3. Gestión del hardware en PowerShell (nivel intermedio) (utilizando llamadas CIM)

Introducción

El hardware es el conjunto de componentes físicos que constituyen un ordenador y que mediante conexiones con dispositivos auxiliares permiten realizar las funciones de procesamiento, almacenamiento y transferencia de datos.

El procesamiento consiste en recibir datos, realizar cálculos y operaciones con los mismos, el almacenamiento consiste en almacenar datos durante un periodo de tiempo y la transferencia consiste en enviar los resultados procesados al exterior.

El hardware son los elementos físicos que se pueden tocar. Sirven para realizar tareas que se resuelven mediante el software, algunos elementos hardware son: la caja donde está la placa base, el procesador, la memoria, el disco duro, etc. También se consideran hardware los dispositivos de entrada y salida como la pantalla, el teclado, el ratón, etc.

Una de las funciones del sistema operativo es controlar y gestionar el uso del hardware del ordenador: procesador, dispositivos de E/S, memoria principal, tarjetas gráficas y el resto de periféricos.

Veamos los principales componentes hardware que tiene un equipo.

Procesador

También conocido como CPU (Central Processing Unit, unidad central de procesamiento), es el cerebro del ordenador. Su función es leer instrucciones y ejecutarlas, estas instrucciones configuran un conjunto de datos codificados en binario que se almacena en la memoria. Todos los programas se componen de instrucciones, cada instrucción se ejecuta mediante un ciclo básico de ejecución que es el período que tarda el procesador en ejecutar una instrucción.

La CPU funciona del siguiente modo: obtiene la obtiene la primera instrucción de la memoria, la decodifica para determinar el código de operación y los datos, después la ejecuta y en algunos casos almacena el resultado. Este proceso se ejecuta continuamente hasta que se leen todas las instrucciones del programa.

Un concepto importante al hablar de CPU y que nos permite comprender algunas de sus funciones es la arquitectura.

Una arquitectura indica qué hace un ordenador y define el conjunto de instrucciones (serie de instrucciones que hacen referencia al conjunto básico de comandos e instrucciones que un microprocesador comprende y puede llevar a cabo) y cómo se codifican, los datos que maneja, los registros internos, etc. Cada arquitectura es distinta aunque puede haber características similares entre arquitecturas.

Para una arquitectura puede haber distintas organizaciones. La organización define cómo realiza las funciones un ordenador.

Por último, la realización del ordenador se encarga de implantar físicamente una organización, un ejemplo de realización es la frecuencia básica del procesador que describe la velocidad a que los transistores de este se abren y cierran. La frecuencia básica del procesador es el punto de operación donde se define la TDP (potencia de diseño térmico).

La frecuencia se mide en gigahertz (GHz) o mil millones de ciclos por segundo.

Otro tipo de realización es la frecuencia turbo máxima indica la frecuencia de un solo núcleo a que el procesador puede operar. La frecuencia se mide en gigahertz (GHz) o mil millones de ciclos por segundo.

Un ejemplo de lo explicado anteriormente pueden ser son las implementaciones de x86-64 (x86-64 está basada en la extensión del conjunto de instrucciones x86 para manejar direcciones de 64 bits): AMD64 e Intel 64 en el caso de Intel 64 (la arquitectura Intel 64 mejora el desempeño permitiendo que los sistemas direccionen más de 4 GB de memoria física y virtual) tiene las organizaciones Core 2 Duo, Core 2 Quad, y Core 2 Extreme, etc. La realización de un Core 2 Quad puede ser de 3,40 GHz, 3,80 GHz, etc.

Los procesadores contienen núcleos, un núcleo describe el número de unidades de procesamiento independientes en un componente computacional individual (matriz o chip).

Un concepto importante es el hilo, o hilo de ejecución, es un término de software para la secuencia de instrucciones de orden básico que puede pasar por o procesarse en un núcleo de CPU individual.

