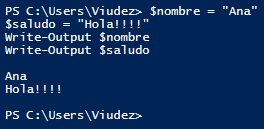
##################################################################### #PARTE1: VARIABLES, PARÁMETROS, ESTRUCTURAS SELECTIVAS Y REPETITIVAS #####################################################################

1. Crea un script que defina dos variables**:** un **$nombre** y un **$saludo**. Luego muestra un

mensaje en el que se muestre ese saludo y ese nombre.

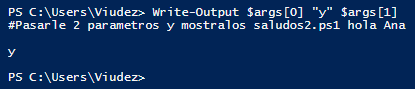
**$nombre =** "Ana"

**$saludo =** "Hola!!!!" Write-Output **$nombre** Write-Output **$saludo**

****

1. Crea un script llamado saludo2.ps1 que muetre 2 parámetros pasados Write-Output **$args**[0] "y" **$args**[1]

#Pasarle 2 parametros y mostralos saludos2.ps1 hola Ana



1. Crea un scipt pero debe solicitar el nombre y el saludo write-output "Introduce un nombre"

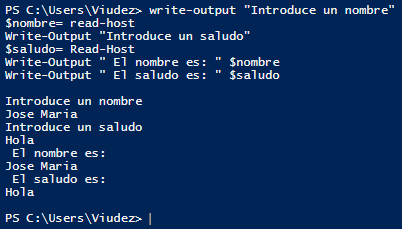
**$nombre=** read-host

Write-Output "Introduce un saludo"

**$saludo=** Read-Host

Write-Output " El nombre es: " **$nombre**

Write-Output " El saludo es: " **$saludo**

****

1. Crea un script que recoja dos variables enteras, lleve a cabo todas las operaciones aritméticas y las muestre.

#pide 2 numeros y muestra las operaciones.

Write-Output "Introduce 2 numeros para hacer operaciones" Write-Output "el primer numero: "

**$uno =** Read-Host

Write-Output "el primer numero: "

**$dos =** Read-Host

**$resultado =** [**int**]**$uno + $dos**

Write-Output "La suma es: "**$resultado**

**$resultado =** [**int**]**$uno - $dos**

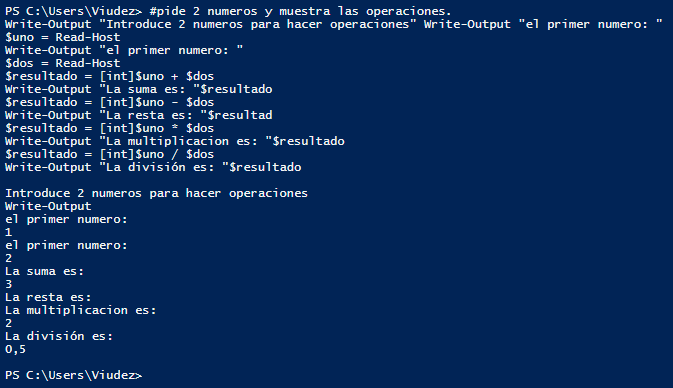
Write-Output "La resta es: "**$resultad**

**$resultado =** [**int**]**$uno \* $dos**

Write-Output "La multiplicacion es: "**$resultado**

**$resultado =** [**int**]**$uno / $dos**

Write-Output "La división es: "**$resultado**

****

1. Crea un script que solicite dos números enteros y muestre SI uno es mayor, menor o igual que el otro.

#Introducir dos numeros y mostar cual es mayor, menor o si son iguales Write-Output "Introduce el primer número"

**$numero1 =** Read-Host

Write-Output "Introduce el segundo número"

**$numero2 =** Read-Host

**if**(**$numero1 -eq $numero2**){

Write-Output "los numeros son iguales"

