



# CFGs ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS EN RED

## IMPLANTACIÓN DE SISTEMAS OPERATIVOS



## U4: Redireccionamiento, tuberías y filtros

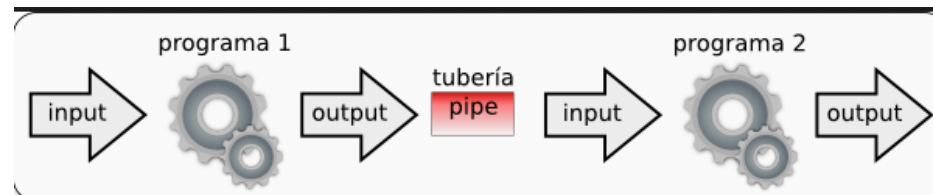
### Índice

- 1.- Introducción
- 2.- Redireccionamiento.
- 3.- Tuberías, formatear la salida y exportar.
- 4.- Filtros y ordenación.



# 1.- Introducción.

Básicamente, tanto las **redirecciones** como las **tuberías** tienen como objetivo obtener la salida de una acción y dirigirla a un lugar diferente del predeterminado (que normalmente es la pantalla)





## 2.- Redireccionamiento.

Las **redirecciones** están orientadas a **la obtención de un archivo de texto** con la salida que ofrece un *cmdlet*.

De esta forma, en lugar de consultar el resultado en ese momento, podremos guardarlo para revisarlo en el futuro.

Existen dos formas de realizar redirecciones:



## 2.- Redireccionamiento.

**Usando el carácter >:** Crea un nuevo archivo y deposita en él la salida del *cmdlet*. En caso de que el archivo exista previamente, sustituye su valor anterior por el nuevo.

**Usando los caracteres >>:** Añade al contenido del archivo la salida del *cmdlet*. En caso de que el archivo no exista previamente, se crea.

Ejemplo:

```
PS C:\Users\Windows> echo "hola" >prueba.txt
PS C:\Users\Windows> cat .\prueba.txt
hola
PS C:\Users\Windows> echo "adios" >>prueba.txt
PS C:\Users\Windows> cat .\prueba.txt
hola
adios
PS C:\Users\Windows> echo "Hasta luego" >prueba.txt
```



## 2.- Redireccionamiento.

### Diferencias entre: Write-Host y Write-output

Write-Host → Muestra en pantalla la información, pero NO redirecciona la salida.

Write-Output (**echo**) → Muestra en pantalla la información, pero Sí redirecciona la salida.

Windows PowerShell

```
PS D:\> Write-Host "Hola"
Hola
PS D:\> Write-Host "Hola" >prueba.txt
Hola
PS D:\> cat .\prueba.txt
PS D:\> Write-Output "Hola"
Hola
PS D:\> Write-Output "Hola" >prueba.txt
PS D:\> cat .\prueba.txt
Hola
PS D:\>
```



### 3.- Tuberías.

El objetivo de las **tuberías** consiste **en conectar la salida de un *cmdlet* con la entrada de otro**, que la tratará como su información de inicio.

Para enlazar los comandos utilizaremos el símbolo | (**tubería o pipe**).

#### Ejemplos:

Get-Command|Measure-Object → Cuenta los valores que recibe de Get-Command

Get-LocalUser|Format-List → Muestra en forma de lista los valores que recibe de Get-LocalUser.



### 3.- Tuberías.

**Measure-Object:** Calcula las propiedades numéricas de los objetos (mínimo, máximo, suma y el promedio). En el caso de los objetos de texto, puede contar y calcular el número de líneas, palabras y caracteres.

