto, pero cuyos ficheros tendrán extensiones diferentes a .txt.

  - Comprueba el estado en el que se encuentra la configuración de visualización de las extensiones de archivo. Para ello, *Panel de control -> Apariencia y personalización -> Opciones del Explorador de archivos ->* Pestaña *Ver*. Entonces, comprueba que la casilla *Ocultar las extensiones de archivo para tipos de archivo conocidos* no está seleccionada. En el caso de que lo esté, elimina la selección.

  Creación y ejecución del primer script

  En este punto crearás el primer script de PowerShell con una funcionalidad mínima. Este script servirá para analizar la problemática relativa a las políticas de ejecución de scripts.

  - Utilizando el explorador de archivos, abre la carpeta *\Scripts*.
  - Utilizando el *Notepad*, crea en esta carpeta un archivo llamado *Prueba01.ps1*. Escribe en él el comando *Get-Alias* y guarda el archivo.

    Para ejecutar un script se introduce el nombre del script junto con su ruta en la línea de comandos de PowerShell. (NOTA: aunque el script a ejecutar esté en el directorio actual, es necesario introducir la ruta completa del script. Esto es debido a que PowerShell no busca por defecto en el directorio actual. No obstante, si el script está en este directorio, la ruta puede indicarse mediante los caracteres '.\', ya que el punto significa el directorio actual).

  - Ubica la consola de PowerShell en el directorio *c:\Scripts*.
  - Ejecuta el script introduciendo el comando siguiente:

    ```
    > .\Prueba01.ps1
    ```

  - El intento de ejecución del script generará un error:

    (21) EJERCICIO. Indica a continuación la primera frase (o sea, la descripción) del error indicado.

  Políticas de ejecución de scripts

  Para conocer las políticas de ejecución de scripts se puede solicitar a PowerShell ayuda conceptual. Hasta ahora solo se ha utilizado ayuda sobre comandos, pero PowerShell también proporciona un

conjunto de temas de ayuda orientados a explicar conceptos generales. Estos temas de ayuda comienzan siempre con la cadena de caracteres "*about_*".

- Para ver el catálogo de temas de ayuda conceptual, ejecuta el siguiente comando:

  ```
  > Get-Help about_*
  ```

- Observa que uno de estos temas es *about_Execution_Policies*. Solicita ayuda de este tema. Lee el apartado *PowerShell execution policies*. En este apartado se explican las directivas (o políticas) de ejecución de PowerShell.
- Observa que la política *Restricted* es la predeterminada en el caso de los sistemas Windows de tipo cliente, como es el caso de Windows 11. Esta política no permite la ejecución de scripts. Esta es la razón por la que no se ha ejecutado con éxito el script *Prueba01.ps1*. Sin embargo, la política *Unrestricted* permite ejecutar todos los scripts, estén firmados o no digitalmente. Esta es la política que se utilizará en estas prácticas para poder realizar pruebas sin ningún tipo de restricción.
- Para conocer la política de ejecución actual, ejecuta el siguiente comando:

  ```
  > Get-ExecutionPolicy
  ```

  El resultado debe ser *Restricted*.
- Para cambiar la política de ejecución a *Unrestricted*, ejecuta el siguiente comando:

  ```
  > Set-ExecutionPolicy Unrestricted
  ```

- Se deniega el acceso al cambio de la política. Esto se debe a que para hacer este cambio, deber abrir la consola con derechos elevados.
- Cierra la consola de PowerShell y ábrela de nuevo ejecutándola como administrador.
- Vuelve a ejecutar el comando

  ```
  > Set-ExecutionPolicy Unrestricted
  ```

- En esta ocasión, el comando debe ejecutarse con éxito.
- Ahora vuelve a ejecutar *Get-ExecutionPolicy* para comprobar que la política de ejecución de scripts se ha cambiado satisfactoriamente.
- Reubica la consola en *C:\Scripts*.
- Con la nueva configuración de la política de ejecución, el script *Prueba01.ps1* debe poder ejecutarse sin problemas. Ejecútalo mediante el comando siguiente:

  ```
  > .\Prueba01.ps1
  ```

• PowerShell ISE

Para crear un script de PowerShell se puede utilizar cualquier editor de textos, como por ejemplo, *Notepad*. Sin embargo, la creación de un script de una mínima complejidad no es una tarea en absoluto sencilla. El programador puede necesitar hacer pruebas sobre los comandos a incluir en el script, obtener realimentación de la ejecución de los mismos, obtener ayuda detallada, trabajar con varios scripts simultáneamente, etc. Debido a ello, para mejorar la productividad de los programadores en la generación de scripts, la plataforma Windows proporciona la herramienta *PowerShell ISE (PowerShell Integrated Scripting Environment)*. Se trata de una herramienta que permite la creación, ejecución y prueba de scripts en un entorno amigable para el usuario.