}

**if**(**$numero1 -gt $numero2**){

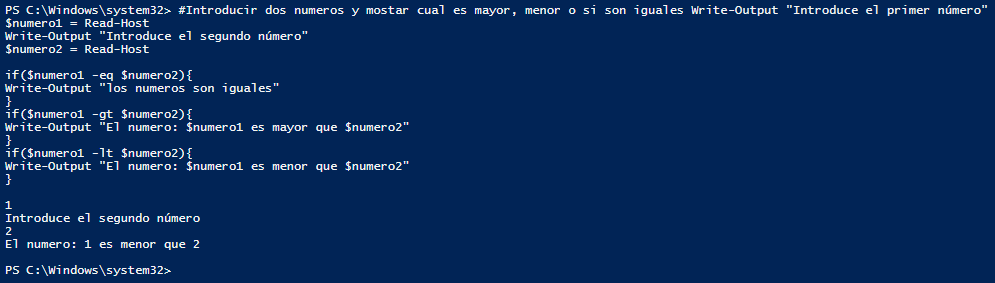
Write-Output "El numero: $numero1 es mayor que $numero2"

}

**if**(**$numero1 -lt $numero2**){

Write-Output "El numero: $numero1 es menor que $numero2"

}



1. Crea un script que solicite un número, verifique que es positivo y programa un bucle para que muestre por consola la palabra FAP tantas veces como indique el número.

#Introducir un numero y mostar un mensaje tantas veces como sea el numero introducido. Write-Output "Introduce el primer número"

**$numero1 =** Read-Host

**if**(**$numero1 -gt** 0){

**while** (**$numero1 -gt** 0){ Write-Output "FAP"

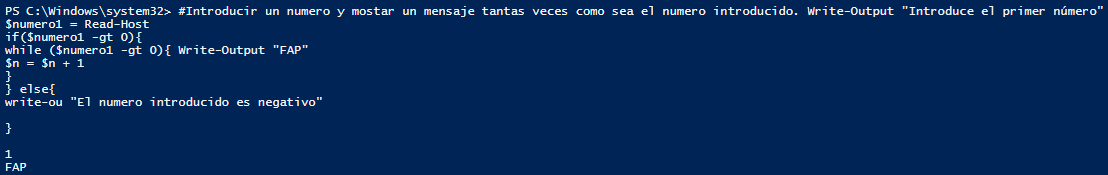
**$n = $n +** 1

}

} **else**{

write**-**ou "El numero introducido es negativo"

}



1. Crea un script que solicite un número. Mientras no esté entre 1 y 100 se solicita otra vez.

Una vez introducido el número correcto**:** si ha cometido algún error al introducir un

número válido debe hacerse un bucle en el que se increpe al usuario tantas veces como errores haya cometido. Si lo hizo bien a la primera saca un mensaje que diga**:** campeón.

#Introducir un numero entre el 1 y el 100, si ha cometido algun error crear #un bucle tantas veces como erroes...

**$salir =** 0

**$bucle =** 0

**while** (**$salir -eq** 0){

Write-Output "Introduce un numero entre el 1 y el 100"

**$numero =** Read-Host

**if** ((**$numero -gt** 0) **-and** (**$numero -lt** 100)){

**$salir=**1

}**else**

**$bucle=$bucle+**1

}

**if** (**$bucle -eq** 0){

Write-Output "Eres un crack"

