Trabajo Práctico Nro. 2 (GRUPAL):

- ***Tema:*** Programación de scripts básicos en PowerShell
- ***Descripción:*** Se programarán todos los scripts mencionados en el presente trabajo, teniendo especialmente en cuenta las recomendaciones sobre programación mencionadas en la introducción de este trabajo. Los contenidos de este trabajo práctico pueden ser incluidos como tema de parciales.
- ***Formato de entrega:*** Según el protocolo especificado anteriormente. Recomendamos realizar la entrega presencial
- ***Documentación:*** Todos los scripts que se entreguen deben tener un encabezado y un fin de archivo. Dentro del encabezado deben figurar el nombre del script, el trabajo práctico al que pertenece y el número de ejercicio dentro del trabajo práctico al que corresponde, el nombre de cada uno de los integrantes detallando nombre y apellido y el número de DNI de cada uno (tenga en cuenta que para pasar la nota final del trabajo práctico será usada dicha información, y no se le asignará la nota a ningún alumno que no figure en todos los archivos con todos sus datos), también deberá indicar el número de entrega a la que corresponde (entrega, primera reentrega, segunda reentrega, etc.).
- ***Evaluación:*** Luego de entregado el trabajo práctico los ayudantes procederán a evaluar los ejercicios resueltos, en caso de encontrar errores se documentará en la carátula del TP que será devuelta al grupo con la evaluación final del TP y una fecha de reentrega en caso de ser necesaria (en caso de no cumplir con dicha fecha de reentrega el trabajo práctico será desaprobado). Cada ayudante podrá determinar si un determinado grupo debe o no rendir coloquio sobre el trabajo práctico presentado.
  Las notas sobre los trabajos también estarán disponibles en el sitio de la cátedra (www.sisop.com.ar) donde además del estado de la corrección se podrá ver un detalle de las pruebas realizadas y los defectos encontrados.

  Importante: Cada ejercicio cuenta con una lista de validaciones mínimas que se realizará, esto no implica que se puedan hacer otras validaciones al momento de evaluar el trabajo presentado
- ***Fecha de entrega:*** **20/05/2020 o 21/05/2020**, dependiendo del día que se curse la materia.

Introducción:

La finalidad del presente práctico es que los alumnos adquieran un cierto entrenamiento sobre la programación de shell scripts en lenguaje PowerSell. Todos los scripts están orientados a la administración de una máquina, o red y pueden ser interrelacionados de tal manera de darles una funcionalidad real.
Cabe destacar que todos los scripts deben poder ser ejecutados en forma batch o interactiva, y que en ningún caso serán probados con el usuario administrator, por lo cual deben tener en cuenta al realizarlos, no intentar utilizar comandos, directorios, u otros recursos que solo están disponibles para dicho usuario, o usuarios del grupo. Todos los scripts deberán funcionar en las instalaciones del laboratorio 266, ya que es ahí donde serán controlados por el grupo docente.
Para un correcto y uniforme funcionamiento, se deberán respetar algunos lineamientos generales:

1. Modularidad:

Si bien es algo subjetivo del programador, se trata de privilegiar la utilización de funciones (internas / externas), para lograr una integración posterior menos trabajosa.

2. Claridad:

Se recomienda fuertemente el uso de comentarios que permitan la máxima legibilidad de los scripts.

3. Verificación:

Todos los scripts deben realizar un control de las opciones que se le indiquen por línea de comando, es decir, verificar la sintaxis de la misma. En caso de error u omisión de opciones, al estilo de la mayoría de los comandos deben indicar mensajes como:

```
"Error en llamada!
 Uso: comando [...]"
```

Donde se indicará entre corchetes [ ], los parámetros opcionales; y sin ellos los obligatorios, dando una breve explicación de cada uno de ellos.

Todos los scripts deben incluir una opción standard '-?' que indique el número de versión y las formas de llamada.

Ejercicios:

IMPORTANTE: la versión mínima a utilizar para confeccionar y evaluar este TP debe ser Powershell 6, ya que es la versión de Powershell Core donde se realizará la corrección

Ejercicio 1:

En base al siguiente código, responder:

```
Param (
    [Parameter(Position = 1, Mandatory = $false)]
    [String] $pathsalida = ".\procesos.txt ",
    [int] $cantidad = 3
)
$existe = Test-Path $pathsalida
if ($existe -eq $true) {
    $listaproceso = Get-Process
    foreach ($proceso in $listaproceso) {
        $proceso | Format-List -Property Id,Name >> $pathsalida
    }
    for ($i = 0; $i -lt $cantidad ; $i++) {
        Write-Host $listaproceso[$i].Name - $listaproceso[$i].Id
    }
} else {
    Write-Host "El path no existe"
}
```

Responder:

1. ¿Cuál es el objetivo del script?
2. ¿Agregaría alguna otra validación a los parámetros?
3. ¿Qué sucede si se ejecuta el script sin ningún parámetro?