Los procesadores actuales poseen tecnologías avanzadas que permiten realizar tareas como la virtualización, Hyper-Threading, etc. Un ejemplo son los procesadores Intel que tienen las siguientes tecnologías avanzadas:

- Tecnología Intel Turbo Boost
- Tecnología Hyper-Threading Intel
- Tecnología de virtualización Intel® (VT-x)
- Tecnología de virtualización Intel® para E/S dirigida (VT-d)
- Intel VT-x con tablas de páginas extendidas (EPT)
- Intel Transactional Synchronization Extensions

- Tecnología Intel SpeedStep
- Tecnologías de monitoreo térmico
- Acceso a memoria rápida Intel
- Intel Flex Memory Access
- Tecnología Intel Identity Protection
- Programa Intel de imagen estable para plataformas (SIPP)
- Tecnología Intel de respuesta inteligente

Con PowerShell se puede obtener información de los procesadores mediante la siguiente llamada CIM

```
Get-CimInstance Win32_Processor
```

Ejemplos

Información sobre la arquitectura

```
(Get-CimInstance Win32_Processor).caption
(Get-CimInstance Win32_ComputerSystem).SystemType
```

Información sobre la organización

```
(Get-CimInstance Win32_Processor).name
```

Información sobre la realización

```
((Get-CimInstance Win32_Processor).name).split("@")[1]
```

Información sobre los procesadores

```
Get-CimInstance -Class Win32_Processor | Select -Property Name, Number*
```

Número de núcleos

```
Get-CimInstance -Class Win32_Processor | Select-Object NumberOfCores
Get-CimInstance -query "select NumberOfCores from Win32_Processor"
```

Carga del procesador

```
Get-CimInstance win32_processor | Select-Object LoadPercentage
```

Ejercicios

Información sobre el procesador

```
Get-CimInstance -class "Win32_Processor"| % {
    Write-Host "CPU ID: "
    Write-Host $_.DeviceID
    Write-Host "CPU Model: "
    Write-Host $_.Name
    Write-Host "CPU Cores: "
    Write-Host $_.NumberOfCores
    Write-Host "CPU Max Speed: "
    Write-Host $_.MaxClockSpeed
    Write-Host "CPU Status: "
    Write-Host $_.Status
    Write-Host
}
```

Saber si el procesador es Intel

```
if((Get-CimInstance  Win32_Processor).Caption  -match "Intel"){"Intel"}else{"No es Intel"}
```

Memoria caché

La memoria caché es un área de almacenamiento dedicada a los datos usados o solicitados con más frecuencia para su recuperación a gran velocidad.

La caché es una memoria que se sitúa entre la unidad central de procesamiento (CPU) y la memoria de acceso aleatorio (RAM) para acelerar el intercambio de datos.

Cmdlets con llamadas CIM que muestran información sobre la memoria caché

```
Get-CimInstance Win32_AssociatedProcessorMemory
Get-CimInstance Win32_CacheMemory
```

Ejemplo

Ver el tamaño de la memoria caché

```
Get-CimInstance Win32_CacheMemory | Select-Object DeviceID,MaxCacheSize
```

Memoria

Se encarga de almacenar los programas que se están ejecutando en el ordenador y los datos necesarios para la ejecución de dichos programas.

En teoría las memorias tiene que ser muy rápidas, de gran tamaño y con bajo precio, pero actualmente no existe ninguna tecnología que reúna estos requisitos, como solución a esta situación existe la jerarquía de memoria.

Dos principios sobre la memoria:

- Menor cantidad, acceso más rápido.
- Mayor cantidad, menor coste por byte.

La idea es que la memoria se organice en niveles, cuanto más cercanos al procesador, más pequeños, rápidos y caros. El objetivo de es conseguir un rendimiento de memoria a gran velocidad y de un tamaño igual al nivel más bajo de la jerarquía. A medida que bajamos en los niveles, la velocidad es menor pero el almacenamiento es mayor.