**Ejemplo:** Cuenta los objetos que hay dentro de la carpeta apuntes

```
PS D:\> ls d:\apuntes -Recurse | Measure-Object
```

```
Count      : 20474
Average    :
Sum        :
Maximum    :
Minimum    :
Property   :
```

**Ejemplo:** Cuenta los objetos de tipo archivo que hay dentro de la carpeta apuntes

```
PS D:\> ls d:\apuntes -file -Recurse | Measure-Object
```

```
Count      : 18043
Average    :
Sum        :
Maximum    :
Minimum    :
Property   :
```





### 3.- Tuberías.

#### Measure-Object:

**Ejemplo:** Calcula cuantos objetos de **tipo archivo** hay dentro de la carpeta apuntes, así como la suma, mínimo, máximo, Media del **tamaño de los archivos**.

```
PS D:\> ls D:\Apuntes\ -Recurse -File | measure-object -property length  
-sum -minimum -maximum -average
```

```
Count      : 18043  
Average    : 3377023,28670398  
Sum        : 60931631162  
Maximum    : 3417005568  
Minimum    : 0  
Property   : Length
```



### 3.- Tuberías: Formatear la salida

Podemos utilizar las tuberías para formatear la salida:

**Format-Table (ft)**

**Format-List (fl)**


**Format-Wide (fw)**



### 3.- Tuberías: Formatear la salida

#### Usar Format-Table para la salida tabular

Si usa el cmdlet **Format-Table** sin ningún nombre de propiedad especificado para formatear la salida del comando **Get-Process**, obtendrá exactamente la misma salida que al aplicar cualquier formato. El motivo es que los procesos se suelen mostrar en formato tabular, como la mayoría de los objetos de Windows PowerShell.

 Copiar

```
PS> Get-Process -Name powershell | Format-Table
```

Handles	NPM(K)	PM(K)	WS(K)	VM(M)	CPU(s)	Id	ProcessName
1488	9	31568	29460	152	3.53	2760	powershell
332	9	23140	632	141	1.06	3448	powershell

Si especifica el parámetro **AutoSize** al ejecutar el comando **Format-Table**, Windows PowerShell calculará los anchos de columna en función de los datos reales que va a mostrar.

#### Nota:

Ajusta el tamaño de las columnas, y corta la columna que no puede mostrar.

```
PS> Get-Process -Name powershell | Format-Table -Property Path,Name,Id,Company -
AutoSize
```

Path	Name	Id	Company
C:\Program Files\Windows PowerShell\v1.0\powershell.exe	powershell	2836	Micr...



### 3.- Tuberías: Formatear la salida

Si al parámetro **AutoSize** le añadimos la opción **-wrap**, ajustará las columnas, pero no truncará los datos.

```
PS C:\WINDOWS\system32> Get-Process -Name powershell | ft
Handles  NPM(K)  PM(K)  WS(K)  CPU(s)  Id
-----  -
      841      31  69736  85116    2,03  920

PS C:\WINDOWS\system32> Get-Process -Name powershell | ft -autosize
Handles NPM(K) PM(K) WS(K) CPU(s)  Id SI Process
Name
-----
      874      31 69736 85116   2,03 15920 38 powe...

PS C:\WINDOWS\system32> Get-Process -Name powershell | ft -autosize -Wrap
Handles NPM(K) PM(K) WS(K) CPU(s)  Id SI Process
Name
-----
      912      31 69736 85116   2,05 15920 38 powershell
```



### 3.- Tuberías: Formatear la salida

El cmdlet **Format-List** muestra un objeto en forma de una lista, donde cada propiedad aparece etiquetada y en una línea diferente:

```
PS> Get-Process -Name powershell | Format-List
```

```
Id      : 2760  
Handles : 1242  
CPU     : 3.03125  
Name    : powershell
```

```
Id      : 3448  
Handles : 328  
CPU     : 1.0625  
Name    : powershell
```

```
PS C:\WINDOWS\system32> Get-Process -name powershell | fl *
```

Puede especificar tantas propiedades como desee:

```
PS> Get-Process -Name powershell | Format-List -Property ProcessName,FileVersion,  
,StartTime,Id
```

```
ProcessName : powershell  
FileVersion : 1.0.9567.1  
StartTime   : 2006-05-24 13:42:00  
Id          : 2760
```



