Prácticas de Infraestructura Informática

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

Sesión 3 - Programación de scripts en PowerShell

Objetivos

Realizar ejemplos de scripts con PowerShell ISE.

Desarrollo

- Todos los scripts indicados en esta práctica se desarrollarán utilizando PowerShell ISE en la máquina PLX-C-51, con el usuario local *Alumno*.

- **Ejemplos de scripts**

  ### Creación de un script sin control de flujo

  En este primer ejemplo se creará un script que ejecutará un conjunto de comandos en secuencia sin utilizar estructuras condicionales ni bucles.

  (1) EJERCICIO. El objetivo de este ejercicio es construir un script que cree un fichero con información relativa a un proceso, cuya identificación es introducida por el usuario a través de la consola. El script debe realizar las siguientes tareas:

  - Enviar a la consola un mensaje explicativo de su cometido. El mensaje a enviar es el siguiente: "`Script que genera un fichero informativo de un proceso indicado mediante su ID`".
  - Solicitar al usuario el ID del proceso. Para ello, se ha de enviar a la consola el siguiente mensaje: "`Introduce el ID del proceso`"."
  - Solicitar al usuario la ruta del directorio en el que se creará el fichero informativo. Para ello, se ha de enviar a la consola el siguiente mensaje: "`Introduce la ruta de la carpeta en la que se creará el fichero informativo del proceso`".
  - Con el ID del proceso y la ruta, el script debe generar un fichero cuyo nombre sea el mismo que el del proceso + la extensión *.txt*. El script escribirá en este fichero en líneas consecutivas el nombre, el ID y la hora de inicio del proceso. A continuación se proporciona un ejemplo del contenido almacenado en este fichero:

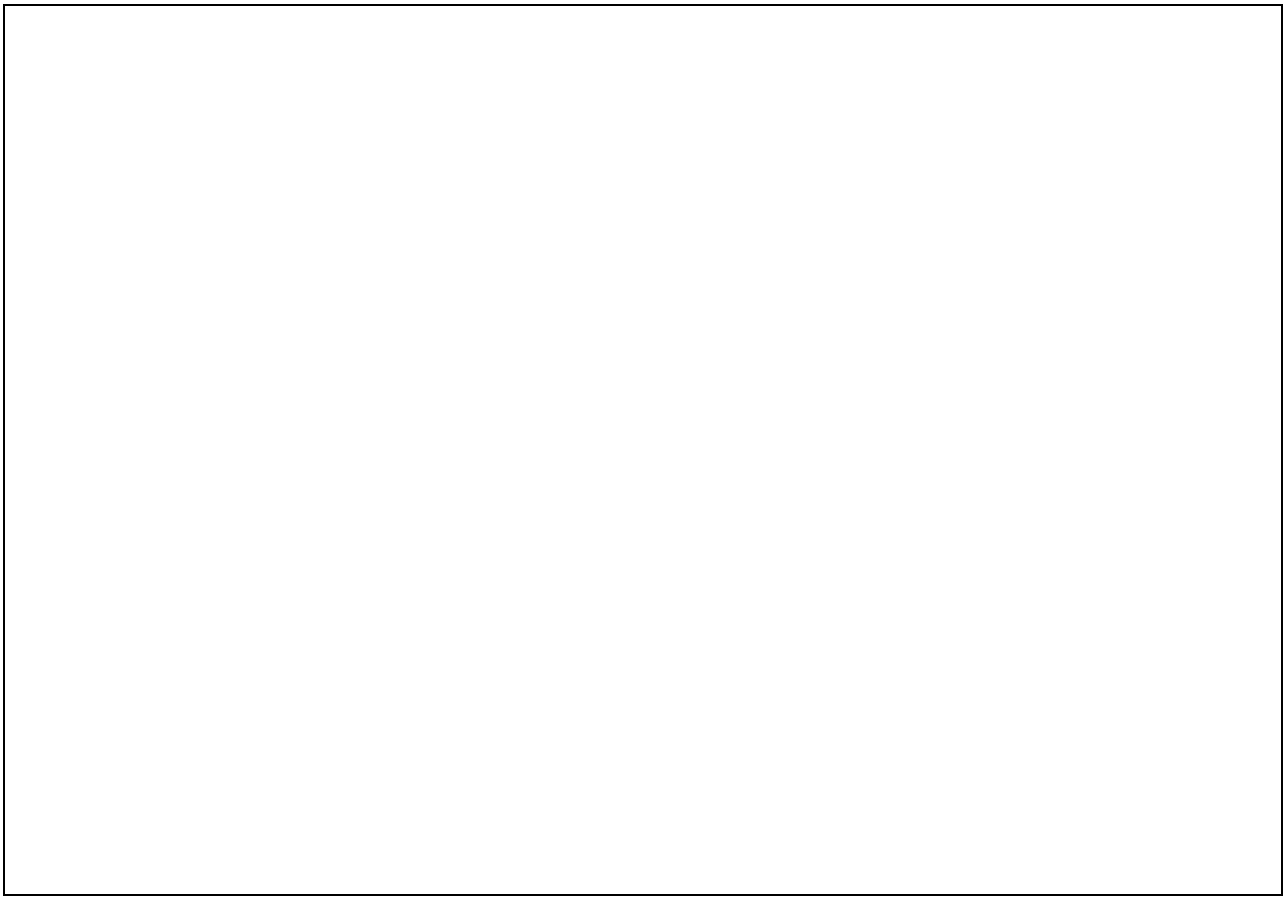
    ```
    Nombre: explorer
    ID: 1248
    Hora de inicio: 02/04/2021 14:08:21
    ```