El primer nivel de la jerarquía es el de los registros que se están en el procesador, son muy rápidos pero limitados; el siguiente nivel es la caché que son zonas de gran velocidad y muy próximas a la CPU en donde se almacena la información que se utiliza con más frecuencia; el siguiente nivel es la memoria principal, a esta memoria se la conoce como RAM (Random Access Memory, memoria de acceso aleatorio) y tiene las siguientes características:

- Es una memoria de acceso aleatorio; se accede directamente a una determinada posición de la memoria sin pasar por las anteriores.
- Es una memoria volátil, mantiene los datos hasta que se corta la alimentación.
- Es una memoria de lectura/escritura, se pueden leer los datos que tiene almacenados y escribir en ella nuevos datos o resultados.

Algunos tipos de memoria RAM son DRAM (Dynamic Random Access Memory, memoria de acceso aleatorio dinámica), SRAM (Static Random Access Memory, memoria estática de acceso aleatorio), SDRAM (Synchronous Dynamic Random Access Memory, memoria de acceso aleatorio dinámica síncrona) y DDR SDRAM (Double Data Rate, doble tasa de transferencia de datos).

Cmdlets con llamadas CIM que muestran información sobre la memoria

```
Get-CimInstance Win32_MemoryArray
Get-CimInstance Win32_MemoryArrayLocation
Get-CimInstance Win32_MemoryDevice
Get-CimInstance Win32_MemoryDeviceArray
Get-CimInstance Win32_MemoryDeviceLocation
Get-CimInstance Win32_PhysicalMemory
Get-CimInstance Win32_PhysicalMemoryArray
Get-CimInstance Win32_PhysicalMemoryLocation
Get-CimInstance Win32_SMBIOSMemory
```

Ejemplos

Lugar físico en donde se encuentran las memorias

```
Get-CimInstance Win32_PhysicalMemory | Select-Object BankLabel
```

Velocidad de reloj del dispositivo de memoria en megahercios

```
Get-CimInstance Win32_PhysicalMemory | Select-Object ConfiguredClockSpeed
```

Voltaje configurado para los dispositivos de memoria en milivoltios

```
Get-CimInstance Win32_PhysicalMemory | Select-Object ConfiguredVoltage
```

Capacidad de la memoria en bytes

```
Get-CimInstance Win32_PhysicalMemory | ForEach-Object {$_.capacity / 1GB}
```

Tipo de memoria

```
Get-CimInstance Win32_PhysicalMemory | Select-Object MemoryType
```

Tipo de implementación del chip de los dispositivos de memoria

```
Get-CimInstance Win32_PhysicalMemory | Select-Object FormFactor
```

Máxima y mínima tensión soportada por los dispositivos de memoria

```
Get-CimInstance Win32_PhysicalMemory | Select-Object MaxVoltage,MinVoltage
```

Velocidad de la memoria en nanosegundos

```
Get-CimInstance Win32_PhysicalMemory | Select-Object Speed
```

Ejercicio

Saber si el fabricante de la memoria es Elpida o no

```
if((Get-CimInstance Win32_PhysicalMemory).Manufacturer -match "Elpida"){"Elpida"}else{"No es Elpida"}
```

Acceso directo a memoria

Acceso directo a memoria (DMA, Direct Memory Access, Acceso directo) es un chip se encarga de la transferencia y accede a la memoria para leer o escribir datos que recibe y envía el dispositivo sin pasar por el procesador.

Cmdlets con llamadas CIM que muestran información sobre DMA

```
Get-CimInstance Win32_DMAChannel
Get-CimInstance Win32_DeviceMemoryAddress
Get-CimInstance Win32_SystemMemoryResource
```

Discos duros

El último nivel en la jerarquía de memoria son los discos duros. Son dispositivos de almacenamiento no volátil, es decir, no se pierde la información cuando se desconecta la energía. La capacidad de almacenamiento de los discos duros es muy superior a la RAM, siendo además de menor precio; sin embargo, el problema está en que es lento acceder a la información, esto se debe a que disco es un dispositivo mecánico y tiene que moverse hasta llegar a la información.