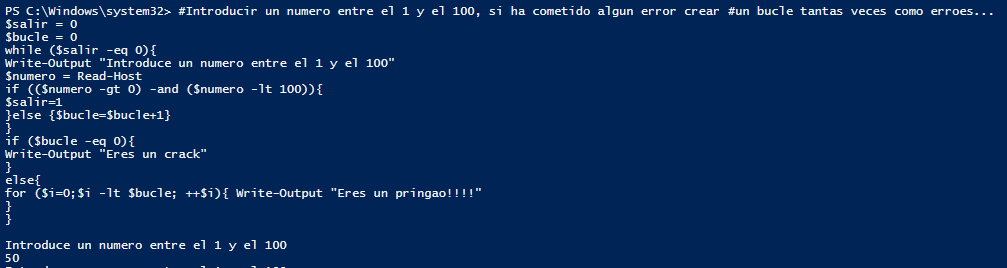
}

**else**{

**for** (**$i=**0;**$i -lt $bucle**; **++$i**){ Write-Output "Eres un pringao!!!!"

}

}



1. Crea un script para crear 3 usuarios . Debe pedir por consola los datos**:** al menos el password de usuario.

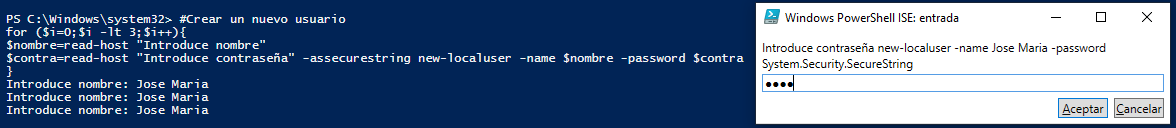
#Crear un nuevo usuario

**for** (**$i=**0;**$i -lt** 3;**$i++**){

**$nombre=**read-host "Introduce nombre"

**$contra=**read-host "Introduce contraseña" **-**assecurestring new-localuser -name **$nombre -**password **$contra**

}



1. Crea un script que espere tres parámetros un llamado **$primero** que espere un [**int**], otro llamado **$segundo** que espere un **String** que si no se introduce debe solicitarlo al usuario. Una vez hecho el programa debe hacer un bucle **for** que muestre el valor de **$segundo** tantas veces como indique **$primero**.

**param** (

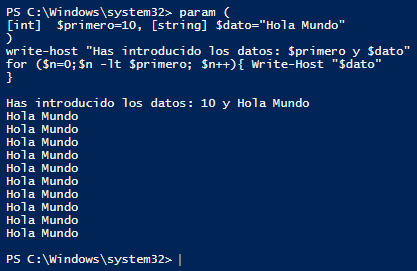
[**int**] **$primero=**10, [**string**] **$dato=**"Hola Mundo"

)

write-host "Has introducido los datos: $primero y $dato"

**for** (**$n=**0;**$n -lt $primero**; **$n++**){ Write-Host "$dato"

}



1. Crea un script llamado Get**-**España.ps1 con parámetros que permita indicar los puestos de un equipo de fútbol y luego los muestre.

Algo así**:**

Get**-**España.ps1 **-**portero 'Casillas' **-**defensa 'Sergio Rambo' **-**pivote 'Busquets' **-**medio 'Javier Alonso' **-**delantero 'Erroberto Gudari'

Alineación de España**:** Portero**:** Casillas Defensa**:** Sergio Rambo Busquets**:** Busquets Centro**:** Javier Alonso Punta**:** Erroberto Gudari

**param** (

[**string**] **$seleccion=**"",

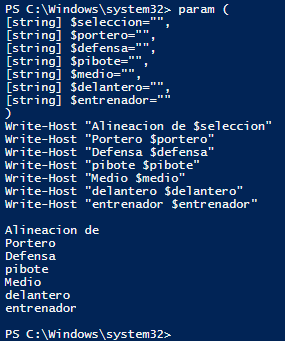
[**string**] **$portero=**"", [**string**] **$defensa=**"", [**string**] **$pibote=**"", [**string**] **$medio=**"", [**string**] **$delantero=**"", [**string**] **$entrenador=**""

)

Write-Host "Alineacion de $seleccion" Write-Host "Portero $portero"

Write-Host "Defensa $defensa" Write-Host "pibote $pibote" Write-Host "Medio $medio"

Write-Host "delantero $delantero" Write-Host "entrenador $entrenador"



############################################### PARTE2**:** MENÚS ###############################################

11.**-** Crea un script que muestre un MENÚ al usuario con cuatro operaciones disponibles**:** crear directorio, eliminar directorio, mostrar directorio y salir.

Una vez elegida una opción correcta el programa debe solicitar un nombre de directorio y hacer la operación seleccionada.

Operaciones**:**

Crear**:** New-Item -Name Carpeta **-**ItemType folder Borrar**:** Remove-Item Carpeta

Dir**:** Get-ChildItem Carpeta

#menú para directorios

**$Salir=**0

**while** (**$salir -eq** "0"){ write-output "Menu"

Write-Output " " Write-Output "1 Crear directorio"

Write-Output "2 Borrar Directorio" Write-Output "3 listar directorio" Write-Output "4 Salir"

Write-Output "Introduce una de las opciones: "

**$menu=** Read-Host

**$sal=**"0"

**switch** (**$menu**) {

* 1. { Write-Output "Crear directorio"

**while** (**$sal -eq** "0"){

Write-Output "Introducir el directorio donde queremos crear"