En la creación de scripts de una mínima complejidad se recomienda siempre la utilización de *PowerShell ISE*, frente al uso de cualquier otro editor de textos de propósito general.

Funcionamiento de PowerShell ISE

- Arranca *Windows PowerShell ISE*.

Área de trabajo de PowerShell ISE

El área de trabajo de *PowerShell ISE* se estructura en dos paneles: el panel de comandos y el panel de scripts.

- Cuando se arranca *PowerShell ISE*, el panel de scripts puede encontrarse en estado visible u oculto. Para conmutar entre ambos estados se utiliza un botón con una flecha denominado *Script*, que se ubica debajo de la barra de herramientas. Pulsa este botón para mostrar/ocultar el panel de scripts. Finalmente, deja dicho panel en estado visible.
  - El *panel de scripts* (panel superior) se utiliza para escribir scripts. Este panel se organiza en pestañas. Cada pestaña se utiliza para desarrollar un script diferente. Se pueden gestionar un máximo de 8 pestañas simultáneamente. La secuencia de comandos escrita en cualquier pestaña de este panel, se puede guardar en un fichero con extensión .ps1, generándose de esta forma el script correspondiente.
  - El *panel de comandos* (panel inferior) proporciona la misma funcionalidad que una consola de comandos de PowerShell. Así, este panel permite ejecutar comandos en modo interactivo. El usuario puede utilizar este panel para realizar pruebas sobre los comandos a incluir en el script que se esté desarrollando. Asimismo, la salida generada por la ejecución de los scripts también se muestra en este panel.
- A modo de ejemplo, escribe en el panel de comandos el cmdlet *Get-Process*. Observa la ayuda sensible al contexto proporcionada por el PowerShell ISE, que muestra los comandos disponibles según vas tecleando. Finalmente, ejecuta *Get-Process* y observa la salida. Es idéntica a la proporcionada por la consola de comandos de PowerShell.

Creación de un script mínimo

A continuación se utilizará la herramienta *PowerShell ISE* para crear un script mínimo. El objetivo del script es buscar en el directorio actual el fichero que ha sido modificado más recientemente y escribir su nombre, así como su fecha y hora de última modificación en la consola. Primero debe escribirse el nombre y a continuación su fecha y hora, todo ello en la misma línea.

Un posible diseño de este script se puede basar en utilizar una variable que almacene todos los objetos de tipo fichero del directorio actual, ordenados por su fecha y hora de última modificación. Una vez almacenada dicha colección de objetos en la variable, se escribe en la consola las propiedades relativas al nombre, así como fecha y hora de última modificación del último objeto almacenado, ya que si los objetos se han almacenado ordenados por fecha y hora, el último objeto es el que contiene la información deseada.

- Antes de crear el script, se preparará un directorio con varios ficheros. Este directorio será utilizado para hacer las pruebas del script. En este punto, se encuentra disponible en PLX-C-51 el directorio *Temp*. Si hay algún fichero en este directorio, bórralo.
- Utilizando el explorador de archivos vas a crear en *Temp* tres ficheros de texto con los nombres Fichero01.txt, Fichero02.txt y Fichero03.txt. Para ello, seguirás las indicaciones que se proporcionan a continuación. El explorador de archivos muestra el campo *Fecha de modificación*, que indica la última vez que el fichero ha sido modificado. Al crear los ficheros, debes conseguir que este campo refleje como mínimo una diferencia de un minuto entre cada fichero, de modo que Fichero03.txt sea el último modificado y Fichero01.txt, el que hace más tiempo que ha sido modificado. Simplemente, espera un minuto tras la creación de un fichero antes de crear el siguiente.
- Antes de comenzar a diseñar el script, ubicar la consola de *PowerShell ISE* en el directorio \Temp. Para ello ejecuta:

  ```
  > Set-Location \Temp
  ```
- Para obtener todos los ficheros del directorio actual y almacenarlos en una variable llamada *$Ficheros*, ejecuta el comando siguiente:

  ```
  > $Ficheros = Get-ChildItem
  ```
- Sin embargo, el comando anterior no garantiza que los ficheros del directorio actual se almacenen en la variable $Ficheros ordenados por fecha y hora de última modificación. Para esto, será necesario utilizar el comando *Sort-Object* de la forma apropiada. En este punto, se

puede utilizar la consola de *PowerShell ISE* para obtener ayuda de *Sort-Object*. Ejecuta en la consola el comando siguiente:

```
> Sort-Object -?
```