Un disco duro consiste en uno o varios platos que están girando, mientras giran hay un componente dentro del disco que llamado brazo mecánico, que en el extremo tiene dos cabezas que leen y escriben sobre la cada una de las superficies del plato (caras), dentro de un disco hay varios platos.

Cmdlet que muestra información sobre los discos duros

```
Get-Disk
```

Parámetros y alias de los parámetros para el cmdlet Get-Disk

```
UniqueId            {Id}
FriendlyName        {}
SerialNumber        {}
Path                {}
Number              {DeviceId}
Partition           {}
VirtualDisk         {}
iSCSISession        {}
iSCSIConnection     {}
StorageSubSystem    {}
StorageNode         {}
StorageJob          {}
CimSession          {Session}
ThrottleLimit       {}
AsJob               {}
Verbose             {vb}
Debug               {db}
ErrorAction         {ea}
WarningAction       {wa}
InformationAction   {infa}
ErrorVariable       {ev}
WarningVariable     {wv}
InformationVariable {iv}
OutVariable         {ov}
OutBuffer           {ob}
PipelineVariable    {pv}
```

Cmdlet con llamada CIM que muestra información sobre los discos duros

```
Get-CimInstance Win32_DiskDrive
```

Ejemplo

Obtener información sobre los discos que hay conectados

```
Get-Disk
```

Buses

Hay un elemento muy importante que interviene en la comunicación entre procesador, memoria y dispositivos de entrada y salida denominado bus, éste se define como un conjunto de líneas por las que se transmite información entre distintos componentes hardware.

Un bus es un subsistema que transfiere datos entre los componentes de una computadora o entre computadoras.

Algunos de los buses que tienen los ordenadores actuales son el PCI (Peripheral Component Interconnect, interconexión de componentes periféricos), Express, ISA (Industry Standard Architecture, Arquitectura estándar de la industria), USB (Universal Serial Bus, Bus universal en serie), IEEE 1394 (también llamado FireWire), etc.

Cmdlets con llamadas CIM que muestran información sobre los buses PCI, ISA

```
Get-CimInstance Win32_AllocatedResource
Get-CimInstance Win32_Bus
Get-CimInstance Win32_DeviceBus
Get-CimInstance Win32_PortResource
```

Cmdlets con llamadas CIM que muestran información sobre los buses IDE

```
Get-CimInstance Win32_IDEController
Get-CimInstance Win32_IDEControllerDevice
```

Cmdlets con llamadas CIM que muestran información sobre los

buses SCSI

```
Get-CimInstance Win32_SCSIController
Get-CimInstance Win32_SCSIControllerDevice
```

Cmdlets con llamadas CIM que muestran información sobre los buses USB

```
Get-CimInstance Win32_ControllerHasHub
Get-CimInstance Win32_USBController
Get-CimInstance Win32_USBControllerDevice
Get-CimInstance Win32_USBHub
```

Cmdlets con llamadas CIM que muestran información sobre los buses IEEE 1394

```
Get-CimInstance Win32_1394Controller
Get-CimInstance Win32_1394ControllerDevice
```

Cmdlets con llamadas CIM que muestran información sobre otros buses y puertos

```
Get-CimInstance Win32_InfraredDevice
Get-CimInstance Win32_ParallelPort
Get-CimInstance Win32_PCMCIAController
Get-CimInstance Win32_SerialPort
Get-CimInstance Win32_SerialPortConfiguration
Get-CimInstance Win32_SerialPortSetting
```

Ejemplos

Buses conectados al sistema operativo

```
(Get-CimInstance Win32_Bus).DeviceID
```

Buses USB conectados al sistema

```
Get-CimInstance Win32_USBControllerDevice |%{($_.Dependent)} | Sort Manufacturer,Description,DeviceID | Ft -GroupBy Manufacturer Description,Service,DeviceID
```

Conexiones E/S

Las conexiones de la entrada y salida se realizan mediante interrupciones y otros tipos de conexiones.

Las IRQ son líneas que llegan al controlador de interrupciones, un componente de hardware dedicado a la gestión de las interrupciones, y que puede estar integrado en el procesador principal o ser un circuito separado conectado al mismo.