Ejercicio 2:

Se cuenta con un archivo de log, donde se registraron todas las llamadas realizadas en una semana por un call center, donde se detalla el inicio y fin de las mismas. Estos registros tienen el siguiente formato, donde se indica quién realizó la llamada (usuario). Se debe considerar el primer registro como el inicio de la llamada y la siguiente como la finalización. Tener en cuenta que el proceso que registra las llamadas tiene problemas de sincronización, por lo que los registros no se encuentran ordenados.

Se pide obtener y mostrar por pantalla los siguientes datos uno debajo del otro:

- Promedio de tiempo de las llamadas realizadas por día.
- Promedio de tiempo y cantidad por usuario por día.
- Los 3 usuarios con más llamadas en la semana.
- Cuántas llamadas no superan la media de tiempo por día y el usuario que tiene más llamadas por debajo de la media en la semana.

Parámetros:

- `-Path`: Directorio en el que se encuentran los archivos de log. Puede ser una ruta relativa o absoluta. No se debe realizar búsqueda recursiva dentro del directorio.

Formato del registro de llamada:

- Fecha y Hora: Formato YYYY-MM-DD HH:mm:ss
- Usuario que realiza la llamada.
- Separador: " _ " (espacio, guión bajo, espacio)

Ejemplo de registros: (***(inicio)*** y ***(fin)*** son a modo de ejemplo, no son parte del archivo)

(inicio) 2020-03-09 14:22:00 _ aerodriguez
(inicio) 2020-03-09 14:22:10 _ fmarino
(inicio) 2020-03-09 14:22:11 _ vbo

(fin) 2020-03-09 14:25:00 _ fmarino
(fin) 2020-03-09 14:25:10 _ aerodriguez
(inicio) 2020-03-09 14:26:40 _ aerodriguez
(fin) 2020-03-09 14:26:41 _ vbo
(fin) 2020-03-09 14:32:00 _ aerodriguez

Criterios de corrección:

Control	Criticidad
Debe cumplir con el enunciado	Obligatorio
El script cuenta con una ayuda visible con Get-Help	Obligatorio
Validación correcta de los parámetros (Param)	Obligatorio
Proveer archivos de ejemplo para realizar pruebas	Obligatorio
Utilizar Format-List para formatear la salida por pantalla	Obligatorio

Ejercicio 3:

Además de problemas de sincronización, el sistema que maneja los logs del call center tiene problemas de almacenamiento. Esto se debe a que se crea un archivo de log por cada semana y empresa que contrata el servicio. Tratando de encontrarle una solución a este problema, Fernando, un administrador de sistemas, tuvo una idea: crear un script que se encargue de eliminar automáticamente los logs viejos cada vez que se crea un archivo nuevo.

Una vez creado un nuevo archivo, se deben evaluar todos los archivos del directorio y dejar solamente los de la última semana de cada empresa. Los archivos siguen la siguiente convención para los nombres: `nombreEmpresa-númeroSemana.log`. Por ejemplo, si en el directorio a evaluar, luego de creado el nuevo archivo existen los siguientes archivos, se deberían borrar los indicados:

```
unlam-11.log (archivo a eliminar)
unlam-12.log (archivo creado)
personal-11.log
movistar-11.log (archivo a eliminar)
movistar-12.log
```

Tener en cuenta que los archivos a eliminar deben cumplir con la convención de nombres mencionada anteriormente.

El script debe recibir los siguientes parámetros:

- `-Directorio`: Directorio en el que se encuentran los archivos de log. Puede ser una ruta relativa o absoluta. No se debe realizar búsqueda recursiva dentro del directorio.

Nota: Para controlar cuando se crea un archivo nuevo, utilizar `FileSystemWatcher` y el evento correspondiente a creación de archivos.

Criterios de corrección:

Control	Criticidad
Debe cumplir con el enunciado	Obligatorio
El script cuenta con una ayuda visible con Get-Help	Obligatorio
Validación correcta de los parámetros (Param)	Obligatorio
Proveer archivos de ejemplo para realizar pruebas	Obligatorio
Utilizar los eventos de FileSystemWatcher	Obligatorio
Usar expresiones regulares para validar que el nombre del archivo cumple con nombreEmpresa-númeroSemana.log	Obligatorio

Ejercicio 4:

Gonzalo, otro administrador de sistemas del call center decidió encarar el problema de almacenamiento de otra forma: comprimir los archivos de log antes de borrarlos, de esta manera no se pierden los archivos en caso de querer consultarlos más adelante y se ahorra espacio hasta que la gerencia tome la decisión de destinar presupuesto a la compra de nuevos discos rígidos. Para ahorrar trabajo, decidió tomar el script de Fernando y modificarlo.