**$directorio=** Read-Host

**if** (Test-Path **$directorio**){

Write-Output "El directorio existe"

}

**else**{

New-Item -Name **$directorio -**Type d

**$sal=**"1"

}

}

}

* 1. { Write-Output "Borrar Directorio"

**while** (**$sal -eq** "0"){

Write-Output "Introducir el directorio donde queremos borrar"

**$directorio=** Read-Host

**if** (Test-Path **$directorio**){ Remove-Item **$directorio**

**$sal=**"1"

}

**else**{

Write-Output "El directorio no existe"

}

}

}

* 1. { Write-Output "listar directorio"

**while** (**$sal -eq** "0"){

Write-Output "Introducir el directorio donde queremos Listar"

**$directorio=** Read-Host

**if** (Test-Path **$directorio**){ dir **$directorio**

**$sal=**"1"

}

**else**{

Write-Output "El directorio no existe"

}

}

}

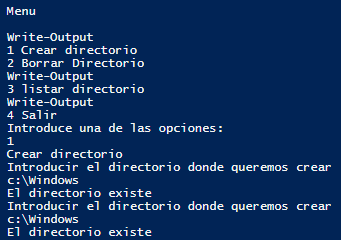
* 1. { Write-Output "Gracias por usar este programa"

**$Salir=**1

}

}

}



############################################################ PARTE3**:** ARRAYS ############################################################

1. Desarrolla un script que defina un **array** de 10 nombres

y los muestre por pantalla con un **while**, con un **for** y con un foreach. #Array que muestre 10 nombres con un for, un foreach y un while

**$nombre=** "Carlos","Mikel","Ion","Marta","Ana","Juan","Fermin","Amaya","Sandra","JJ"

Write-Output "Mostrar un array de tres maneras diferentes" Write-Output "Con un FOR"

Write-Output " "

**for**(**$i=**0;**$i -lt $nombre**.Length;**$i++**){ Write-Output **$nombre**[**$i**]

}

Write-Output "Con un FOREACH" Write-Output " " foreach (**$i in $nombre**){

Write-Output **$i**

}

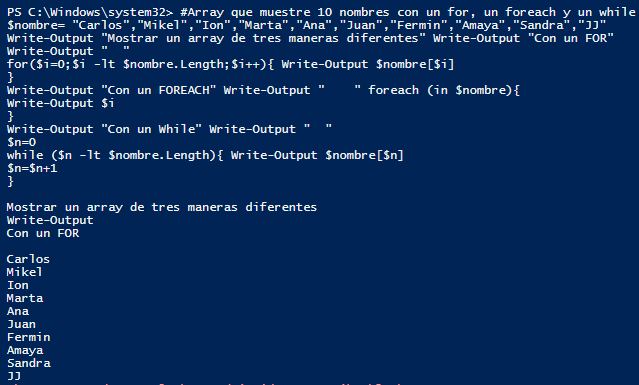
Write-Output "Con un While" Write-Output " "

**$n=**0

**while** (**$n -lt $nombre**.Length){ Write-Output **$nombre**[**$n**]

**$n=$n+**1

}



1. Desarrolla script que defina un **array** de 10 números inicializados a 0.

A continuacíon haz un bucle (**for** o foreach) que inicialice el **array** con números enteros aleatorios

entre 10 y **-**10. Debes investigar el cmdlet Get-Random para conseguir esos números.