Cmdlet con llamada CIM que muestra información sobre interrupciones

Get-CimInstance Win32_IRQResource

Cmdlet que muestra información sobre conexiones de entrada y salida

Get-PnpDevice

Parámetros y alias de los parámetros para el cmdlet Get-PnpDevice

```
FriendlyName        {}
InstanceId          {DeviceId}
Class               {}
PresentOnly         {}
Status              {}
CimSession          {Session}
ThrottleLimit       {}
AsJob               {}
Verbose             {vb}
Debug               {db}
ErrorAction         {ea}
WarningAction       {wa}
InformationAction   {infa}
```

```
ErrorVariable       {ev}
WarningVariable     {wv}
InformationVariable {iv}
OutVariable         {ov}
OutBuffer           {ob}
PipelineVariable    {pv}
```

Cmdlets con llamadas CIM que muestran información sobre otros tipos de conexiones de entrada y salida

```
Get-CimInstance Win32_DeviceSettings
Get-CimInstance Win32_OnBoardDevice
Get-CimInstance Win32_PNPAllocatedResource
Get-CimInstance Win32_PNPDevice
Get-CimInstance Win32_PNPEntity
Get-CimInstance Win32_SystemDriverPNPEntity
```

Ejercicio

Detectar si hay un dispositivo USB conectado y copiar el contenido en una carpeta temporal

```
#Almacenar en un fichero los dispositivos que suelen conectarse
#(Get-PnpDevice | Where-Object {$_.FriendlyName -EQ 'Dispositivo de almacenamiento USB'}).InstanceId | Out-File usb.txt
#Almacenar en un fichero las letras de unidades que se suelen usar
(Get-Partition | Where-Object Type -EQ "Basic").DriveLetter | Out-File unidades.txt

#Ver los dispositivos conectados y si hay uno nuevo listar unidades de disco y realizar copia recursiva
gc .\usb.txt | %{
    #$_
     if((Get-PnpDevice | Where-Object {$_.FriendlyName -EQ 'Dispositivo de almacenamiento USB'}).InstanceId -contains $_)
    {
        #"USB conocido"
```

```
    }
    else
    {
        "USB extraño, copiar contenido"
        #Si la letra de la unidad es nueva copiamos contenido
        gc .\unidades.txt | %{
              if((Get-Partition | Where-Object Type -EQ
"Basic").DriveLetter -contains $_)
            {
                "Unidad $_ conocida"
            }
            else
            {
                 "Unidad $_ desconocida, copiar contenido en
c:\temp"
                Copy-Item $_ -Destination c:\temp\usb\ -Force
-Recurse -WhatIf
            }
        }
    }
}
```

Dispositivos de entrada y salida

La transferencia de datos se realiza mediante el sistema de entrada y salida, de esta forma es posible comunicarse con el exterior y así poder recibir datos y enviar los resultados.

Los dispositivos de E/S tienen dos partes: un dispositivo controlador y el dispositivo en sí. El dispositivo controlador es un chip o un conjunto de chips que controlan físicamente el dispositivo. La comunicación entre el dispositivo controlador y el sistema operativo se realiza mediante un software llamado driver. Los dispositivos de E/S se pueden dividir en dos: dispositivos de bloque y de carácter.

Los dispositivos de bloque almacenan información en bloque de

tamaño fijo, algunos ejemplos son los discos duros, CDs y memorias USB. Los dispositivos de carácter, como su propio nombre indica, envían información en forma de carácter, un ejemplo es el teclado.

Los dispositivos en sí pueden ser de entrada, salida o ambos. Se conocen como periféricos.