  - **Escribe este script con Powershell ISE y almacénalo en la carpeta C:*Scripts* con el nombre *Prueba03.ps1*.**

    NOTAS:

  - En la realización de este script puede ser útil la creación de una variable intermedia para almacenar el nombre completo del fichero (con la extensión .txt), junto con su ruta. Para esta variable se propone el nombre siguiente: *$Ruta_Completa*.
  - Será necesario definir otras variables además de la indicada anteriormente.
  - Será necesario también utilizar una técnica muy habitual en el scripting: la concatenación de cadenas. En PowerShell las cadenas de caracteres se concatenan con el operador '+'. Así la expresión "abc"+"def" genera la cadena "abcdef".

- Para introducir comentarios se utiliza el carácter #.
- Puedes utilizar la carpeta "*C:\Temp*", creada en la sesión de prácticas anterior, para almacenar los ficheros informativos. Si hay ficheros almacenados en esta carpeta, elimínalos.



Creación de un script con control de flujo básico

En este segundo ejemplo se trata de probar dos ejemplos de estructuras de control de flujo: una estructura condicional y un bucle.

(2) EJERCICIO. En este ejercicio se plantea la creación de un script que defina un *array* conteniendo los números del 0 al 9. El script debe ejecutar un bucle que recorra todos los elementos del *array* e imprima en la consola solamente los pares.

Este script se realizará con la herramienta PowerShell ISE y se almacenará en la carpeta C:*Scripts* con el nombre *Prueba04.ps1*.

NOTAS:

- Para la realización de este script se utilizará un bucle de tipo *Do-While*. Solicita ayuda conceptual sobre esta estructura de programación ejecutando el comando:

  ```
  > Get-Help about_do
  ```

  En este script será necesario evaluar condiciones. Las condiciones se plantean utilizando operadores de comparación. Para obtener ayuda sobre este tipo de operadores, ejecuta el comando:

  ```
  > Get-Help about_Comparison_Operators
  ```

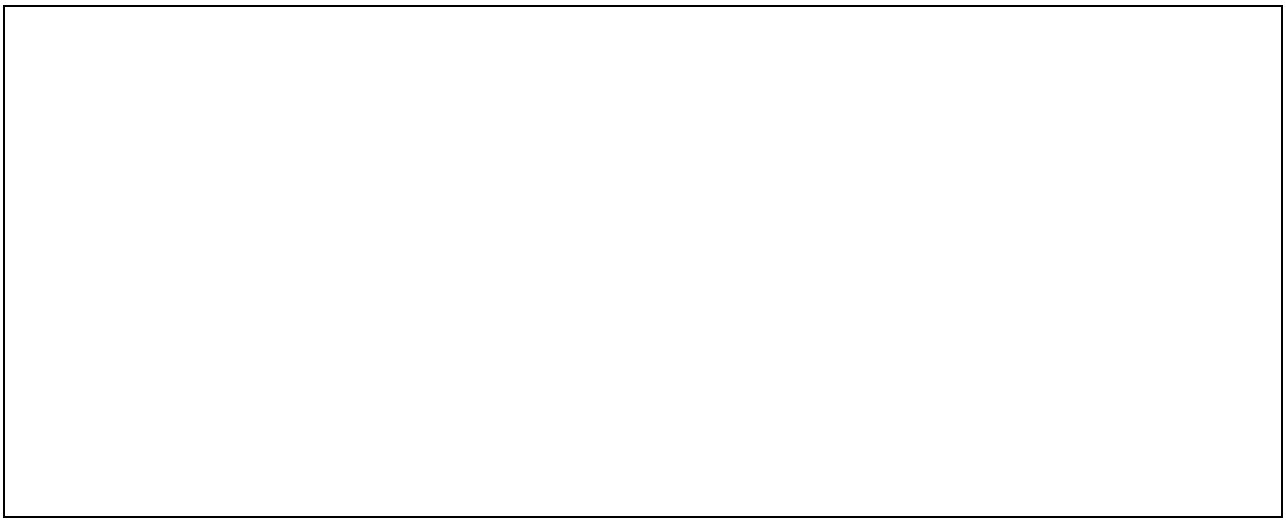
- En este script será necesario ejecutar una sentencia condicionalmente. Para obtener ayuda sobre las estructuras condicionales, ejecuta el comando:

  ```
  > Get-Help about_If
  ```