# Array de 10 numeros aleatorios entre 10 y -10

**$numeros=**0,0,0,0,0,0,0,0,0,0

foreach(**$n in $numeros**){

**$numeros**[**$n**]**=** Get-Random 10 **-**Minimum **-**10 Write-Output **$numeros**[**$n**]

}

1. Bucle **for** que contabilice cuántos numeros positivos,negativos e iguales a 0 hay en el

**array** anterior

# Array de 10 numeros aleatorios entre 10 y -10

**$numeros=**0,0,0,0,0,0,0,0,0,0

**$pos=**0

**$neg=**0

**$igu=**0

#generar numeros aleatorios entre 10 y -10 foreach(**$n in $numeros**){

**$numeros**[**$n**]**=** Get-Random 10 **-**Minimum **-**10 Write-Output **$numeros**[**$n**]

**if** (**$numeros**[**$n**]**-ge** 0 **-and $numeros**[**$n**]**-eq** 0){

**$pos=$pos+**1

}

**if** (**$numeros**[**$n**]**-lt** 0){

**$neg=$neg+**1

}

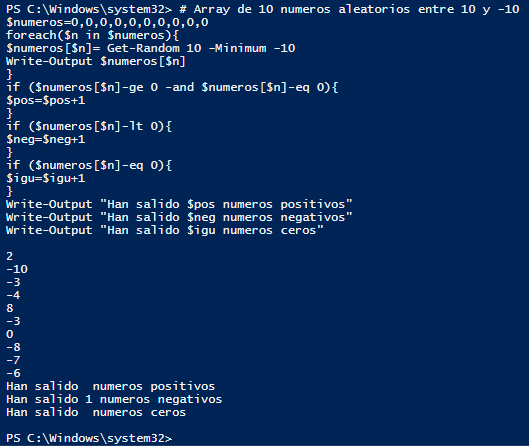
**if** (**$numeros**[**$n**]**-eq** 0){

**$igu=$igu+**1

}

}

Write-Output "Han salido $pos numeros positivos" Write-Output "Han salido $neg numeros negativos" Write-Output "Han salido $igu numeros ceros"



1. Desarrolla un script que solicite un nombre de directorio.Debe crear ese directorio,meterse en él y crear 10 directorios cuyo nombre debe ser un número aleatorio, y dentro de cada uno de ellos debe crear 10 ficheros de texto cuyo nombre debe ser un número aleatorio.

#Introducir un nuevo directorio, crear 10 dentro de este con nombres que sean numeros aleatorios.

#Dentro de cada nuevo directorio crear 10 ficheros de texto.

**$i=**0

#while existe el directorio

**while** (**$i -eq** 0){

Write-Output "Introduce el nuevo directorio"

**$direc=** Read-Host

**if** (Test-Path **$direc**){

Write-Output "El directorio ya existe."

}

**else** { **$i=**1 }

}

New-Item -Name **$direc -**Type d cd **$direc**

# crear los directorios y los ficheros dentro de cada directorio

**for** (**$n=**0;**$n -lt** 10;**$n++**){

**$direc1=**Get-Random 400

New-Item -Name **$direc1 -**type d

Write-Output " "

Write-Output **$direc1**

Write-Output " " cd **$direc1**

**for** (**$h=**0;**$H -lt** 10;**$h++**){

**$fich=**Get-Random 400

New-Item -Name **$fich**".txt" **-**type f

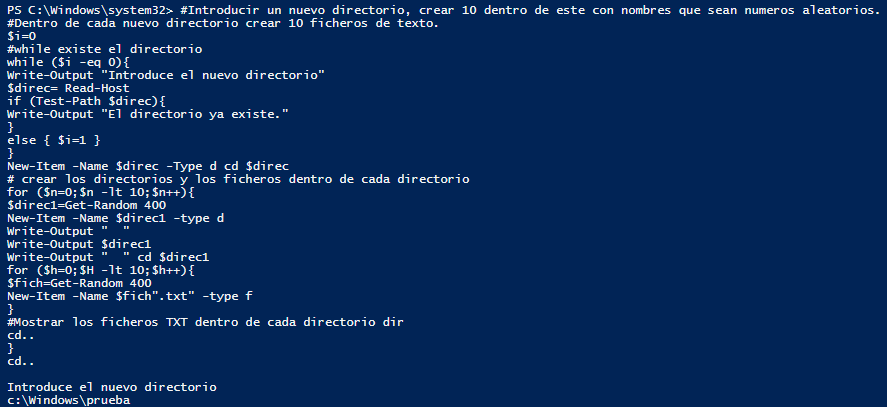
}

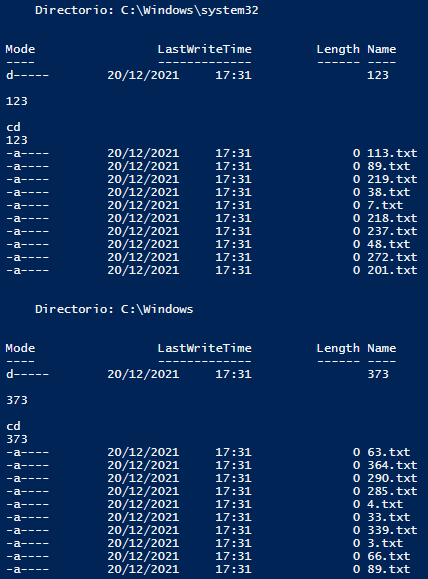
#Mostrar los ficheros TXT dentro de cada directorio dir

cd..