Entrada

Cmdlet con llamada CIM que muestra información sobre el teclado

```
Get-CimInstance Win32_Keyboard
```

Cmdlet con llamada CIM que muestra información sobre el ratón

```
Get-CimInstance Win32_PointingDevice
```

Salida

Cmdlets con llamadas CIM que muestran información sobre el monitor y vídeo

```
Get-CimInstance Win32_DesktopMonitor
Get-CimInstance Win32_DisplayControllerConfiguration
Get-CimInstance Win32_VideoController
Get-CimInstance Win32_VideoSettings
```

Cmdlet con llamada CIM que muestra información sobre el dispositivo de sonido

```
Get-CimInstance Win32_SoundDevice
```

Cmdlets con llamadas CIM que muestran información sobre las impresoras

```
Get-CimInstance Win32_DriverForDevice
Get-CimInstance Win32_Printer
Get-CimInstance Win32_PrinterConfiguration
Get-CimInstance Win32_PrinterController
Get-CimInstance Win32_PrinterDriver
```

```
Get-CimInstance Win32_PrinterDriverDll
Get-CimInstance Win32_PrinterSetting
Get-CimInstance Win32_PrintJob
Get-CimInstance Win32_TCPIPPrinterPort
```

Entrada y salida

Cmdlets con llamadas CIM que muestran información sobre el CD y DVD

```
Get-CimInstance Win32_CDROMDrive
Get-CimInstance Win32_PhysicalMedia
```

Cmdlets que muestra información sobre los adaptadores de red

```
Get-NetAdapterHardwareInfo
Get-NetAdapter -Physical
```

Cmdlets con llamadas CIM que muestran información sobre los adaptadores de red

```
Get-CimInstance Win32_NetworkAdapter
Get-CimInstance Win32_NetworkAdapterConfiguration
Get-CimInstance Win32_NetworkAdapterSetting
```

Ejemplos

Ver las impresoras que hay en el sistema

```
Get-CimInstance Win32_Printer
```

Ver los dispositivos plug and play

```
Get-CimInstance -query "select Name,Status from Win32_PnPEntity"
```

Ver el estado de los dispositivos plug and play

```
Get-CimInstance -Class Win32_Processor | Select -Property Name, Number*
```

Ver información sobre la webcam

```
Get-CimInstance -query "select * from Win32_PnPEntity where caption like '%cam%'"
Get-CimInstance Win32_PnPEntity | where {$_.caption -match 'webcam'}
```

Ejercicio

Imprimir un texto en todas las impresoras conectadas al equipo

```
(Get-CimInstance Win32_Printer).name | %{"Texto"| Out-Printer}
```

Placa base y otros componentes

Cmdlet con llamada CIM que muestra información sobre el equipo

```
Get-CimInstance Win32_ComputerSystem
```

Cmdlet con llamada CIM que muestra información sobre el fabricante del equipo

```
Get-CimInstance Win32_SystemEnclosure
```

Cmdlets con llamadas CIM que muestran información sobre la placa base

```
Get-CimInstance Win32_BaseBoard
Get-CimInstance Win32_MotherboardDevice
```

Ejemplos

Fabricante de placa base

```
(Get-CimInstance Win32_BaseBoard).Manufacturer
```

Identificación del equipo

```
(Get-CimInstance Win32_SystemEnclosure).SerialNumber
```

BIOS

Cuando un ordenador se enciende, generalmente el procesador busca la BIOS y la ejecuta, la BIOS (Basic Input-Output System) no sólo es el primer paso del proceso de arranque, sino que también proporciona una interfaz de bajo nivel para dispositivos periféricos. La BIOS comprueba los dispositivos hardware conectados al ordenador y localiza un dispositivo con el que arrancar el sistema.

En los ordenadores que utilizan la BIOS, una vez que ésta ha comprobado los dispositivos hardware, intenta localizar un dispositivo con el que arrancar. Cuando el dispositivo que arranca es el disco duro, la BIOS carga en memoria el programa que resida en el primer sector de este dispositivo, ese programa se llama MBR (Master Boot Record, Registro maestro de arranque).

A parte del MBR también se puede cargar la tabla de partición GUID (GPT), en la actualidad el MBR ha sido sustituido por la tabla de partición GUID.