- Para determinar si un número es par, se puede utilizar el operador aritmético *módulo o resto de*. Para obtener ayuda sobre los operadores aritméticos, ejecuta el comando:

```
> Get-Help about_Arithmetic_Operators
```

- La sentencia *Do-While* deberá ejecutarse hasta que se hayan procesado todos los elementos del array, pero, ¿cómo puede detectarse dicha condición? Cuando se intenta acceder a un elemento de un *array* con un índice superior al número de elementos del mismo, PowerSell devuelve el valor nulo. Así pues, para detectar cuándo un índice incrementado llega al final de un *array*, basta con comparar con el valor nulo, que se obtiene de la variable *$null* predefinida en el entorno de PowerShell.



- **Formateo extendido de cadenas**

  Se trata de un mecanismo de composición de cadenas de caracteres. La composición se realiza mediante el operador –f. El esquema de uso de este operador es el siguiente:

  'Cadena_Patron_de_Formateo' –f "Valor1", "Valor2", "Valor3", …

  A la izquierda del operador se proporciona una cadena que actúa como patrón de formateo. Esta cadena incluye campos indicados mediante llaves y números, o sea, {0}, {1}, {2}, etc. Estos campos se sustituyen por los valores proporcionados a la derecha del operador.

  El formateo extendido se puede utilizar para generar cadenas que son enviadas a la consola, o bien a ficheros de texto. A continuación se plantea un conjunto de ejemplos orientados a comprender el funcionamiento del formateo extendido.

  - Ejecuta el siguiente comando

    ```
    > Write-Host ('{0} ... {1}' –f "Hola0", "Hola1")
    ```

  - Observa que el primer valor proporcionado a la derecha del operador –f se sustituye en la cadena patrón por el campo {0} y el segundo, por el campo {1}.
  - Ejecuta el siguiente comando

    ```
    > Write-Host ('{1} ... {0}' –f "Hola0", "Hola1")
    ```

  - Observa que el primer valor proporcionado a la derecha de –f se sustituye por el campo {0}, independientemente de la posición que ocupe este campo en la cadena patrón. Asimismo, el segundo valor proporcionado se sustituye por el campo {1}, y así sucesivamente.

    Los valores a la derecha del operador –f pueden ser variables de cualquier tipo. Su contenido se interpreta como cadena de caracteres y se integra en la cadena patrón donde corresponda.
  - Ejecuta los siguientes comandos

    ```
    > $a = 123; $b = "Hola"; $c = 5.5
    > Write-Host ('$a = {0}; $b = {1}; $c = {2}' –f $a, $b, $c)
    ```

    Los campos de la cadena patrón tienen la capacidad de establecer anchos de columna y alineaciones a la derecha y a la izquierda. Esto es de gran utilidad para realizar impresiones en forma de tabla. Los anchos de columna se indican a la derecha del número del campo

correspondiente, utilizando una coma (,) como separador. Si el ancho de columna va precedido de un signo menos (-), el valor se imprime alineado a la izquierda de la columna. En el caso de no especificar el signo menos, la alineación se realiza a la derecha.