}

cd..





1. Crea un script que recoja como parámetro un nombre de DIRECTORIO.

Verifica que ese directorio exista. Listar los contenidos del directorio ordenados por tamaño, y de cada elemento debes sacar el tamaño y el nombre.

**$i=**0

**while** (**$i -eq** 0){

Write-Output "Introduce el directorio"

**$direc=** Read-Host

**if** (Test-Path **$direc**)

{ **$i=**1 }

**else**

{ Write-Output "El directorio no existe." }

}

#muestra el contenido del directorio, solo el tamaño y el nombre y lo ordena por tamaño. cd **$direc**

Get-ChildItem | Select-Object length, name | Sort-Object **-**Descending length

cd..



1. Utilizando arrays y números aleatorios, desarrolla un script que genere 10 contraseñas de 6 caracteres

Añádele parámetros para poder indicarle longitud concreta. #Creo un array con caracteres.

**$caracteres=**"a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","0","1","2","3","4",

# creaoms los 10 nombres de 6 caracteres aleatorios

**for** (**$h=**0;**$h -lt** 10;**$h++**){ [**string**]**$nombre=**"" **for**(**$i=**0;**$i -lt** 7;**$i++**){

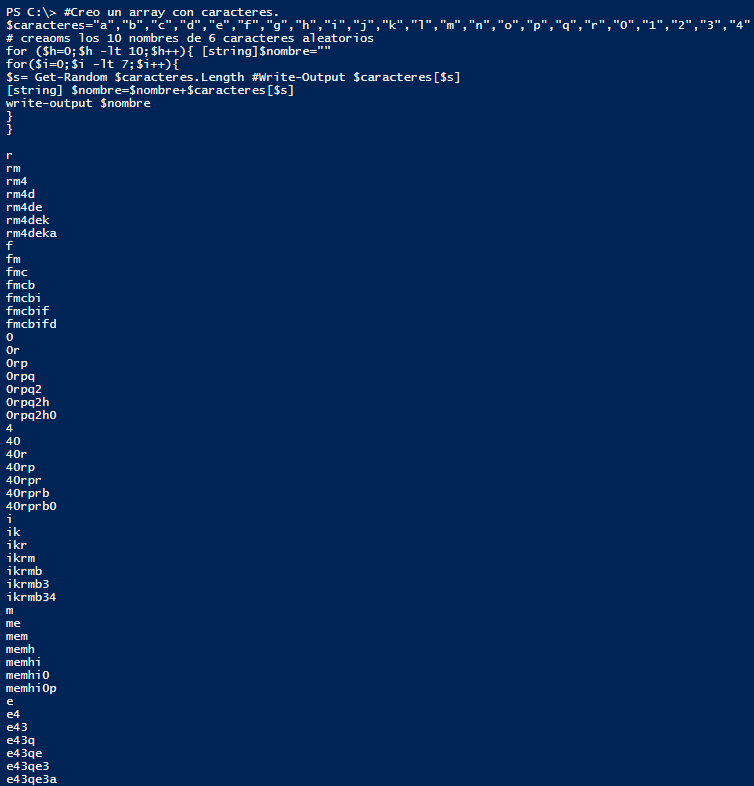
**$s=** Get-Random **$caracteres**.Length #Write-Output $caracteres[$s]

[**string**] **$nombre=$nombre+$caracteres**[**$s**]

write-output **$nombre**

}

}



1. Crea un script llamado CHECKPORTS que defina un **array** con los puertos que queremos que estén en listening.

Ejecuta el comando para sacar los puertos y verifica que están abiertos los indicados en el **array**.

Muestra un mensaje que indique qué puertos están correctamente en estado listening y cuáles no.