- Entre los parámetros de *Sort-Object* observarás *–Property*, que es el parámetro a utilizar para ordenar en base a una determinada propiedad. Sin embargo, ¿cuál será el nombre de la propiedad de los objetos de tipo fichero que indica la última fecha y hora de modificación del fichero? Para conocer esto, se puede utilizar el comando Get-Member en la consola. Ejecuta el comando siguiente:

```
> Get-ChildItem Fichero01.txt | Get-Member
```

- El comando *Get-Member* proporciona los métodos y propiedades del objeto de tipo fichero. Entre las propiedades puedes observar *LastWriteTime*. Para probar esta propiedad, ejecuta en la consola los siguientes comandos:

```
> $a = Get-ChildItem Fichero01.txt
> $a.LastWriteTime
```

- Una vez realizadas estas pruebas, se puede plantear el primer comando del script. Dicho comando puede ser el siguiente:

```
> $Ficheros = Get-ChildItem | Sort-Object –Property LastWriteTime
```

- Antes de introducir este comando en el script, ejecútalo en la consola para explorar el contenido de la variable *$Ficheros*.
- Si el directorio actual tiene más de un fichero, como es el caso de este ejemplo, la variable *$Ficheros* contendrá una colección de objetos de tipo fichero (o más exactamente, de tipo FileInfo). Para conocer las propiedades de esta colección de objetos, ejecuta el siguiente comando en la consola:

```
> Get-Member –InputObject $Ficheros
```

- Observarás que una de las propiedades de esta colección de objetos es *Count*. Esta propiedad indica el número de objetos almacenados en la variable.

(22) EJERCICIO. Escribe a continuación la sentencia que te permite ver el contenido de la propiedad *count*.

- Prueba dicha sentencia en la consola de Powershell ISE.
- Utilizando la propiedad *count* se puede acceder al último objeto almacenado en la variable, que gracias a la ordenación realizada, contendrá la información relativa al fichero deseado (el último escrito). Por ejemplo, para obtener el nombre de este fichero, prueba el siguiente comando:

```
> $Ficheros[$Ficheros.count-1].name
```

  Asegúrate de que se comprendes bien el comando anterior.
- En este punto, ya se han hecho todas las pruebas necesarias para escribir el script. Entonces, en el panel de scripts escribir las siguientes líneas de comandos:

```
1 $Ficheros = Get-ChildItem | Sort-Object –Property LastWriteTime
2 Write-Host $Ficheros[$Ficheros.count-1].name, $Ficheros[$Ficheros.count-1].LastWriteTime
```

- Para salvar el script, menú *Archivo -> Guardar*. Entonces seleccionar la carpeta \Scripts, y guarda el script con el nombre *Prueba02.ps1*.

Ejecución del script

- Menú *Depurar -> Ejecutar o continuar*. O bien, mediante la flecha verde de la barra de herramientas.
- Comprueba que después de la operación anterior, en el panel de salida se obtiene el resultado esperado (Fichero03.txt).
- Utilizando el explorador de archivos, crea en *\Temp* un nuevo fichero llamado Fichero04.txt. Al crearlo en este momento, será el último fichero escrito.

- Vuelve a ejecutar el script *Prueba02.ps1*, comprobando que genera como resultado Fichero04.txt.
- Finalmente, realiza una modificación cualquiera en Fichero01.txt y guarda el fichero. Entonces ejecuta de nuevo el script *Prueba02.ps1*, comprobando que genera como resultado Fichero01.txt.

• Acceso a los scripts desarrollados en PLX-C-51

En este apartado se plantea el problema de cómo acceder a los scripts desarrollados en PLX-C-51, por ejemplo, para copiarlos en tu *pendrive*. Para ello, es necesario acceder a los scripts almacenados en PLX-C-51 desde el sistema anfitrión.

La solución es compartir la carpeta *C:\Scritps* de PLX-C-51, y acceder a ella desde el anfitrión utilizando el identificador UNC de esta carpeta, o sea, utilizando la nomenclatura siguiente:

\\servidor\recurso_compartido

- En PLX-C-51, comparte la carpeta *C:\Scritps*. Por defecto, se autoriza el acceso a esta carpeta al usuario *Alumno*. No es necesario autorizar a otros usuarios.

  (23) PREGUNTA. ¿Qué identificador UNC debes utilizar en el anfitrión para acceder a la carpeta *Scritps* en PLX-C-51? Indícalo a continuación.

- En el anfitrión, utilizando la herramienta *Buscar*, introduce el identificador UNC de la carpeta *Scritps*. Se solicitan credenciales de acceso. Utiliza las del usuario *Alumno*. En este punto se establece el acceso al recurso y puedes acceder a los scripts.