- A continuación se plantea la escritura de las variables $a, $b y $c (definidas en los comandos anteriores), utilizando campos, anchos de columna de 20 caracteres, y justificación a la izquierda en cada columna.

```
> Write-Host ('$a = {0, -20}; $b = {1, -20}; $c = {2, -20}' -f $a, $b, $c)
```

- Tras la ejecución de este comando, observa la diferencia en el formateo generado, en relación con el comando en el que no se han utilizado anchos de columna.

(3) EJERCICIO. Con objeto de probar más detenidamente el formateo extendido de cadenas, debes realizar un script que imprima en la consola una tabla con todos los procesos en ejecución en el sistema. La tabla tendrá tres columnas. En la columna de la izquierda se imprimirá el nombre de cada proceso, alineado a la izquierda y con un ancho de 25. En la columna central se imprimirá la hora de arranque de cada proceso, alineada a la izquierda y con un ancho también de 25. Finalmente, en la columna de la derecha se escribirá la prioridad base de cada proceso, alineada a la derecha y con un ancho de 10. A continuación se indica, a modo de ejemplo, las primeras líneas de salida generadas por el script, ejecutado en un computador determinado.

```
Nombre                    Hora de arranque          Prioridad
======                    ================          =========
AggregatorHost                                             8
ApplicationFrameHost      02/02/2022 11:59:35              8
csrss                                                     13
csrss                                                     13
ctfmon                    02/02/2022 11:57:57             13
dllhost                   02/02/2022 13:27:42              8
...                       ...                            ...
```

Este script se realizará con la herramienta PowerShell ISE y se almacenará en la carpeta C:*Scripts* con el nombre *Prueba05.ps1*.

- **El bucle ForEach**

  Es una construcción iterativa de gran utilidad. Se usa para recorrer todos los elementos de una colección de objetos, de modo que se pueda llevar a cabo un determinado procesamiento con cada objeto de la colección. La sintaxis del bucle *ForEach* es la siguiente:

  `foreach ($<item> in $<collection>){statemet list>}`

  El número de iteraciones de este bucle es igual al número de elementos en *$<collection>*. En cada iteración, la variable *$<item>* toma el valor de un elemento de la colección, empezando por el elemento de índice [0], hasta alcanzar el último elemento.

  - Ejemplo:

```
$ArrayPrimos = 2, 3, 5, 7, 11, 13, 17, 19


foreach( $numero in $ArrayPrimos )
{
   Write-Host ( $numero )
}
```

  - Escribe el ejemplo anterior en PowerShell ISE y ejecútalo, comprobando su correcto funcionamiento. No es necesario que guardes este ejemplo en ningún fichero.

(4) EJERCICIO. Utilizando un bucle *ForEach*, escribe un script que liste los ficheros ubicados en la carpeta actual que tengan la extensión introducida en consola por el usuario.

  - A continuación se muestra la salida realizada por este script cuando el directorio actual es *C:\Scritps*.

```
Este script imprime los archivos del directorio actual con la extensión que indiques

Introduce la extension (debes incluir el punto): .ps1

Prueba01.ps1
Prueba02.ps1
Prueba03.ps1
Prueba04.ps1
Prueba05.ps1
```

  - Puedes probar el correcto funcionamiento del script en el directorio *C:\Windows\system32*, en el que hay múltiples ficheros con diversas extensiones (*.exe*, *.dll*, *etc*.)

    Este script se realizará con la herramienta PowerShell ISE y se almacenará en la carpeta C:*Scripts* con el nombre *Prueba06.ps1*. Escribe el contenido del script a continuación.

- **Ejercicio para la evaluación de scripts**

  Script *HandleInfo.ps1*. El objetivo de este ejercicio es construir un script que cree un fichero informativo sobre la utilización de *handles* por parte de los procesos en ejecución en el sistema. El nombre del fichero informativo será proporcionado por el usuario. El script creará el fichero en el directorio *C:\Temp*. No obstante, primero comprobará la existencia de dicho directorio y, en el caso de que no existiera, enviará un mensaje señalizando el problema y abortará la ejecución.

  Los *handles* son los identificadores utilizados por los procesos para hacer referencia a todo tipo de objetos utilizados en el sistema, como por ejemplo, carpetas, ficheros, ventanas, claves de registro, objetos de memoria, etc. Habitualmente, los procesos manejan un elevado número de *handles*.