Cmdlets con llamadas CIM que muestran información sobre la BIOS

```
Get-CimInstance Win32_BIOS
Get-CimInstance Win32_SystemBIOS
```

Ejemplo

Información de la BIOS

```
(Get-CimInstance Win32_BIOS).Version
```

Ejercicio

Si la versión de la BIOS del equipo se encuentra dentro del fichero, hay que actualizarla

```
#Fichero bios.txt contiene _ASUS_ - 1072009
"_ASUS_ - 1072009" | Out-File bios.txt -Append

gc .\bios.txt | %{
if((Get-CimInstance Win32_BIOS).version -eq $_)
{
"Actualizar BIOS del equipo"+(hostname)
}
```

Batería

Cmdlets con llamadas CIM que muestran información sobre la batería

```
Get-CimInstance Win32_Battery
Get-CimInstance Win32_CurrentProbe
Get-CimInstance Win32_PortableBattery
Get-CimInstance Win32_PowerManagementEvent
Get-CimInstance Win32_VoltageProbe
```

Ejemplo

Porcentaje de carga de batería restante

```
(Get-CimInstance Win32_Battery).EstimatedChargeRemaining
```

Ejercicio

Comprobar si se utiliza la batería o la corriente alterna

```
if((Get-CimInstance -Class Win32_Battery).BatteryStatus -eq 1)
{
    'Batería interna activada'
}
else
{
    'Utilizando de corriente alterna'
}
```

Refrigeración

Cmdlets con llamadas CIM que muestran información sobre la refrigeración del equipo

```
Get-CimInstance Win32_Fan
Get-CimInstance Win32_HeatPipe
Get-CimInstance Win32_Refrigeration
Get-CimInstance Win32_TemperatureProbe
Get-CimInstance MSAcpi_ThermalZoneTemperature -Namespace "root/wmi"
```

Ejemplo

Temperatura de CPU (hay que ejecutar PowerShell como administrador)

```
$Temperatura = Get-CimInstance MSAcpi_ThermalZoneTemperature -Namespace "root/wmi"
$Kelvin = $Temperatura.CurrentTemperature / 10
$Celsius = $Kelvin - 273.15
$Celsius.ToString() + "C "
```

Ejercicios

Obtener la mayor cantidad de información sobre el hardware

```
(Get-CimInstance Win32_BaseBoard).Manufacturer
(Get-CimInstance Win32_BIOS).Version
(Get-CimInstance Win32_Bus).DeviceID
Get-CimInstance Win32_ComputerSystem
Get-CimInstance Win32_Processor
Get-CimInstance Win32_SystemEnclosure
Get-CimInstance Win32_Battery
Get-CimInstance Win32_Printer
```

Obtener información sobre el hardware creando un objeto con todos los datos

```
#Llamadas CIM
$ComputerSystem=Get-CimInstance Win32_ComputerSystem
$BaseBoard=Get-CimInstance Win32_BaseBoard
$BIOS=Get-CimInstance Win32_BIOS
$Processor=Get-CimInstance Win32_Processor
$Battery=Get-CimInstance Win32_Battery

#Crear un objeto con todos los datos sobre el hardware
[PSCustomObject]@{
 Model = $ComputerSystem.Model
 ManufacturerBoard = $BaseBoard.Manufacturer
 BIOSVersion = $BIOS.SMbiosbiosversion
 BIOSSerialNumber = $BIOS.serialnumber
 ManufacturerProcessor=$Processor.Manufacturer
 MaxClockSpeed=$Processor.MaxClockSpeed
 DeviceIDBattery=$Battery.DeviceID.trim()
}
```

Crear una estructura de directorios con información hardware

```
#Leer del fichero el componente hardware y la llamada CIM que
hay que realizar, ejemplo
#procesador,Win32_Processor
```

```
#Guardar la información sobre la llamada CIM dentro del fichero que tiene como nombre la primera parte de cada línea del fichero

gc .\hardware.txt | %{
$linea=$_.split(",")
mkdir $linea[0]
cd $linea[0]
Get-CimInstance $linea[1] | Out-File $linea[0]
cd ..
}
```