A diferencia del script de Fernando, este script se tiene que ejecutar manualmente ya que no utiliza eventos para saber cuándo se creó un archivo nuevo. Tener en cuenta que los nuevos archivos se deben agregar al ZIP existente para no perder los ya procesados.

El script debe recibir los siguientes parámetros:

- `-Directorio`: Directorio en el que se encuentran los archivos de log. Puede ser una ruta relativa o absoluta. No se debe realizar búsqueda recursiva dentro del directorio.
- `-DirectorioZip`: Directorio en el que se generarán los archivos comprimidos de los clientes.
- `-Empresa`: Nombre de la empresa a procesar. (opcional)

Criterios de corrección:

Control	Criticidad
Debe cumplir con el enunciado	Obligatorio
El script cuenta con una ayuda visible con Get-Help	Obligatorio
Validación correcta de los parámetros (Param)	Obligatorio
Proveer archivos de ejemplo para realizar pruebas	Obligatorio
Utilizar el cmdlet Compress-Archive	Obligatorio

Ejercicio 5:

Por un requerimiento gubernamental, una universidad pública tiene la obligación de obtener estadísticas de aprobación y deserción de su alumnado. El sistema que mantiene el registro de las notas de los alumnos tiene la opción de exportarlas con el siguiente formato:

```
DNI|IdMateria|PrimerParcial|SegundoParcial|RecuParcial|RecuNota|Final
42736222|1|4|4|||
42736222|2|7|1|4|8
40586527|1||2|||
39873564|5|9|10|||
```

Para obtener dichas estadísticas, la universidad convoca a los alumnos de Sistemas Operativos a desarrollar un script que analice el archivo exportado y genere un archivo con los siguientes datos requeridos por el gobierno agrupados por materia:

- Cantidad de alumnos aptos para rendir final (sin final dado y notas en parciales/recuperatorio entre 4 y 6 inclusive).
- Cantidad de alumnos que recursarán (notas menor a 4 en final o en parciales y/o recuperatorio).
- Cantidad de alumnos con posibilidad de rendir recuperatorio (sin recuperatorio rendido y al menos una nota de parcial menor a 7).
- Cantidad de alumnos que abandonaron la materia (sin nota en al menos un parcial y sin recuperatorio rendido para dicho parcial).

El script debe recibir los siguientes parámetros:

- `-Nomina`: Ruta del archivo a procesar. Puede ser una ruta relativa o absoluta.

El formato del archivo de salida debe ser el siguiente:

```
"Materia","Final","Recursan","Recuperan","Abandonaron"
"1","1","2","0","0"
"2","6","3","7","0"
"3","5","6","0","10"
```

Criterios de corrección:

Control	Criticidad
Debe cumplir con el enunciado	Obligatorio
El script cuenta con una ayuda visible con Get-Help	Obligatorio
Validación correcta de los parámetros (Param)	Obligatorio
Proveer archivos de ejemplo para realizar pruebas	Obligatorio
Utilizar Import-Csv y Export-Csv para el manejo de los archivos	Obligatorio

Ejercicio 6:

Realizar un script que permita realizar la suma de todos los números fraccionarios contenidos en un archivo. Dichos números se encuentran separados por coma como se ve en el siguiente ejemplo:

```
16/8,8/9,1:5/8,9/2,-7/5
```

Tener en cuenta que:

- Los número fraccionarios pueden ser negativos.
- Los números pueden estar expresados en notación mixta, es decir, un número entero seguido de un número fraccionario. En el ejemplo anterior: `1:5/8`, equivale a `13/8`.

El resultado debe mostrarse por pantalla expresado como una fracción simplificada (el menor denominador posible que mantenga al numerador como número entero). Además, este resultado debe guardarse en un archivo ubicado en el mismo directorio del script que se llame `salida.out`. Si el archivo ya existe, sobrescribirlo.

Parámetros:

- `-Path`: Ruta del archivo de entrada ya sea de manera relativa o absoluta.

Nota: se asume que el formato del archivo es siempre correcto. Archivos vacíos o con un solo número fraccionario se consideran como archivos válidos.

Criterios de corrección:

Control	Criticidad
Debe cumplir con el enunciado	Obligatorio
El script cuenta con una ayuda visible con Get-Help	Obligatorio
Validación correcta de los parámetros (Param)	Obligatorio
Validación del archivo de entrada. Debe ser un archivo válido	Obligatorio
Proveer archivos de ejemplo para realizar pruebas	Obligatorio