### 3.- Tuberías: Formatear la salida

#### Usar Format-Wide para la salida de un solo elemento

De forma predeterminada, el cmdlet **Format-Wide** muestra solo la propiedad predeterminada de un objeto. La información asociada a cada objeto se muestra en una sola columna:

```
PS C:\WINDOWS\system32> get-command -verb format
```

CommandType	Name
Function	Format-Hex
Function	Format-Volume
Cmdlet	Format-Custom
Cmdlet	Format-List
Cmdlet	Format-SecureBootUEFI
Cmdlet	Format-Table
Cmdlet	Format-Wide

También puede especificar una propiedad no predeterminada:

PowerShell

```
Get-Command -Verb Format | Format-Wide -Property Noun
```

output

Custom

List

Wide

Hex

Table

```
PS C:\WINDOWS\system32> get-command -verb format|format-wide_
```

Format-Hex	Format-Volume
Format-Custom	Format-List
Format-SecureBootUEFI	Format-Table
Format-Wide	



### 3.- Tuberías: Formatear la salida

**Out-host -paging (oh) :** Nos muestra la información paginada.

(El antiguo more de toda la vida)

#### **Ejemplo:**

.- Muestra el listado de todos los alias de forma paginada.

`Get-Alias|out-host -paging`



### 3.- Tuberías: Exportar la salida.

#### ConvertTo-HTML

La salida se genera en HTML

Se puede reenviar a un archivo, utilizando |out-file “NombreArchivo.html”

#### Ejemplo:

La lista de procesos la exportamos en un archivo html

```
PS C:\Users\Windows> Get-Process | ConvertTo-Html | Out-File "procesos.html"
```

Podemos visualizarlo:

```
PS C:\Users\Windows> Invoke-Item .\procesos.html
```

```
PS C:\Users\Windows> .\procesos.html
```





### 3.- Tuberías: Exportar la salida.

#### Export-CSV

La salida del comando se exporta a un archivo de texto separado por comas (CSV).

Ejemplo:

```
Get-Process | Export-Csv Procesos.csv  
cat .\Procesos.csv
```

#### Export-CliCml

La salida del comando se exporta a un archivo xml

Ejemplo:

```
PS D:\> Get-LocalUser | Export-Clixml usuarios.xml
```

```
.\usuarios.xml
```



## 4.- Filtros y ordenación

Los filtros nos permitir seleccionar los objetos que cumplen una determinada condición.

La ordenación nos servirá para tener los objetos clasificados.

Haremos uso de las tuberías para filtrar y ordenar.



## 4.- Filtros

**Out-Gridview** : Muestra la información en modo rejilla y nos permite filtrar la información.

### Ejercicio:

- 1.- Crea en tu directorio de trabajo tres archivos: uno.jpg, dos.jpg y tres.bmp.
- 2.- Muestra el contenido de tu directorio (**Get-ChildItem**) de trabajo y aplícale la tubería **Out-Gridview**. → **ls|out-Gridview**
- 3.- Aplícale un filtro para quedarte con los archivos terminados en jpg.

¿ Le ves alguna utilidad ?





## 4.- Filtros

### Out-GridView y ConvertTo-HTML

**Ejercicio:** Realiza los ejercicios que se plantean.

<https://www.youtube.com/watch?v=NyBsKLEWClg>



## 4.- Filtros

**Where-Object** (Alias → ?) : Nos permite seleccionar la información que cumple una condición.

Sintaxis:

cmdlet | **where-Object** {**\$\_.propiedad –comparador “expresión”**}

**\$\_. Propiedad** → Para referirnos a una propiedad del objeto.

**\$\_** → Para referirnos a objeto de la canalización



## 4.- Filtros

### Where-Object:

cmdlet | **where-Object** {\$\_propiedad –comparador “expresión”}

Operador de comparación	Significado	Ejemplo (devuelve el valor True)
-eq	Es igual a	1 -eq 1
-ne	Es distinto de	1 -ne 2
-lt	Es menor que	1 -lt 2
-le	Es menor o igual que	1 -le 2
-gt	Es mayor que	2 -gt 1
-ge	Es mayor o igual que	2 -ge 1
-like	Es como (comparación de caracteres comodín para texto)	"file.doc" -like "f*.do?"
-notlike	No es como (comparación de caracteres comodín para texto)	"file.doc" -notlike "p*.doc"
-contains	Contiene	1,2,3 -contains 1
-notcontains	No contiene	1,2,3 -notcontains 4