#Mostar los puertos abiertos

**$Puertos=**Get-NetTCPConnection #| Where-Object -FilterScript {$\_.state -eq "listen"} #Write-Output $Puertos

Write-Output "Los puertos que estan abiertos son:"

**for**(**$i=**0;**$i -lt $puertos**.length;**$i++**){ **if** (**$Puertos**[**$i**].State **-eq** "listen"){

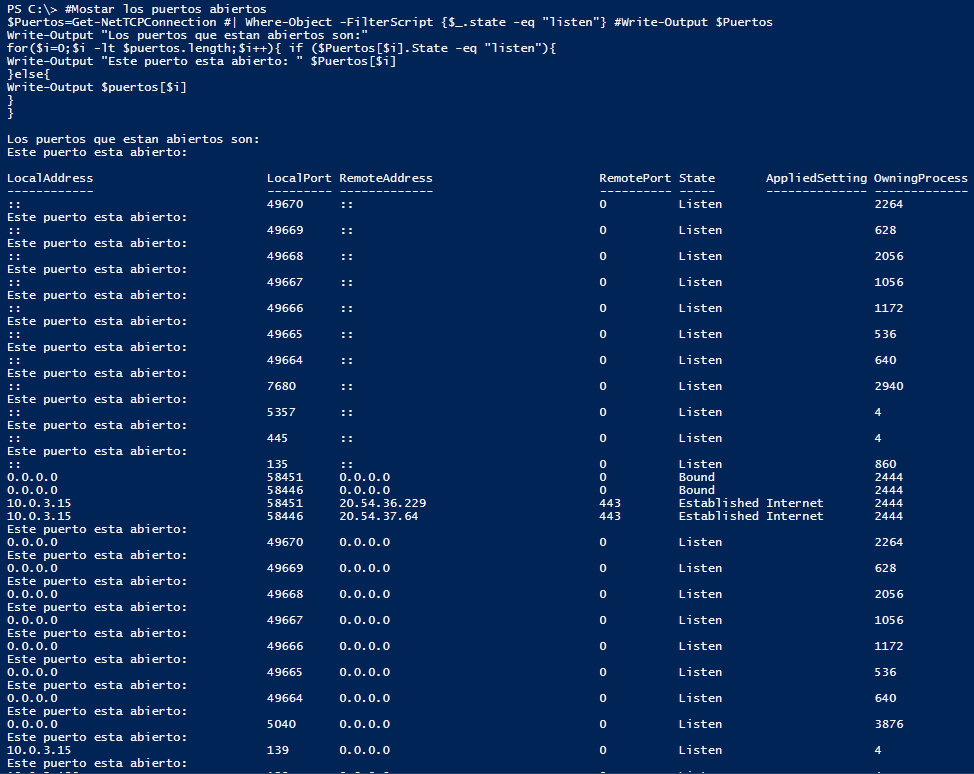
Write-Output "Este puerto esta abierto: " **$Puertos**[**$i**]

}**else**{

Write-Output **$puertos**[**$i**]

}

}



1. Crea un script llamado ALERTDISK que defina un **array** con las UNIDADES y el límite de capacidad.

Saque el tamaño ocupado en las unidades y que nos avise si se ha superado el límite impuesto

#Write-Output "Introduce el espacio minimo que tiene que tener libre (GB): " #$espacio= Read-Host

#$espacio=$espacio\*1000000 #Write-Output $espacio

**$Unidades=**Get-PSDrive | select root, used, free

Write-Output **$Unidades**.Length

**for** (**$unid=**0;**$unid -lt $Unidades**.Length;**$unid++**) {

$tamaño**=$Unidades**[**$unid**].free #Write-Output $tamaño

**if** (**$Unidades**[**$unid**].root **-ne** "" **-and $Unidades**[**$unid**].root **-ne** "\"){

**if** (**$Unidades**[**$unid**].free **-lt** 2000000){

Write-Output **$Unidades**[**$unid**].root "tiene poco espacio libre. Espacio libre: "**$Unidades**[**$unid**].free