  El script debe imprimir los procesos según el número de *handles* manejados en orden creciente. Asimismo, los procesos se organizarán en tres tramos. En el primer tramo se imprimen los procesos que manejen entre 0 y 100 *handles*. En el segundo tramo, los que manejen entre 101 y 1000, y en el tercer tramo, los que manejen más de 1000.

  En este fichero se escribirá una línea por cada proceso y, de cada uno de éstos, se indicará el ID, nombre y el número de *handles* manejado.

  Este script se realizará con la herramienta PowerShell ISE y se almacenará en la carpeta C:*Scripts* con el nombre *HandleInfo.ps1*.

  NOTAS:

  - Para determinar si existe el directorio *C:\Temp* se deberá obtener ayuda sobre el comando *Test-Path*. Si se quieren conocer los valores recibidos (de entrada) y generados (de salida) por un CmdLet se debe obtener ayuda del mismo utilizando el parámetro –Full.
  - El número de *handles* manejados por un proceso se almacena en la propiedad *HandleCount* de los objetos de tipo proceso.
  - Para escribir en el fichero la información correspondiente a cada proceso (ID, nombre y número de *handles*) será necesario componer primero toda esta información en una única cadena antes de ser escrita en el fichero. Para componer la cadena, debe utilizarse formateo extendido, usando anchos de columna apropiados. El ID y el nombre del proceso deben alinearse a la izquierda y el número de *handles* a la derecha.
  - Escribir una cabecera apropiada para la tabla a generar.
  - El procesamiento de los procesos debe realizarse utilizando bucles de tipo *foreach*.
  - A continuación se indica un ejemplo con el formato del fichero a generar.

```
######### Procesos entre 0 y 100 handles #########

Id      Nombre                     Número de handles
-------------------------------------------------------
104     Registry                                   0
0       Idle                                       0
932     fontdrvhost                               33
...     ...                                      ...


####### Procesos entre 101 y 1000 handles ########

Id      Nombre                     Número de handles
-------------------------------------------------------
1120    svchost                                  104
3028    SgrmBroker                               107
1144    svchost                                  110
```

```
...      ...                                          ...

######## Procesos con más de 1000 handles ########

Id       Nombre                         Número de handles
=================================================
6368     msedge                                    1071
1388     dwm                                       1088
4404     SystemSettings                            1101
...      ...                                        ...
```

- A continuación se proporciona una posible estructura del script perfilada mediante comentarios.

```
# ------------------ Descripción general del cometido del script -----------------
#
# El objetivo de este script es generar un fichero informativo que liste todos los
# procesos ordendos por el número de handles abiertos. El listado se organiza
# en tres tramos. En el primer tramo se imprimen los procesos entre 0 y 100 handles,
# en el segundo tramo, entre 101 y 1000 handles, y en el tercer tramo los
# que utilizan más de 1000.
# El nombre del fichero en el que se genera el informe es introducido por
# consola.
# El fichero generado se almacena en el directorio C:\Temp. Si este directorio no
# existe, se aborta la ejecución del script.
# -------------------------------------------------------------------------------

# Comprobación de la existencia del directorio C:\Temp. Si este directorio no
# existe se envía un mensaje a la consola indicado el problema y se aborta la
# ejecución del script









# Solicitar el nombre del fichero de salida y cargarlo en la variable $Fichero


# Generación de una línea en blanco en la consola


# Guardar en el array $Procesos todos los procesos en ejecución
# ordenados por su número de handles


# Creación del fichero informativo


######## Procesos entre 0 y 100 ########

# Escribir en el fichero el intervalo de procesos



# Escribir en el fichero una cabecera apropiada para la tabla de procesos







# Escribir en el fichero informativo los procesos cuyo Nº de handles se encuentre
# entre 0 y 100. Para ello, se utilizará un bucle foreach.
# Dentro del cuerpo del bucle se generará en la variable $Linea la información
# correspondiente a cada proceso. Se utilizará formateo extendido de cadenas
# para formatear cada línea apropiadamente.
```

```
######## Procesos entre 101 y 1000 ########

# Escribir en el fichero el intervalo de procesos


# Escribir en el fichero una cabecera apropiada para la tabla de procesos




# Escribir en el fichero informativo los procesos cuyo Nº de handles se encuentre
# entre 101 y 1000.









######## Procesos con más de 1000 handles ########

# Escribir en el fichero el intervalo de procesos


# Escribir en el fichero una cabecera apropiada para la tabla de procesos




# Escribir en el fichero informativo los procesos cuyo Nº de handles sea mayor de 1000
```