## 4.- Filtros

### Where-Object:

cmdlet | **where-Object** {\$\_propiedad –comparador “expresión”}

Operador lógico	Significado	Ejemplo (devuelve el valor True)
-and	And lógico; devuelve True si ambos lados son True.	(1 -eq 1) -and (2 -eq 2)
-or	Or lógico; devuelve True si cualquiera de los lados es True.	(1 -eq 1) -or (1 -eq 2)
-not	Not lógico; invierte True y False.	-not (1 -eq 2)
!	Not lógico; invierte True y False.	!(1 -eq 2)



## 4.- Filtros

### Ejercicio : Where-Object

**Como administrador nos puede interesar aquellos puertos en los que se ha producido conexión.**

```
PS C:\Users\manolo> Get-NetTCPConnection | ?{$_.State -eq "Established"} | ft LocalAddress, LocalPort, State
```

LocalAddress	LocalPort	State
-----	-----	-----
192.168.2.231	55750	Established
192.168.2.231	55239	Established
192.168.2.231	54421	Established
192.168.2.231	54420	Established
192.168.2.231	54416	Established
192.168.2.231	54347	Established
127.0.0.1	54269	Established
192.168.2.231	54238	Established
192.168.2.231	54237	Established
127.0.0.1	50041	Established





## 4.- Filtros

### Ejercicio : Where-Object

**Averiguar cuántos archivos mayores de 100MB tenemos.**

- ▶ `Get-ChildItem -Recurse |Where-Object {$_.Length -gt 100MB}`
- ▶ Where-Object → actúa como filtro de búsqueda
- ▶ `$_` → Representa el objeto recibido
- ▶ `.Length` → Representa la propiedad tamaño del objeto



## 4.- Filtros

### Ejercicio : Where-Object

**Averiguar cuántos archivos mayores de 100MB tenemos. Mostrar en forma de tabla, el nombre, el tamaño y el directorio.**

```
PS C:\Users\Windows> ls -Recurse | Where-Object {$_.Length -gt 100MB} | ft  
Name,Length,Directory
```



## 4.- Filtros: Ordenación

**Sort-Object** → Ordena los objetos que recibe por la columna o propiedad indicada.

- Descending → Los ordena de mayor a menor
- Property Length → Indicamos la propiedad.

Ejemplos:

Get-Command |Sort-Object -descending

```
PS C:\Users\manolo> ls |Sort-Object -Property LastWriteTime -Descending
```



## 4.- Filtros: Ordenación

**Ejercicio** : Where-Object y sort-Object

**Averiguar cuántos archivos mayores de 100MB tenemos. Mostrar la información ordenada por Tamaño (Length). Solo nos interesa Name y length.**

```
PS C:\Users\manolo> ls d:\ -Recurse | ?{$_.Length -gt 100Mb} | Sort-Object -Property Length -Descending | ft Name, Length
```

**NOTA IMPORTANTE:** Format-table debe ir al final.



## 4.- Filtros: Selección de objetos a mostrar.

**Select-object:** Podemos utilizarlo para: mostrar unas determinadas propiedades o indicar el número de objetos a mostrar.