}

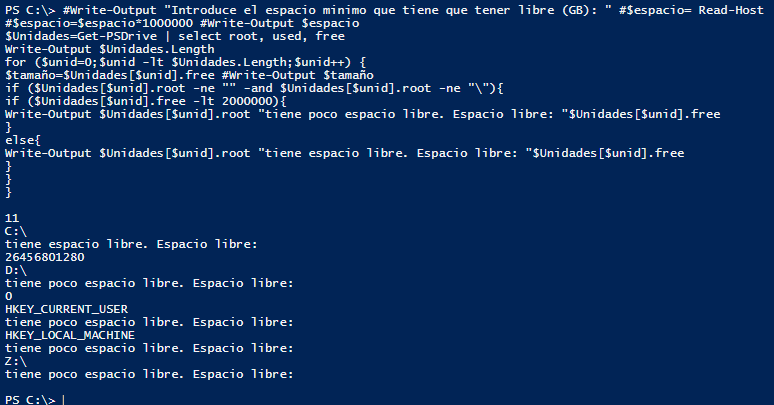
**else**{

Write-Output **$Unidades**[**$unid**].root "tiene espacio libre. Espacio libre: "**$Unidades**[**$unid**].free

}

}

}



1. Crea un script que defina un **array** con USUARIOS.

Los campos de cada elemento deben ser nombre\_cuenta, nombre y password. El programa debe mantener ese **array** con un menú en el que se pueda**:**

**-**añadir

**-**eliminar

**-**buscar por nombre

**-**mostrar todos

Al crear uno nuevo deben solicitarse datos y el campo password debe ocultarse al escribir.

#Gestion de usuario

**$tabla\_usuario=**@{ "Juan"**=**@{nombre**=**"juan";cuenta**=**"juan";contrasena**=**"12345678"}; "Pablo"**=**@{nombre**=**"Pablo";cuenta**=**"Pablo";contrasena**=**"12345678"}; "Ana"**=**@{nombre**=**"Ana";cuenta**=**"Ana";contrasena**=**"12345678"}

}

**$valor=**0

**while** (**$valor -ne** 5){

Write-Output "Menu de gestion de un usuario"

Write-Output " " Write-Output " 1 Crear un usuario"

Write-Output " 2 Borrar un usuario" Write-Output " 3 Buscar un usuario"

Write-Output " 4 Mostar todos los usuarios" Write-Output " 5 Salir"

**$valor=**Read-Host

**switch**(**$valor**){

* 1. { Write-Output " Crear usuario"

Write-Output " " # $verdadero=0

#comprobamos que el nuevo usuario no existe #Introducimos los datos en la tabla de usuarios Write-Output "Introduce el nombre del usuario: "

**$usuario=**Read-Host

Write-Output "Introduce la cuenta de usuario:"

**$cuenta=**Read-Host

Write-Output "Introduce la contraseña de usuario:"

**$contra=**Read-Host

**$tabla\_usuario=**@{**$usuario=**@{nombre**=$usuario**;cuenta**=$cuenta**;contrasena**=$contra**}}

}

* 1. { Write-Output " Borrar usuario"

Write-Output " " write-output "Introduce el usuario que quieres eliminar."

**$usuario=** read-host

#Write-Output $tabla\_usuario[$usuario].Values

**if**(**$tabla\_usuario**.ContainsKey(**$usuario**)) { write-output " El usuario se ha eliminado:"

**$tabla\_usuario**.Remove(**$usuario**)

}

**else**{Write-Output "El usuario introducido no existe"

}

}

3{

Write-Output " Buscar usuario"

Write-Output " " write-output "Introduce el usuario que quieres buscar."

**$usuario=** read-host

**if**(**$tabla\_usuario**.ContainsKey(**$usuario**)) { write-output " El usuario buscado es:"

Write-Output **$usuario**.key

write-output $(**$usuario**.Value["cuenta"]) write-output $(**$usuario**.Value["contrasena"])

Write-Output " "

}

**else**{

Write-Output "El usuario introducido no existe"

}

}

4{

Write-Output " Mostar usuarios"

Write-Output " "

foreach(**$usuario in $tabla\_usuario**.GetEnumerator()){

Write-Output **$usuario**.key

write-output $(**$usuario**.Value["cuenta"]) write-output $(**$usuario**.Value["contrasena"])

Write-Output " "

}

}

}

Write-Output " Has salido del programa."