-First Numero

-Last Numero

**Ejemplo: Muestra los 5 procesos que consume más CPU.**

```
PS D:\> Get-Process | Sort-Object -Property CPU -Descending | Select-Object
-First 5
```

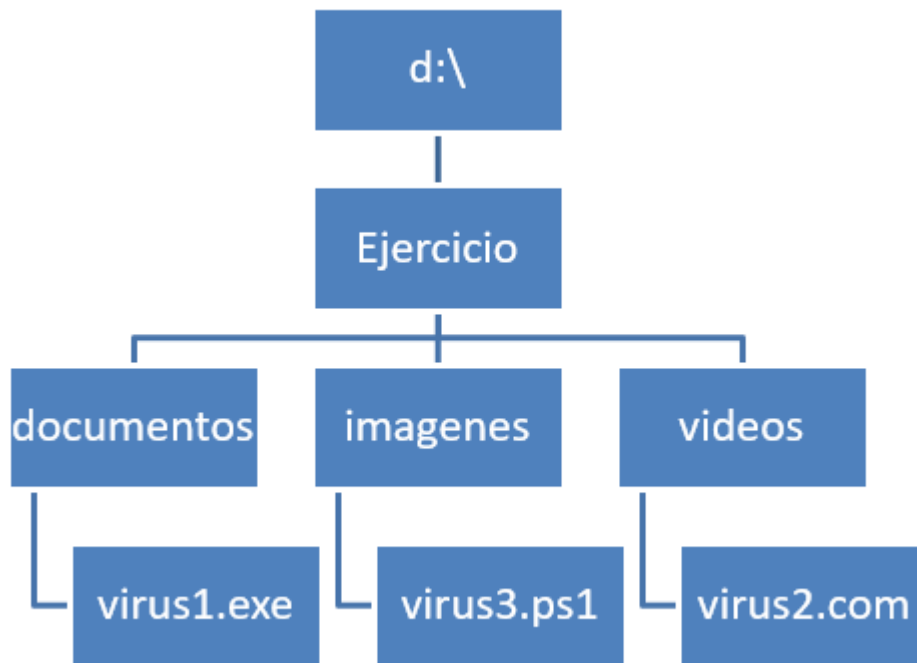
Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
-----	-----	-----	-----	-----	--	--	-----
287	39	200176	81480	16.923,44	17864	61	chrome
1902	76	125380	192680	235,39	18168	61	POWERPNT
2240	71	140560	207716	180,72	17852	61	chrome
874	64	743720	778804	146,13	3316	61	powershell
1596	44	461756	437804	89,11	1248	61	chrome



## 4.- Filtros: Para cada objeto

**ForEach-Object (%)**: Realiza una operación con cada uno de los objetos que recibe.

cmdlet | ForEach-Object { Bloque\_Codigo }



### Ejemplo:

Borra todos los archivos del tipo virus\*.

¿Te funciona?:

Rm -r d:\ejercicio\virus\*

Intenta:

```
PS D:\> ls .\ejercicio\virus* -r|% {rm $_.FullName}
```



## 4.- Filtros: Agrupación

### Group-Object

Nos muestra la información agrupada por alguna propiedad.

Muestra **el número, el nombre y el grupo.**

Ejemplos:

```
PS C:\Users\Windows> Get-Service |group-object Status
Count Name                               Group
-----
158 Stopped                               {AarSvc_1fca3d40, AJRouter, ALG, Ap...
116 Running                               {Appinfo, AppMgmt, AudioEndpointBui...

PS C:\Users\Windows> Get-Service |group-object Status |fl *

Values : {Stopped}
Count   : 158
Group   : {AarSvc_1fca3d40, AJRouter, ALG, AppIDSvc...}
Name    : Stopped

Values : {Running}
Count   : 116
Group   : {Appinfo, AppMgmt, AudioEndpointBuilder, Audiosrv...}
Name    : Running
```



## 4.- Filtros: Agrupación

### Group-Object

Ejercicio: Serías capaz decirme cuantos archivos hay por extensión.  
Quédate con las 10 que más archivos tienen.

```
PS D:\> ls -r apuntes | Group-Object -Property Extension | Sort-Object -Property Count -Descending | Select-Object -First 10
```





## Sugerencias/mejoras del tema



### Sugerencias /mejoras del tema